

# DESIGN OF HIGH-PERFORMANCE MICROPROCESSOR CIRCUITS

Edited by

Anantha Chandrakasan

William J. Bowhill

Frank Fox

# **DESIGN OF HIGH-PERFORMANCE MICROPROCESSOR CIRCUITS**

IEEE Press  
445 Hoes Lane, P.O. Box 1331  
Piscataway, NJ 08855-1331

**IEEE Press Editorial Board**  
Robert J. Herrick, *Editor in Chief*

M. Akay	M. Eden	M. S. Newman
J. B. Anderson	M. E. El-Hawary	M. Padgett
P. M. Anderson	R. F. Hoyt	W. D. Reeve
J. E. Brewer	S. V. Kartalopoulos	G. Zobrist
	D. Kirk	

Kenneth Moore, *Director of IEEE Press*  
Catherine Faduska, *Senior Acquisitions Editor*  
Robert Bedford, *Assistant Acquisitions Editor*  
Marilyn G. Catis, *Marketing Manager*  
Anthony VenGraitis, *Project Editor*

Cover design: Sharon Klein, *Sharon Klein Graphic Design*

#### **Technical Reviewers**

Dr. Stuart K. Tewksbury, *Stevens Technical Institute, Hoboken, NJ*

Dr. Lewis Terman, *IBM Corporation, Yorktown Heights, NY*

Brock Barton, *Texas Instruments, Dallas, TX*

R. Jacob Baker, *University of Idaho, Boise, ID*

#### **Books of Related Interest from the IEEE Press**

##### ***LOW-POWER CMOS DESIGN***

Edited by Anantha Chandrakasan and Robert Broderson

1998 Hardcover 644 pp IEEE Order No. PC5703 ISBN 0-7803-3429-9

##### ***CMOS: Circuit Design, Layout, and Simulation***

R. Jacob Baker, Harry W. Li, and David E. Boyce

1998 Hardcover 944 pp IEEE Order No. PC5689 ISBN 0-7803-3416-7

##### ***HIGH-PERFORMANCE SYSTEM DESIGN: Circuits and Logic***

Edited by Vojin G. Oklobdzija

1999 Hardcover 560 pp IEEE Order No. PC5765 ISBN 0-7803-4716-1

##### ***INTEGRATED CIRCUITS FOR WIRELESS COMMUNICATIONS***

Edited by Asad A. Abidi, Paul R. Gray, and Robert G. Meyer

1999 Hardcover 688 pp IEEE Order No. PC5716 ISBN 0-7803-3459-0

# **DESIGN OF HIGH-PERFORMANCE MICROPROCESSOR CIRCUITS**

**Anantha Chandrakasan**

*Massachusetts Institute of Technology  
Cambridge, MA*

**William J. Bowhill**

*Compaq Computer Corporation  
Shrewsbury, MA*

**Frank Fox**

*Rambus Inc.  
Mountain View, CA*



The Institute of Electrical and Electronics Engineers, Inc., New York

This book and other books may be purchased at a discount from the publisher when ordered in bulk quantities. Contact:

**IEEE Press Marketing**  
Attn: Special Sales  
445 Hoes Lane  
P.O. Box 1331  
Piscataway, NJ 08855-1331  
Fax: +1 732 981 9334

For more information about IEEE Press products, visit the IEEE Online Catalog Store at <http://www.ieee.org/ieeestore>.

© 2001 by the Institute of Electrical and Electronics Engineers, Inc.  
3 Park Avenue, 17<sup>th</sup> Floor, New York, NY 10016-5997

*All rights reserved. No part of this book may be reproduced in any form,  
nor may it be stored in a retrieval system or transmitted in any form,  
without written permission from the publisher.*

Printed in the United States of America.

10      9      8      7      6      5      4      3      2      1

**ISBN 0-7803-6001-X**  
**IEEE Order No. PC5836**

**Library of Congress Cataloging-in-Publication Data**

Design of high-performance microprocessor circuits / Anantha Chandrakasan,  
William J Bowhill, Frank Fox, editors.

p. cm.

Includes bibliographical references and index.

**ISBN 0-7803-6001-X**

1. Microprocessors – Design and construction. 2. Logic circuits. I. Chandrakasan,  
Anantha P. II. Bowhill, William J-III. Fox, Frank, 1952

TK7895.M5 D47 2000

621.3815–dc21

00-036977

# CONTENTS

**Preface**      **xix**

**PART I    OVERVIEW**      **1**

**CHAPTER 1    IMPACT OF PHYSICAL TECHNOLOGY ON  
ARCHITECTURE**      **3**

*John H. Edmondson*

1.1 Introduction	3
1.1.1 Suitability of CMOS Technology	3
1.1.2 Physical Technology Impact at Various Scales	4
1.1.3 RISC and the Importance of Well-Chosen Architecture	4
1.1.4 Key High-Level Decisions and Trade-Offs	5
1.1.5 Other Technology Issues	5
1.2 Implementing Processor Architecture in CMOS Technology	6
1.2.1 Dynamic Logic	6
1.2.2 Advanced Logic Styles	9
1.2.3 Dynamic Logic Examples from High-Performance Processor Designs	9
1.2.4 Datapaths	12
1.2.5 RAMs	12
1.3 Choosing the Cycle Time of a High-Performance Microprocessor	14
1.3.1 Critical Loops	15
1.4 Comparison of PA8000, 21164, and 21264 Processors	15
1.5 Trend in Interconnect Resistance	17
1.5.1 Interconnect Inductance	17
1.5.2 Architectural Consequences of Interconnect Trends	18
1.6 Trend in Power Consumption	18
1.6.1 Power Estimation and Scaling	18
1.6.2 Energy-Delay Product	20
1.6.3 Physical Limits of Heat Dissipation	20
1.6.4 Active Power Control	21
1.6.5 $V_{dd}$ Reduction	21
1.6.6 Ultimate Impact of Power	21
1.7 Advanced Packaging	21
1.8 Conclusion	23
References	23

**PART II TECHNOLOGY ISSUES 25****CHAPTER 2 CMOS SCALING AND ISSUES IN SUB-0.25 μm SYSTEMS 27***Yuan Taur*

2.1 MOSFET Scaling Theory	27
2.1.1 Constant-Field Scaling	27
2.1.2 Two-Dimensional Scale Length Theory	29
2.2 CMOS Scaling Issues below 0.25 μm	31
2.2.1 Power Supply and Threshold Voltage	31
2.2.2 Gate Oxides	35
2.2.3 Channel Profile Design	36
2.3 Interconnect RC Delay	39
2.3.1 Interconnect Scaling	39
2.3.2 Global Wire Issues	41
2.4 Low-Temperature CMOS	42
References	44

**CHAPTER 3 TECHNIQUES FOR LEAKAGE POWER REDUCTION 46***Vivek De, Yibin Ye, Ali Keshavarzi, Siva Narendra, James Kao, Dinesh Somasekhar, Raj Nair, and Shekhar Borkar*

3.1 Introduction	46
3.2 Transistor Leakage Current Components	47
3.2.1 p-n Junction Reverse Bias Current ( $I_1$ )	48
3.2.2 Weak Inversion ( $I_2$ )	49
3.2.3 DIBL ( $I_3$ )	49
3.2.4 GIDL ( $I_4$ )	49
3.2.5 Punchthrough ( $I_5$ )	50
3.2.6 Narrow Width Effect ( $I_6$ )	50
3.2.7 Gate Oxide Tunneling ( $I_7$ )	50
3.2.8 Hot Carrier Injection ( $I_8$ )	51
3.3 Circuit Subthreshold Leakage Current	52
3.3.1 Transistor Stack Effect	52
3.3.2 Steady-State Leakage Model of Transistor Stacks	53
3.3.3 Transient Model of Transistor Stack Leakage	55
3.4 Leakage Control Techniques	55
3.4.1 Standby Leakage Control by Input Vector Activation	55
3.4.2 Embedded Dual- $V_t$ Design for Domino Circuits	57
3.4.3 Adaptive Body Biasing	58
Acknowledgments	61
References	61

**CHAPTER 4 LOW-VOLTAGE TECHNOLOGIES 63***Tadahiro Kuroda and Takayasu Sakurai*

4.1 Low-Voltage Low-Threshold-Voltage Circuit Design	63
4.1.1 Power, Energy, and Speed	63
4.1.2 Low- $V_{DD}$ , Low- $V_{TH}$ Design Space	64
4.1.3 Design Issues at Low $V_{DD}$ and Low $V_{TH}$	65
4.2 Power-Down Scheme	66
4.3 Controlling $V_{TH}$ through Substrate Bias	67
4.3.1 Variable Threshold-Voltage CMOS (VTCMOS) Technology	67
4.3.2 VTCMOS Circuit Techniques	69
4.3.3 VTCMOS Performance and Penalty	71
4.4 Processor Design Examples	75
4.4.1 TX3900	75
4.4.2 SH-4	77
4.5 Conclusion	78
References	79

**CHAPTER 5 SOI TECHNOLOGY AND CIRCUITS 80***Ghavam G. Shahidi, Fari Assaderaghi,  
and Dimitri Antoniadis*

5.1 Introduction	80
5.2 Device Design Considerations for PD SOI versus FD SOI	81
5.3 Device Results	83
5.3.1 SOI Device Self-Heating	83
5.3.2 $V_T$ Variation Due to Floating Body	84
5.4 PD-SOI CMOS Digital Circuits	87
5.4.1 History Dependence of Propagation Delay	87
5.4.2 “Pass-Gate” Transient Leakage	90
5.4.3 Circuit Issues in PD-SOI CMOS	91
5.5 SOI for Low Power	95
5.6 Conclusion	96
References	96

**CHAPTER 6 MODELS OF PROCESS VARIATIONS IN DEVICE  
AND INTERCONNECT 98***Duane Boning and Sani Nassif*

6.1 Introduction: Sources of Variation	98
6.2 Overview: Statistical Descriptions	99
6.2.1 Lumped Statistics	99
6.2.2 Separation of Inter-Die and Intra-Die Variation	100
6.2.3 Inter-Die Variation	100
6.2.4 Intra-Die Variation	101

6.3 Survey of Process Variations	102
6.3.1 Device Geometry Variations	102
6.3.2 Device Material Parameter Variations	102
6.3.3 Device Electrical Parameter Variations	103
6.3.4 Interconnect Geometry Variations	103
6.3.5 Interconnect Material Parameter Variations	104
6.4 Methods to Characterize and Address Variation	105
6.4.1 Statistical Device Models	105
6.4.2 Sensitivity Analysis	106
6.4.3 Worst-Case Analysis	106
6.4.4 Spatial Variation Modeling and Mismatch	107
6.5 Application of Methods to Interconnect Impact Analysis	108
6.5.1 Example: Random and Wafer Level Variation Impact	109
6.5.2 Example: Interconnect Sensitivity Analysis	111
6.5.3 Example: Statistical Interconnect Impact	111
6.5.4 Example: Deterministic Interconnect Variation Impact	112
6.6 Conclusion	114
References	114

## PART III CIRCUIT STYLES FOR LOGIC 117

### CHAPTER 7 BASIC LOGIC FAMILIES 119

*Kerry Bernstein*

7.1 Introduction	119
7.2 Nonclocked Logic	119
7.2.1 Static Combinatorial CMOS Logic	120
7.2.2 DCVS	124
7.2.3 Pass-Gate Logic	125
7.3 Clocked Logic	128
7.3.1 Domino Logic—Single and Differential	128
7.3.2 Latched Evaluate Logic	130
7.4 Self-Timed and Asynchronous Logic	131
7.4.1 Self-Resetting CMOS	131
7.4.2 Clock-Delayed Domino	132
7.5 Implementation Issues	133
7.5.1 Circuit Style Selection Criteria	133
7.5.2 Design Metrics	137
7.6 Conclusion	138
Acknowledgments	138
References	138

### CHAPTER 8 ISSUES IN DYNAMIC LOGIC DESIGN 140

*Paul Gronowski*

8.1 Introduction to Dynamic Logic	140
8.1.1 Basics of Dynamic Logic	140
8.1.2 Examples of Dynamic Logic	142

8.1.3 Comparison of Dynamic Logic to Standard Complementary CMOS Logic	145
<b>8.2 Design Issues with Dynamic Logic</b>	<b>146</b>
8.2.1 Charge Leakage	146
8.2.2 Charge Sharing	148
8.2.3 Capacitive Coupling Issues	150
8.2.4 Minority Carrier Charge Injection	153
8.2.5 Supply Noise and Variation	155
<b>8.3 Conclusion</b>	<b>156</b>
References	157

**CHAPTER 9 SELF-TIMED PIPELINES 158***Ted Williams*

9.1 Introduction	158
9.2 Individual Stages	160
9.3 Definitions	163
9.4 Self-Timed Control Interconnections	164
9.5 Overall Pipeline Latency and Throughput	168
9.5.1 Ring Performance Graphs	169
9.5.2 Performance Region Edges	171
9.6 Applications	173
9.7 Margin, Testing, and Power Issues	176
9.8 Conclusion	178
References	179

**CHAPTER 10 HIGH-SPEED VLSI ARITHMETIC UNITS: ADDERS AND MULTIPLIERS 181***Vojin G. Oklobdzija*

10.1 Introduction	181
10.2 High-Speed Addition: Algorithms and VLSI Implementation	181
10.2.1 Full Adder	181
10.2.2 Ripple Carry Adder	183
10.2.3 Carry Skip Adder	183
10.2.4 Variable Block Adder	184
10.2.5 Carry Lookahead Adder	185
10.2.6 Recurrence Solver-Based Adders	190
10.2.7 Ling Adder	191
10.2.8 Conditional-Sum Addition	192
10.2.9 Carry Select Adder	192
10.2.10 DEC "Alpha" 21064 Adder	193
10.3 Multiplication	193
10.3.1 Algorithm	193
10.3.2 High-Performance Multipliers	195
10.4 Conclusion	202
References	203

**PART IV CLOCKING 205****CHAPTER 11 CLOCKED STORAGE ELEMENTS 207***Hamid Partovi*

11.1	On Clocking Strategy	207
11.2	The Nonideal Nature of Clock Signals	209
11.2.1	Jitter	209
11.2.2	Skew	210
11.3	The Basic Latch-Pair	210
11.4	The Basic Flip-Flop	211
11.5	Rules for Robust Design--1	213
11.6	Timing Properties of Sequential Logic	215
11.6.1	Edge-Triggered Clocking	216
11.6.2	Level-Sensitive Clocking	217
11.6.3	Slack-Passing and Time-Borrowing	219
11.6.4	Two-Wire Non-Overlapping Clocks	220
11.7	Comparing Latch-Pairs and Flip-Flops	220
11.8	High-Performance Clocked Storage Elements	221
11.8.1	The Modified Svensson Latch	222
11.8.2	The Transmission-Gate Level-Sensitive Latch	223
11.8.3	The Amplifier-Based Flip-Flop	223
11.8.4	The Latch and Flip-Flop Hybrid Element	225
11.9	Rules for Robust Design--2	228
11.10	Performance Metrics for Clocked Storage Elements	229
11.11	Latching Elements for Dynamic Circuits	231
11.12	Recommendations and Conclusion	233
	References	233

**CHAPTER 12 DESIGN OF HIGH-SPEED CMOS PLLs AND DLLs 235***John George Maneatis*

12.1	Introduction	235
12.2	PLL Architectures	236
12.2.1	Loop Components	238
12.3	Delay-Locked Loops	238
12.3.1	DLL Frequency Response	239
12.3.2	DLL Design Strategy	240
12.3.3	Alternative DLL Structures	240
12.4	Phase-Locked Loops	241
12.4.1	PLL Frequency Response	242
12.4.2	PLL with Higher-Order Roll-Off	244
12.4.3	PLL Design Issues	247
12.4.4	PLL Design Strategy	248
12.5	Advanced PLL Architectures	249
12.6	DLL/PLL Performance Issues	250

12.6.1 Output Jitter	250
12.6.2 Causes of Jitter	251
12.6.3 Supply/Substrate Noise Response	252
12.6.4 Observations on Jitter	252
12.6.5 Minimizing Supply Noise Sensitivity	253
12.6.6 Supply Noise Filters	254
12.6.7 Minimizing Substrate Noise Sensitivity	254
12.7 DLL/PLL Circuits	255
12.7.1 VCDLs and VCOs	255
12.7.2 Phase Detectors	257
12.7.3 Charge Pumps	257
12.7.4 Loop Filters	258
12.7.5 Circuit Summary	259
12.8 Self-Biased Techniques	259
12.9 Conclusion	260
References	260

**CHAPTER 13 CLOCK DISTRIBUTION 261***Daniel W. Bailey*

13.1 Introduction	261
13.1.1 Definitions	262
13.2 Objectives	265
13.2.1 Ideal Clock Objectives	265
13.2.2 Evaluation of Objectives	266
13.3 Implementation	268
13.3.1 Final Stage Drivers	268
13.3.2 Predriver Network	269
13.3.3 Examples	271
13.3.4 Trends	273
13.4 Clock Driver Layout	273
13.4.1 Electromigration	275
13.4.2 Productivity	275
13.4.3 Yield	276
13.5 Variation	276
13.5.1 Process	276
13.5.2 Power Supply	277
13.5.3 Temperature	278
13.5.4 Data-Dependent Noise	279
13.6 Conclusion	279
References	280

**PART V MEMORY SYSTEM DESIGN 283****CHAPTER 14 REGISTER FILES AND CACHES 285***Ronald Preston*

14.1 Basic Architecture	286
14.1.1 General Purpose Registers	286

14.1.2 Single-Cycle Caches	286
14.1.3 Multicycle Cache Arrays	289
14.2 Basic SRAM Cell Design and Operation	290
14.2.1 Read Operation	291
14.2.2 Write Operation	294
14.2.3 Variants on the Basic 6T Cells	297
14.3 Address Path Design	299
14.3.1 Preliminary Decoders	299
14.3.2 Final Row Drivers	299
14.3.3 Word Line Hierarchies and Layout	300
14.4 Read Path Design	301
14.4.1 Twisted Bit Line Architectures	301
14.4.2 Bit Line Precharging	301
14.4.3 Column Multiplexing	302
14.4.4 Sense Amplifier Design	302
14.4.5 Hierarchical Read Paths	304
14.5 Write Path Design	305
14.5.1 Write Amplifier Design	305
14.5.2 Column Multiplexing	305
14.6 Redundancy	306
14.6.1 Implementing Redundancy	307
14.7 Reliability Issues	307
14.7.1 Alpha Particles and Cosmic Rays	307
References	308

## CHAPTER 15 EMBEDDED DRAM 309

*Tadaaki Yamauchi and Michihiro Yamada*

15.1 Introduction	309
15.2 DRAM Basis	309
15.2.1 DRAM Architecture	309
15.2.2 Memory Cell	310
15.2.3 Basic DRAM Core Operation	312
15.2.4 Memory Cell Array Architecture of Embedded DRAM	314
15.3 Voltage Generator	315
15.3.1 Basic Configuration	315
15.3.2 Back Bias Generator	316
15.3.3 Bit Line Precharge and Cell Plate Voltage Generators	318
15.3.4 Voltage-Down Converter	318
15.4 Embedded DRAM	319
15.4.1 Features	319
15.4.2 Recent Embedded DRAM LSIs	320
15.4.3 Process Issues of Embedded DRAM	321
15.4.4 Design Issues of Embedded DRAM	323
15.4.5 The Future of Embedded DRAM	326
References	327

**PART VI INTERCONNECT AND I/O 329****CHAPTER 16 ANALYZING ON-CHIP INTERCONNECT EFFECTS 331***Noel Menezes and Lawrence Pileggi*

16.1 Introduction	331
16.1.1 Process Scaling, Interconnect Scaling, and Noise	331
16.1.2 Local and Global Interconnect	332
16.1.3 Interconnect Modeling	332
16.1.4 The Duality between Noise and Delay	334
16.2 Simplified Interconnect Analysis	335
16.2.1 The Elmore Delay and Its Properties	335
16.2.2 A Noise Bound for Coupled Interconnect	337
16.3 Model Order Reduction	338
16.3.1 An Example	339
16.3.2 Calculating Moments	340
16.3.3 Moment Matching—AWE	342
16.3.4 Applying AWE to Delay and Noise Analysis	344
16.3.5 Other Model Order Reduction Techniques	345
16.4 Driver Models	346
16.4.1 Resistance Shielding	346
16.4.2 The Effective Capacitance Drive Model for RC Loads	347
16.4.3 N-Port Driver Models	349
16.5 Conclusion	350
References	350

**CHAPTER 17 TECHNIQUES FOR DRIVING INTERCONNECT 352***Shannon V. Morton*

17.1 Introduction	352
17.2 Technology Scaling Trends	352
17.2.1 Relative Spatial Dimensions	352
17.2.2 Faster Edge Rates	356
17.2.3 Longer Electrical Lengths On-Chip	357
17.2.4 Process Variation	358
17.2.5 Architectural Issues	358
17.2.6 Power Dissipation	359
17.3 Problems and Solutions Regarding Capacitance	359
17.3.1 Power Dissipation	360
17.3.2 Delay Variations	360
17.3.3 Crosstalk Noise	361
17.3.4 Potential Solutions	361
17.4 Problems and Solutions Regarding Inductance	364
17.4.1 Delay Variations	367
17.4.2 Crosstalk Noise	368
17.4.3 Potential Solutions	369
17.5 Problems and Solutions Regarding Resistance	370
17.6 Problems and Solutions Regarding Long Distance Routing	371
17.6.1 Repeater Insertion	371

17.6.2 Buffer Insertion	373
17.6.3 Low Swing Signaling	373
17.6.4 Active Regenerators	374
17.6.5 Current Sensing	374
17.6.6 Wave Pipelining	374
17.7 Conclusion	375
Acknowledgments	375
References	375
<b>CHAPTER 18 I/O AND ESD CIRCUIT DESIGN</b>	<b>377</b>
<i>Stephen C. Thierauf and Warren R. Anderson</i>	
18.1 Introduction	377
18.2 Power Supply Considerations	378
18.2.1 Split Power Supply Systems	378
18.2.2 Power Supply Clamps	379
18.3 Off-Chip-Driver Edge Rate Control	380
18.4 Mixed-Voltage I/O	381
18.4.1 Floating Well Driver	381
18.4.2 Open Drain Signaling	382
18.4.3 Cascoding	383
18.4.4 Level Shifting	384
18.5 Impedance Matching	384
18.6 Precompensation Drivers	385
18.7 Input Receivers	385
18.8 The ESD Threat	386
18.9 ESD Models	387
18.10 Circuit Topology of the ESD Protection Network	388
18.10.1 The Qualities of Good ESD Protection	388
18.11 ESD Protection Design Elements and Methods	389
18.11.1 Nonsilicided and Silicide-Blocked NMOS	389
18.11.2 Silicided NMOS	390
18.11.3 Double Diode ESD Protection	392
18.12 Power Supply Clamps	393
18.13 CDM Considerations	394
References	395
<b>CHAPTER 19 HIGH-SPEED ELECTRICAL SIGNALING</b>	<b>397</b>
<i>Stefanos Sidiropoulos, Chih-Kong Ken Yang, and Mark Horowitz</i>	
19.1 Transmission Lines	398
19.1.1 Image Currents	398
19.1.2 Reflections	399
19.1.3 Attenuation	400
19.1.4 Methods of Signaling	401
19.2 Link Performance Metrics	402
19.2.1 FO-4 Delay Metric	402
19.2.2 Link Margins	403

19.3 Transmitters	405
19.3.1 Output Drivers	406
19.3.2 Impedance, Current, and Slew-Rate Control	407
19.3.3 Maximizing Bit Rate	409
19.4 Receivers	410
19.4.1 Input Noise	411
19.4.2 Receiver Design	412
19.4.3 Demultiplexing Issues	413
19.5 Clock Generation	414
19.5.1 PLLs and Phase Noise	414
19.5.2 Dual-Loop Architectures	416
19.5.3 Timing Circuit Scaling	417
19.6 Future Trends	418
19.6.1 Equalization	419
19.6.2 Multilevel Signaling	421
19.6.3 DAC and ADCs	422
19.7 Conclusion	422
Acknowledgments	423
References	423

## PART VII RELIABILITY 427

### CHAPTER 20 ELECTROMIGRATION RELIABILITY 429

*J. Joseph Clement*

20.1 Introduction	429
20.2 Materials and Process Effects on Electromigration	430
20.2.1 Interconnect Technology Evolution	430
20.2.2 Effect of Interlevel Dielectrics	431
20.2.3 Microstructure, Line Width, and Line Length	432
20.3 Electromigration Lifetime	434
20.3.1 Electromigration Transport	434
20.3.2 Accelerated Lifetime Characterization	435
20.3.3 Modeling Electromigration	436
20.3.4 Pulsed dc and ac Operation	437
20.4 Designing for Electromigration Reliability	439
20.4.1 Reliability Budgeting	439
20.4.2 CAD Tools	439
20.4.3 CAD Reliability Analysis	442
20.4.4 rms Current Limits	445
20.5 Conclusion	445
Acknowledgments	446
References	446

### CHAPTER 21 HOT CARRIER RELIABILITY 449

*Kaizad Mistry*

21.1 What Are Hot Carriers?	449
21.2 How Do Hot Carriers Degrade MOSFETs?	452

21.3 Modeling the Degradation	454
21.4 Circuit Effects	457
21.5 Ensuring Circuit Reliability	460
21.6 Example Analysis	461
21.7 Transistor Scaling Trends	463
References	464

**PART VIII CAD TOOLS AND TEST 467****CHAPTER 22 OVERVIEW OF COMPUTER-AIDED DESIGN TOOLS 469***Yao-Tsung Yen*

22.1 Introduction	469
22.2 Micro-Architecture Design and Circuit Feasibility Study Tools	470
22.3 RTL Model Design Tools	471
22.4 Datapath/Memory Design Tools	473
22.4.1 Datapath/Memory Schematic Design Tools	473
22.4.2 Datapath/Memory Layout Design Tools	475
22.5 Control Logic Design Tools	475
22.6 Chip Assembly and Global Net Route	476
22.7 Chip-Level Layout, Circuit, and Timing Verification	476
22.7.1 Layout Verification	476
22.7.2 Circuit and Timing Analysis	477
22.8 Test Patterns Generation	478
22.9 Conclusion	478
References	479

**CHAPTER 23 TIMING VERIFICATION 480***Victor Peng*

23.1 Introduction	480
23.2 Timing Verification Goals and Analysis	480
23.2.1 Speed (Set-Up Time) Analysis	480
23.2.2 Functional (Hold Time) Analysis	481
23.2.3 The Timing Analysis Process	482
23.2.4 Max Path Modeling	483
23.2.5 Min Path Modeling	483
23.3 Key Factors in High-Speed Design and Timing Verification	484
23.3.1 Product Goals	484
23.3.2 Clocking and Storage Element Strategy	484
23.3.3 Circuit Structures and Design Guidelines	489
23.4 Timing Verification of Non-memory Custom Blocks	490
23.4.1 Dynamic Logic	490
23.4.2 Pass-Gate Logic	491
23.4.3 Self-Timed Logic	492

23.5 Timing Verification of Memory Blocks	492
23.6 Design Flow and Full-Chip Timing Verification	494
23.7 Future Challenges	497
References	498

## CHAPTER 24 DESIGN AND ANALYSIS OF POWER DISTRIBUTION NETWORKS 499

*David Blaauw, Rajendran Panda, and Rajat Chaudhry*

24.1 Introduction	499
24.2 Power Distribution Design	501
24.2.1 Power Grid Integrity Problems	501
24.2.2 Power Distribution Design Styles	502
24.2.3 Design Methodology for Power Distribution Networks	505
24.2.4 Voltage Drop Mitigation	507
24.2.5 Power Grid Resonance	509
24.3 Power Distribution Analysis	510
24.3.1 Fast Linear Solvers for Power Grid Analysis	511
24.3.2 Simulation Vector Generation	512
24.4 Power Grid Models	514
24.4.1 Resistance Models for Power Networks	514
24.4.2 Capacitance Models for Power Networks	515
24.4.3 Inductance Models for Power Networks	517
24.4.4 Comparison between R, RC, and RLC Analysis	519
24.5 Conclusion	520
References	520

## CHAPTER 25 TESTING OF HIGH-PERFORMANCE PROCESSORS 523

*Dilip K. Bhavsar*

25.1 Introduction	523
25.2 Basic Concepts in Testing	524
25.2.1 Components of Testing	524
25.2.2 Fault Modeling and Testing Methods	525
25.2.3 Test Development	527
25.2.4 Testing in Production	529
25.3 Designing for Testability	530
25.3.1 Scan Design Techniques	530
25.3.2 Built-in Self-Test (BiST) Techniques	534
25.3.3 IEEE 1149.1 Standard Test Access Port and Boundary Scan Register	540
25.4 Conclusion	542
References	543

## INDEX 545

# PREFACE

The performance of microprocessors has increased exponentially since their introduction in the early 1970s. This growth has been made possible by dramatic technical advances in semiconductor manufacturing, circuit design, computer architecture, and CAD tools. Although many technical problems have been resolved along the way, many new challenges emerge as transistors are counted in billions, and clock frequencies in gigahertz.

This book discusses the design of the next generation of microprocessors in deep submicron CMOS technologies. The topics in the book were chosen to cover the essence of high-performance design. The chapters were written by some of the world's leading technologists, designers, and researchers who are pushing the state of the art as they grapple daily with the technical hurdles that must be overcome before the microprocessors of tomorrow are shipped. All levels of system abstraction are covered, but the emphasis rests squarely on circuit design. Examples are drawn from processors designed at AMD, Digital/Compaq, IBM, Intel, MIPS, Mitsubishi, Motorola and Toshiba.

The book was written for students of VLSI design as well as practicing circuit designers, architects, system designers, CAD tool developers, process technologists, and researchers. It assumes a basic knowledge of digital circuit design and device operation, and covers a broad range of circuit styles and VLSI design techniques. Each chapter stands alone so the reader can pick and choose topics of interest and read them in any order.

The 25 chapters are grouped into eight sections:

- Section I discusses the impact of physical technology on architectural choices.
- Section II deals with technology and covers CMOS scaling, leakage current reduction, low-voltage devices, silicon on insulator, and models of process variations.
- Section III explores contemporary circuit design styles including a survey of logic families, robust dynamic circuit design, asynchronous logic, self-timed pipelines, and high-speed arithmetic units.
- Section IV deals with clocks, including edge triggered flip-flops, level-sensitive and pulse latches; phase-locked and delay-locked loops; and on-chip clock distribution schemes.
- Section V considers memory design, including register files and caches as well as embedded DRAM—structures that occupy a large portion of real estate on today's microprocessors. The limited speed of signal transmission both on and off chip has emerged as a serious performance bottleneck.
- Section VI covers techniques for driving on-chip interconnect, I/O and pad design, ESD, and high-speed signaling in general.
- Section VII deals with the realities of wear out mechanisms, such as electromigration and hot carriers, the necessity of considering these phenomena at the design stage, and their overall impact on device reliability.
- Section VIII provides a broad overview of the CAD tools needed for high-

performance microprocessor design, including timing verification, and the analysis of power distribution schemes. The book concludes with a chapter on testing.

We would like to acknowledge the many people who contributed to the completion of this book. First we would like to thank the authors of the chapters. The preparation of the text often required the authors to work long nights and weekends, while still meeting their demanding commitments at work.

Each chapter went through a rigorous review process. We would like to thank many expert reviewers for their insightful and constructive feedback: Vishwani Agrawal, Bell Labs; Dimitri Antoniadis, MIT; Lawrence Bair, Compaq Computer Corp.; Paul Gronowski, Compaq Computer Corp.; Jim Lloyd, Lloyd Technology Associates; Liam Madden, MIPS Technologies, Inc.; Siva G. Narendra, Intel Corp.; Steven M. Nowick, Columbia University; Hamid Partovi, Advanced Micro Devices, Inc.; John Poulton, Chip2Chip, Inc.; Ronald Preston, Compaq Computer Corp.; Kewal K. Saluja, University of Wisconsin-Madison; Stefanos Sidiropoulos, Rambus Inc.; Don Stark, Rambus Inc.; Duke Xanthopoulos, MIT; and Y. T. Yen, YX Technologies, Inc.; as well as other reviewers who wish to remain anonymous.

During the book's preparation we received exceptional support and guidance from the IEEE publishing organization. We would like to particularly thank Rob Bedford, Marilyn G. Catis, John Griffin, and Anthony VenGraitis from the IEEE Press; Maureen Allen from Keyword Publishing Services, and Suzanne Ingrao, from Ingrao Associates.

Finally, we would like to acknowledge the contributions of countless design engineers and technologists whose ingenuity has fueled the microprocessor revolution that is transforming our lives and economies. To quote Isaac Newton, we are ". . . standing upon the shoulders of Giants."

Anantha Chandrakasan  
*Massachusetts Institute of Technology*  
*Cambridge, MA*

William J. Bowhill  
*Compaq Computer Corporation*  
*Shrewsbury, MA*

Frank Fox  
*Rambus Inc.*  
*Mountain View, CA*

August 2000

PART



# OVERVIEW

# IMPACT OF PHYSICAL TECHNOLOGY ON ARCHITECTURE

John H. Edmondson  
*Compaq Computer Corporation*

The fundamental problem in the design of microprocessors is that of mapping an abstract computer architecture onto the physical technology in which the microprocessor will be manufactured. Computer architectures are composed of logic and storage elements, which are abstractions. When realized in a particular technology, logic values, storage states, and the connections between elements map to logic circuits, state circuits, and signals on metal interconnections in the physical technology. Inevitably, there are limitations resulting from the properties of the physical technology that determine the range of realizable architectures. In addition, there are opportunities presented when a particular architectural component maps well into the physical technology. Given the architect's goals of finding a realizable and reasonably optimal architecture that meets customer requirements, it is critically important to understand the limitations and opportunities arising from the physical technology.

## 1.1 INTRODUCTION

The architecture goal is to find a design that implements the required function with the best possible trade-off between various measures of success. Metrics that are applicable to almost any design are performance, power consumption, die area and other costs (e.g., package cost), and reliability. Generally, limits to functional capabilities are necessary to meet product requirements for one or more of these metrics. The choices embodied in the architecture determine the trade-offs between these various metrics and product goals.

Often, a distinction is made between *architecture* and *micro-architecture*. *Architecture* is used to describe an abstract requirement specification and *micro-architecture* is used to describe the specific composition of elements that realizes an *architecture* in a specific design. In the broader sense, both are architecture, and in this chapter the distinction is not made because most discussions apply at both levels to some extent.

### 1.1.1 Suitability of CMOS Technology

CMOS integrated circuit technology is far from ideal in realizing the architectural abstractions. The basic gate has limited fan-in. Gate delay is strongly dependent on fan-in, fan-out, and wire length. Resistance and inductance in wires creates significant

problems for the designer in submicron technology. Efficient realization of storage elements requires special circuit approaches that require careful design and additional electrical verification to assure correct operation. Advanced logic circuit styles can be used, with similar design and verification requirements. Finally, advanced logic circuits and storage circuits do not support all similar architectural concepts equally well. Instead, they favor one configuration over another.

Despite the problems just mentioned, CMOS technology has certain properties that have made it the key driver of progress in computer design for many years. The most important property is scaling and manufacturability. Manufacturers have been able to build CMOS integrated circuits at low cost and with a steady reduction in feature size causing rapid improvements in circuit speed, cost/function, and functions/chip. Other important properties are highly efficient implementation of RAMs, low power consumption relative to competing technologies, and that automated design methodologies are highly successful in CMOS. Advancement in CMOS technology will remain a driving force in high-performance microprocessors as well as in other computing technologies such as embedded computing and DSP for many years to come.

### 1.1.2 Physical Technology Impact at Various Scales

Physical technology's impact on architecture occurs at several scales. At the chip scale, the growth in the number and size of architectural functions that could be realized on a single chip has been a fundamental factor in microprocessor advancement. At a more macroscopic scale, physical arrangement of elements and careful attention to interconnection density and length can be critical. The floorplan of a chip has direct consequences visible at the architecture level. For example, the latency of accesses to a primary data cache is very dependent on how successfully the floorplan manages to optimize the routing between the load-store execution unit, the cache, and the integer execution unit.

At the scale of a section or function unit, mapping functional components into regular structures is important. Examples are mapping a group of computing and storage elements into a datapath, mapping an architectural function to a VLSI array, fitting the architectural function to an advanced logic style such as domino logic, or implementing part or all of a function as a RAM.

At the scale of a circuit, there can be significant architectural impact. For example, arithmetic circuits generally improve strongly from special circuits and careful circuit design in general. For another example, the reliability of a static RAM cell is a critical factor in determining the need for error-tolerant architectural features such as parity checking or error correcting codes.

### 1.1.3 RISC and the Importance of Well-Chosen Architecture

One important development in architecture has been the focus on architecture's role in efficiently utilizing the underlying physical technology. The RISC breakthrough is a key example of this idea. Among other things, early RISC architects recognized that by scaling the complexity and cost of microprocessor architecture down to the point at which an effective pipeline could be implemented on a single IC, a large increase in performance was possible. More generally, they recognized that a good match between

an architecture and the implementation technology is crucial to achieving maximum performance and cost efficiency.

Many of the RISC concepts would arguably improve performance and reduce cost in any technology. But as a practical matter, RISC ideas and practices have evolved with CMOS technology. Today's RISC computers are typically complicated and feature-laden relative to the original RISC implementations, but they still reflect careful trade-offs by designers between architectural complexity, implementation cost, and clock speed—trade-offs that are made with the target technology very much in mind.

#### 1.1.4 Key High-Level Decisions and Trade-Offs

Once the instruction set architecture is chosen, and regardless of whether it is a RISC architecture, there are many high-level decisions made by the architect that are profoundly influenced by the technology. Here are a few examples. What percentage of chip area should be used for cache? Which is better: a larger, slower single-level on-chip cache or a two-level hierarchy with smaller, faster first-level cache? How many functional units and register file ports are affordable? How complex a memory system is feasible?

In high-performance microprocessor design, the chip size constraint is determined either by the maximum size the manufacturing process can support or by the maximum cost allowable for the product. In either case, the chip size is often very large in comparison to other IC designs.

Determining the size of architectural components is not easy. The most obvious approach is to compare the proposed element to a similar element in an existing design. Another approach is practical if the proposed element is to be implemented using a regular array or datapath. In this case, the size can be directly estimated by analysis of the wiring and the circuitry within the basic cells. If these approaches aren't applicable, the remaining approach is to use metrics based on gates, wiring, and other factors; but these tend to be less accurate.

The cycle time of the processor is probably the single most important decision of all, and also one of the most difficult. Section 1.3 explores this decision in detail.

#### 1.1.5 Other Technology Issues

Other technology issues that are important to the microprocessor designer are the trends in physical properties of interconnections, power consumption, and packaging. In the past, very long interconnect routes were not as difficult a problem during top-level architectural decision making, but the trend in technology is for interconnection length limitations to become a first-order constraint. Power dissipation is an aspect of physical technology that has become increasingly important to the architect as technology is scaled. Advances in packaging are allowing more and faster signals, which taken together with higher levels of integration of the computer system elements are enabling significant advances in scalable multiprocessing and supercomputing. These issues are all discussed in later sections of this chapter.

A direct consequence of physical technology is the need for constantly improving design methodology. Design methodology is the combination of the CAD tools, design strategies, and verification strategies that together ensure a successful design. While design methodology is mainly driven by physical technology and is always relevant

when discussing limitations imposed by technology, this chapter does not explore methodology in any direct way.

## 1.2 IMPLEMENTING PROCESSOR ARCHITECTURE IN CMOS TECHNOLOGY

The computer architect works in the abstract domain of binary logic levels, logic gates, and storage elements. An arbitrarily complex logic function can be built from a logic family containing only a few basic types of logic gates (NAND gates alone, for example). But logical correctness does not ensure feasibility, much less minimum cycle time or minimum cost. Similarly, basic CMOS latches and logic gates can be combined to build storage arrays, but efficient implementation requires special circuits and design techniques. This section explores the problem of efficiently mapping an architecture to CMOS, while optimizing for speed, power, die area, and other metrics. It is crucial to understand the structures that work well in CMOS and to plan the architecture so that it maps on to those structures wherever possible.

Here is a list of some of the more useful mappings into CMOS:

- Reduce critical logic functions to wide fan-in NOR, or AND-NOR with wide fan-in NOR, and use dynamic logic to efficiently realize the function.
- Use domino logic for high speed and efficient use of area.
- Use other advanced logic design styles into which certain logic functions map particularly well.
- Organize a function into a 2-dimensional array or datapath where the regular structure provides advantages.
- Use RAM arrays where appropriate.

It is important to recognize that as physical technology continues to scale and evolve, the useful mapping techniques will evolve as well. Additionally, new logic design techniques will likely be invented. Highly successful architecture will always be sensitive to the limitations and peculiar opportunities of the supporting technology, but the more technology-dependent suggestions given in this section will inevitably become outdated.

### 1.2.1 Dynamic Logic

Dynamic logic (including domino logic) can provide significant speed and area improvements in high-performance design. This section briefly introduces dynamic logic and gives examples of dynamic logic usage in microprocessors. Dynamic logic can greatly improve the speed of critical paths, because dynamic gates are considerably faster than static equivalents. However, use of dynamic logic introduces new timing constraints and restricts the generality of logic functions that can be practically realized.

Note that there are many important design issues in the implementation of dynamic logic circuits, both to ensure correct operation and to achieve the highest possible speed, but these issues are mainly beyond the scope of this discussion. Issues that directly influence architecture are mentioned here, while many issues that affect architecture much less are ignored.

### 1.2.1.1 Zero Detection Circuit Example

A zero-detect function of a 16-bit data bus is a simple example of an optimization opportunity at the scale of a function unit. It is possible to implement this function in standard CMOS gates, building the 16-wide fan-in OR function using a tree structure composed of NAND and NOR gates. This implementation would be significantly slower than one using dynamic logic to implement the 16-wide fan-in NOR gate in one dynamic gate, as shown in Fig. 1.1. In addition, it would take more area and it would be more difficult to route the interconnect than in the dynamic version.

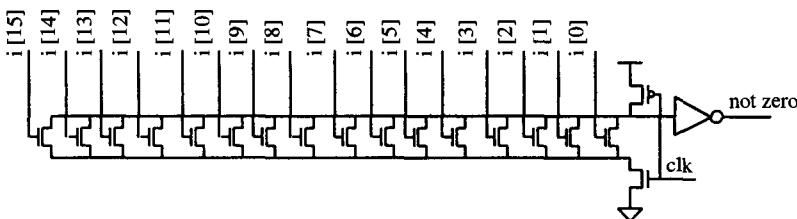


Figure 1.1 Sixteen-bit zero detector in dynamic CMOS.

However, the dynamic version places a new constraint on the design. The circuit is clocked, so the input data must be set up prior to the clock signal rising, and it must remain stable and valid throughout the clock-high period. Both the speed advantage and the timing constraint may be visible at the architecture level.

### 1.2.1.2 Comparator Example

Figure 1.2 shows an important extension of the wide fan-in NOR strategy, the AND-NOR with wide fan-in on the NOR part. The circuit is an 8-bit comparitor. The dynamic node will fall and the output will rise during the evaluate phase if any of the A input bits are not equal to the corresponding bit of the B input. The key point here is that an AND function is realized in each NMOS pull-down stack. This AND-NOR structure appears in many interesting dynamic control circuits.

In Fig. 1.2, the bottom NMOS transistors in each stack are forced off during the precharge phase. This eliminates the need for a clocked pull-down device similar to the one in Fig. 1.1, reducing stack height. With wide fan-in, the circuit slows significantly

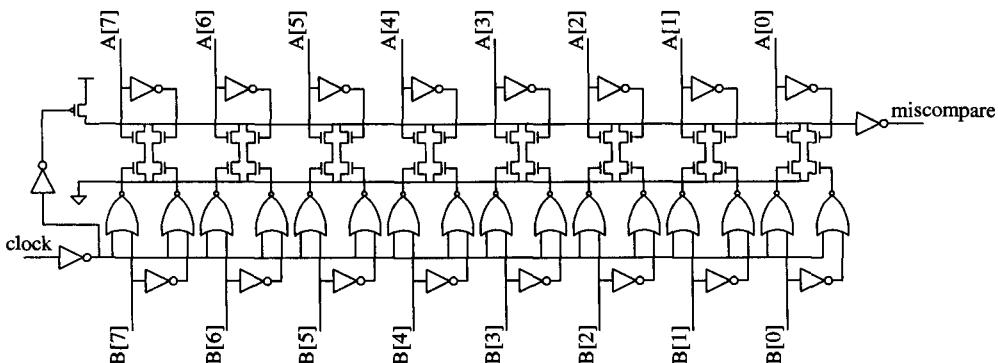


Figure 1.2 Dynamic comparator example.

as dynamic stack heights grow, and problematic electrical issues are worsened, so it is best to minimize stack heights, though the parameters may vary as the CMOS technology is changed. (Stack height is the number of devices connected in series between the dynamic output node and the power supply node. For practical purposes the stack height is quite limited. This translates to a limitation on logic functions in dynamic gates.)

### 1.2.1.3 Domino Logic

Domino logic is an extension of dynamic logic in which the output of one dynamic gate (the output of the inverter connected to the dynamic node) is connected directly to the next dynamic gate. Figure 1.3 shows the basic NMOS domino structure. The NMOS pull-down network in each gate can implement a variety of logic functions, though there are limits to design flexibility due to speed and electrical considerations.

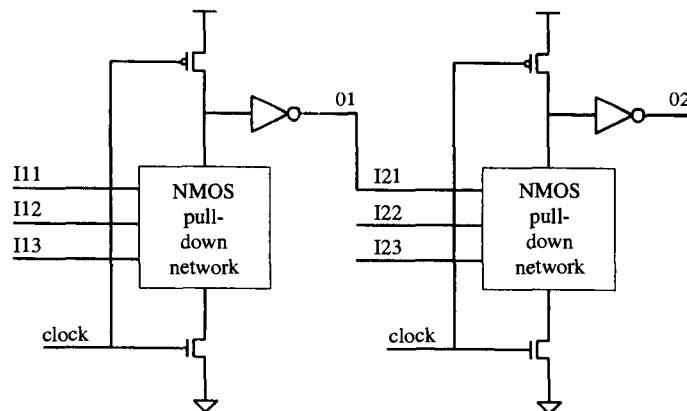


Figure 1.3 Domino logic.

As the name implies, the stages of domino logic evaluate one after another once the clock rises. Since the NMOS pull-down network can implement a wide fan-in NOR or AND-NOR function, complex logic functions can be realized with multiple stages of domino logic. However, some logic functions cannot be mapped into successive stages of domino logic. The key constraints that determine which logic functions are realizable are:

1. Wide fan-in must map to a NOR function, and
2. Each stage is noninverting.

For example, if the AND of 32 bits is required at the output of the first stage of a logic function, this can only be realized by inverting all the inputs, and connecting them to a 32-bit dynamic NOR gate. By DeMorgan's theorem the result at the output of the inverter is the NAND over the 32 bits of data. But, if the second stage of this logic function must OR the results of the 32 logical ANDs (where each is an AND of 32 bits), domino logic cannot be used in the second stage. This is because an extra level of inversion is required to make the logic function correct, but domino logic works only if each domino stage is noninverting. The first stage in this example could not be done as a dynamic NAND because it is not practical to implement a 32-high NAND stack.

(In fact, stack heights beyond three or four devices connected in series tend to be impractical.) These two constraints taken together limit the case in which domino logic can be used. For the architect, the challenge is to find functions that will produce the desired architectural behavior and will map successfully onto domino logic.

#### **1.2.1.4 PLAs and Self-Timed Dynamic Logic**

The preceding example of AND into OR is essentially the classic PLA structure. Because it can't be made to domino, two clock edges are required, one to enable evaluation in the AND plane and the other to evaluate the OR plane. In a single clock environment, the two planes could be evaluated in the two phases of one cycle but this is too slow to be useful in many cases. Designers use self-timed logic to solve this problem. A timer is built to trigger the OR plane evaluation after the AND plane is guaranteed to have completed its evaluation. Typically, this is accomplished by implementing one dummy AND term that is slightly more loaded and slightly less strongly driven than the worst real AND term. Such logic is difficult to design and there must always be some margin included such that OR plane evaluation is delayed more than necessary in the typical case. In general, cases that don't domino could be managed through self-timed design, but it is difficult, failure prone, and never as fast as domino logic. In principle, self-timing can be applied in any case in which domino logic won't work, but generally it is not practical because of the difficulty of building a dependable timer.

### **1.2.2 Advanced Logic Styles**

Other logic styles also have their place. For example, the XOR function is extremely important in arithmetic logic and in error correcting logic. However, domino logic will not support XOR in an intermediate logic stage. DCVS logic [1] has been used for such circuits because of the ease of implementing a fast XOR function, particularly an XOR sum of multiple bits (e.g., the even or odd parity function). Dual-rail domino logic has also been used because a “finish” signal can be generated from a dual-rail domino gate that can be used to start evaluation of the next stage, solving the XOR implementation problem in a different way [1].

### **1.2.3 Dynamic Logic Examples from High-Performance Processor Designs**

Examples from two processor designs are shown below. Notice that dynamic and domino logic styles are used to implement wide fan-in logic that is fairly fast, that stack heights are small with the 2-input AND-NOR occurring frequently, and that considerable effort and creativity was used to force the architectural function to fit a fast circuit implementation.

#### **1.2.3.1 Alpha 21164 Scoreboard Example**

Figure 1.4 is taken from a paper describing the Alpha 21164 microprocessor [2]. The 21164 operates at a CPU frequency of 300 MHz in 0.5 CMOS technology. The figure depicts the 21164's instruction scoreboard.

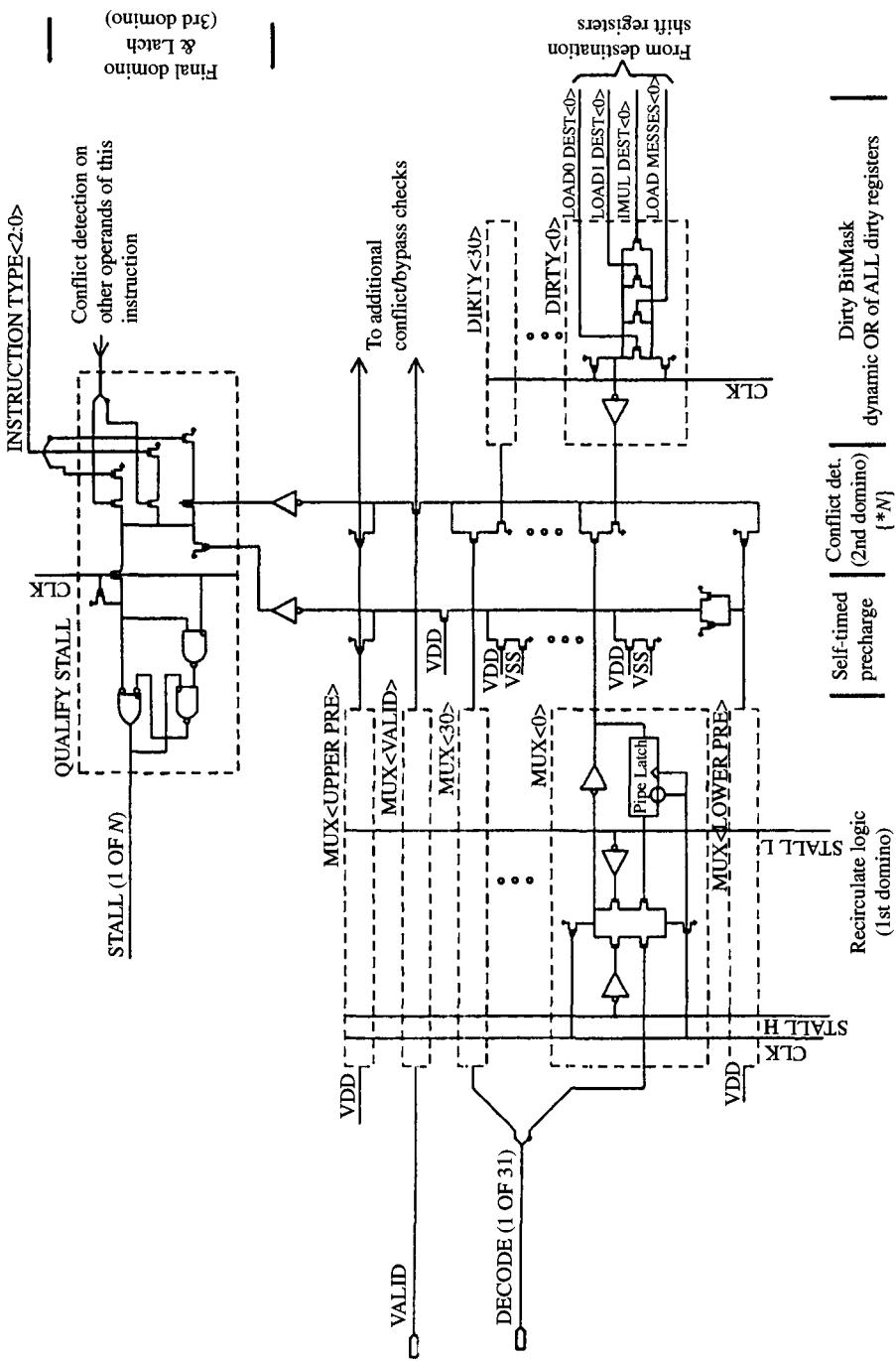


Figure 1.4 Alpha 21164 scoreboard.

The design works with fully decoded register indexes. That is, there is a 31-bit binary “one-hot” vector representing each source and destination register instead of a 5-bit binary encoding. In this fully decoded representation, equality comparison maps to a two-input AND function. Converting to decoded forms of indexes is a powerful method that can often be used to reduce stack heights dramatically and enable the use of dynamic logic. In decoded-form logic, the logic cells tend to be small and they can usually be laid out in an efficient array.

Details of the circuit’s operation can be found in the original journal article. There are three domino stages. The cross-coupled structure in the final stage is a glitch latch that catches the result of the dynamic evaluation at the same time as precharge begins to clear it away, a required technique for single-wire clocking.

### 1.2.3.2 Alpha 21264 Issue Logic Example

The 21264 example (Fig. 1.5) is a clever design that demonstrates noticeable trade-offs between architecture and circuits. The figure is taken from a paper describing the 21264 Alpha microprocessor’s integer instruction issue logic [3]. The figure shows logic that performs the issue arbitration function. The arbiter must choose from among the up to 20 queued instructions and pick up to two instructions for execution in an associated pair of execution units. If only one instruction is ready the arbiter still mustn’t pick the same instruction for both units. This is the “pick-2” function. The logic design has a number of interesting consequences at the architectural level.

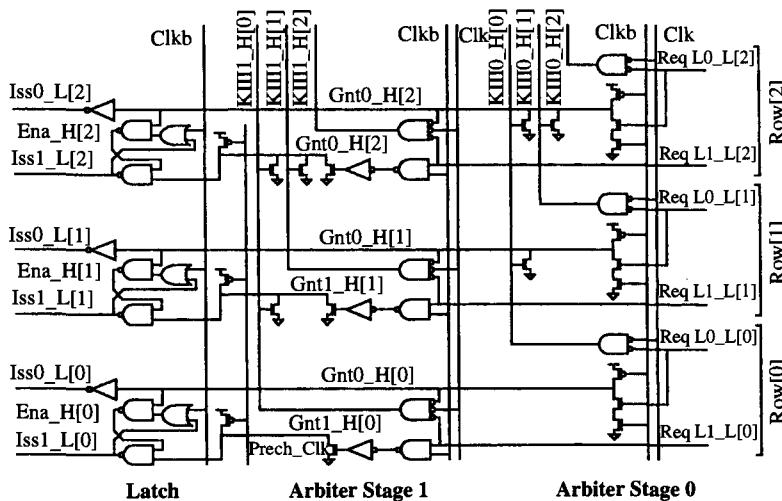


Figure 1.5 Alpha 21264 issue logic example.

Figure 1.5 shows a simplified version of the pick-2 arbiter. The simplified version handles requests from only three instructions, but the underlying concept extends easily to larger numbers of requesting instructions. Each arbiter stage is a dynamic priority encoder circuit that picks the highest priority single requester. The circuit works by allowing each requester to kill the grants to all lower priority instructions. If an instruction is not requesting, it kills its own grant and permits lower priority instructions to be

granted. The second stage gates the request for the second execution unit with failure to obtain grant in the first stage. This way, an instruction that is granted in the first stage will not be a requester in the second stage, and a lower priority requester can win in the second stage. Notice that this logic dominos such that the falling edge on Gnt0\_H signals in the first stage enable the second stage priority to be resolved. The set-reset flip-flop at the output of the arbiter ensures that at most one of the two issue signals is asserted for a given instruction.

The arbiter in Fig. 1.5 has a number of interesting architectural consequences. First, the 21264 has four execution units, but since the designers do not have a solution to the “pick-4” problem that would meet the cycle time requirement, they limit instructions to waiting for at most two execution units while in the queue. There is some degradation in architectural throughput when compared to a more ideal scheme. To determine the exact performance costs of such choices, architects simulate different options in a fast abstract model of the architecture.

Notice also that the arbiter is not symmetric. If the second highest priority instruction is requesting only the first execution unit, and the highest priority instruction is requesting both units, the lower priority instruction will not be granted at all. An ideal algorithm would place the second priority in the first execution unit and the first priority instruction in the second unit. Again, simulation would be used to determine the resulting performance cost.

### **1.2.3.3 Other Dynamic Logic Examples**

Dynamic logic, including domino logic has been used in many high-performance microprocessors for arithmetic circuits, instruction issue logic, cache access logic, and other logic [4]–[6]. The architectures represented in these example references are the Power-PC, Hewlett-Packard’s PA-RISC, and the MIPS architecture. Many other examples can be found in available research literature.

## **1.2.4 Datapaths**

The datapath is a well-known technique to improve design efficiency for data and address logic in a processor. By restricting each bit of a datapath to a certain bit column, and restricting function units to broad rows in the datapath structure, significant layout and routing efficiency is achieved in constructing a section. There may be inefficiency in individual components, but the payback is significant in the larger scale. As an added benefit, it can be quite a bit easier to estimate the size of a datapath during early design phases than it would be if the section were not laid out in such a regular way.

## **1.2.5 RAMs**

RAMs are the single most optimized circuit in CMOS process technology. Process engineers work on basic RAM cells from early in development of a technology. Circuits in RAMs (e.g., sense amps, column MUXes, address decoders) are usually the most carefully crafted and analyzed on any chip. RAMs can be static, dynamic, single-ported, multiported, and so on. Later chapters of this book go into extensive detail on SRAM and DRAM design. For the processor architect, RAMs are important building blocks used to realize dramatic performance gains.

Caches are usually fairly large, single-ported SRAM structures. Typically they are composed of multiple banks for higher speed and reduced power. Register files are

typically smaller RAMs that are optimized for a large number of ports. Individual ports are usually specialized to either writing or reading, though read-write ports are also sometimes used.

RAM cells are at the heart of many other structures, such as the part of the instruction queue that holds instruction information (information that is read out and passed to the register file and the execution units when an instruction is issued). These are often multiported structures.

Content addressable memories (CAMs) are used for translation buffers and other components. In general a CAM can be thought of as a RAM in which the address decoder has been replaced by a bank of comparitors coupled with RAM cells to hold the match value. In a typical design, comparitor cells are RAM cells with XNOR gates that compare the bits of the address to their stored value and conditionally pull down a dynamic logic node to indicate hit or miss. This circuit is similar to the circuit in Fig. 1.2.

The comparitor array in a CAM is usually expensive (in power and area) and somewhat slow, limiting practical CAM storage sizes. However, CAMs are critical architectural components, particularly in high-performance memory subsystems.

For the architect, the main point is that RAMs are far better than a collection of latches. From an architecture point of view, a RAM is a large decoder, a large number of storage elements, and a large multiplexor (or demultiplexor) for selecting one set of the storage elements to read (or write). But the implementation of a RAM is very efficient in area, speed, and power consumption, and the efficiency is most dramatic for single-ported RAMs. So the architect should map any large, regular collection of storage elements into the simplest RAM structure possible. Additionally, architectures that use a RAM-like structure to solve a problem or improve performance may be more efficient than those that use a different solution.

RAM area increases roughly as the square of the number of ports. This is because each port requires extra control lines (word lines) and extra data wires (bit lines). As the area increases, the circuits slow down, and power increases. So the architect should strongly prefer single-ported structures.

Larger RAMs are typically constructed from banks of small RAM arrays. Banks may be accessed simultaneously. This suggests an important strategy for building logical multiported storage arrays. Map a logically multiported storage structure to single-ported RAM banks by ensuring that the multiple simultaneous accesses do not access the same RAM banks. If the multiple accesses are not naturally separated into different banks of the storage array, then the architecture can enforce that access restriction by stalling or aborting all but one of any group of conflicting accesses. In large arrays, the resulting complexity is very likely to be worth the improved design efficiency.

Another efficiency strategy is to merge multiple related structures. For example, if the instruction cache and the branch predictor storage array are merged, area can be saved because the RAMs can share address decode logic. Where the array is organized into multiple banks, complexity can be reduced if the same banking rules apply to both accesses. If an instruction fetch doesn't conflict with another simultaneous fetch in the instruction cache, then the corresponding branch prediction accesses won't conflict either. This might not be the case if the two arrays were separate, with each banked in a different way.

Associativity in caches is another trade-off. Direct mapped caches are faster for two reasons. First, there is no MUX required after the RAM read. The MUX is present in associative caches to select the data associated with the tag that hits. In direct

mapped caches, there is only one data item to choose, so this MUX is eliminated. Secondly, in a direct mapped cache it is possible to begin operating on the data prior to determining if the tag hits. This is because the pipeline control logic of the processor can be used to abort the calculation in the event of a cache miss prior to committing the result to permanent architectural state storage. With associative caches, the correct data is not known until the hit calculation completes. Finally, a direct mapped cache is just simpler and smaller. There is less logic. However, if associativity is an architectural requirement, it is possible to save power by accessing just the way or set in which a cache hit occurred, at the cost of added latency.

There are many possibilities in RAM design. Difficult architectural problems can be solved with a variation on a standard RAM design. It is difficult for the architect to accurately determine what might be feasible and equally difficult for the RAM designer to anticipate the design constraints. The best outcome will result from a partnership between architects and the RAM designers.

One high-level trade-off in most microprocessor designs is that between adding more functionality and adding more cache to the chip. This is a difficult choice. Increases in on-chip cache size can bring significant improvement in architectural throughput. However, larger caches are slower and occupy chip area that could be used for more or larger architectural components that can also increase throughput. Sometimes small very fast primary caches are backed by second-level on-chip caches that are larger and slower. This approach gives the designers flexibility to optimize the second level cache for power and area efficiency. However, the other approach of a carefully crafted large fast single-level data cache has also been used with considerable success. Overall, decisions about how much RAM to use and how to organize it are difficult. Successful examples of high-performance microprocessors vary widely in these choices. Section 1.4 explores this issue further by comparing the choices of three actual designs.

### 1.3 CHOOSING THE CYCLE TIME OF A HIGH-PERFORMANCE MICROPROCESSOR

Choosing the cycle time of a new processor design is probably the single most important decision to be made when beginning a new design. At the highest level, the decision is a trade-off between architectural efficiency (instructions per cycle, executing a given type of application) and processor clock frequency. The product of the two is the net performance in instructions per second. Given the difficulty and art required to find a good architecture, and the amount of work required to estimate the speed of CMOS structures, there is no easy answer to the question.

The art of processor design is such that there is no simple procedure that allows the architect to determine the advantage to be gained from a given cycle time change. It takes considerable time to accurately evaluate the possibilities.

Beyond the architectural evaluation, it is very difficult to accurately estimate the circuit delay of each architectural component, particularly when the component is a new structure or a new arrangement of structures. If a proposed component is an exact or scaled copy of a component of an existing design, a reasonably accurate estimate is possible. But, before a new component is finally implemented, it can be quite difficult to accurately estimate its delay characteristics. An initial estimate can be arrived at quickly by comparison to well-understood components. Of course, major portions of new designs are copied more or less exactly from a previous design, but new designs usually

add significantly to what has been done before. Working estimates can easily be subject to a 10–20% error, making accurate evaluation of trade-offs difficult.

Experienced microprocessor architecture and design teams reach the cycle time decision through a combination of judgment and data. Judgment is not simply applied after exhaustive data collection. There are too many possibilities to explore. At each step, designers and architects decide which direction to explore, which alternative to examine, and they make small decisions adjusting the design center of one or a small number of working proposals as they go.

### 1.3.1 Critical Loops

Similar to the idea of a critical path, a critical loop is logic that is required by the architecture to repeat or loop in a given number of cycles, usually one cycle. Architects should identify critical loops early in the design process because these will ultimately be the paths that set the cycle time.

Some functional latencies can be pipelined over multiple cycles with only minor loss in architectural throughput. Critical loops are different. If the logic in critical loops is unable to operate in the allocated loop time, a significant performance loss is the result. The execute and bypass loop in a typical microprocessor is a critical loop because a two-cycle latency for basic operations like ADD would mean very significant loss in architectural throughput for almost all integer programs. Another example is the latency from execution of a load instruction to the execution of the first instruction using the data fetched by the load instruction. It is a critical loop even though the latency is often more than a single cycle.

## 1.4 COMPARISON OF PA8000, 21164, AND 21264 PROCESSORS

It is instructive to compare the PA8000 [7] from Hewlett-Packard Corporation to the Alpha 21164 [2] and 21264 [8] designs from Digital Equipment Corporation. The on-chip cache and cycle time decisions are very different. The issues that stand out in these comparisons are the decision for complex out-of-order execution versus simpler in-order execution, the complexity of the cache organization choice, and the trade-off between cycle time and other architectural features.

The 21164 and PA8000 were both implemented in roughly equivalent 0.5 µm CMOS technology. The PA8000 die is about 10% larger than the 21164, and both are probably about as large as can be manufactured in their technology generation. In terms of physical technology, they are quite similar while in architecture they are quite different.

The PA8000 has no on-chip cache, choosing instead to use separate large tightly coupled off-chip primary instruction and data caches. These off-chip caches, built from very fast SRAM chips, were typically much larger than could have been implemented on chip. CPU modules with cache sizes ranging from 0.5 Mbyte up to multiple Mbytes were built.

In contrast, the 21164 has on-chip primary instruction and data caches that are 8 Kbytes each, and a second-level mixed instruction and data cache that is 96 Kbytes. These caches require about 30% of the 21164 die area.

The designers of the 21164 exploit the flexibility that CMOS offers in the caches. The primary data cache is dual-ported for reads, supporting execution of two load

instructions per cycle regardless of address, and the primary instruction cache bus width is wide enough to supply four instructions per cycle. The second-level cache is three-way set associative and is pipelined so that it can continuously provide data to either primary cache at the rate of two 64-bit words (or four 32-bit instructions) per CPU cycle.

The PA8000 caches are single ported, though the data cache data bus is wide enough to support two loads accessing adjacent memory locations simultaneously, and the instruction cache data bus is equally wide to support four instructions per cycle from the instruction cache.

The 21164 operating frequency is 300 MHz, while the PA8000 operates at 180 MHz. Even with its slower cycle time, the PA8000 is significantly higher in performance. The published SPECint95 result for the PA8000 at 180 MHz is 11.8. A reasonable estimate of the 21164 at 300 MHz is 9 SPECint95. (The 21164 paper gives the SPECint92 performance results but not the SPECint95 results. However, a version of the Alpha 21164 implemented in 0.35 micron CMOS has a SPECint95 of 13.3 at 433 MHz [9]. The 9 SPECint95 estimate above is calculated by derating the 13.3 SPECint95 figure by the ratio of operating frequencies and adjusting a bit for architectural enhancements in the later 21164 version.)

Compare the work done per cycle on the SPECint95 benchmark suite for these designs (SPECint95/MHz). The PA8000 delivers 0.066 SPECint95/MHz, while the 21164 delivers 0.031 SPECint95/MHz. The 21164 must operate at over twice the frequency to beat the PA8000 in SPECint95 performance. (In fact, the 0.35 micron version of the 21164 did just that.)

The PA8000 SPECint95/MHz advantage over the 21164 is a direct result of very different cycle time and on-chip complexity decisions. Placing the PA8000 primary caches off-chip left room for a complicated out-of-order execution design on the chip, and the resulting larger cache sizes enhanced architectural efficiency significantly. However, the maximum cycle time of the processor was limited significantly by the speed of the off-chip caches.

The Alpha 21164 designers made distinctly different choices for operating frequency, area on the chip devoted to cache, architectural complexity and throughput. The 21164 is an in-order execution processor that is considerably simpler than the PA8000 design. The designers chose a faster operating frequency and traded off instruction throughput per MHz to achieve the lower cycle time. Moreover, the on-chip caches were able to operate at the 21164's higher speed.

In contrast to the Alpha 21164, the Alpha 21264 is a complex out-of-order execution microprocessor implemented in 0.35  $\mu$ m technology. It only contains primary caches on the die (separate instruction and data caches), and the secondary cache is connected to the chip via a high bandwidth interconnection bus. Its SPECint95/MHz figure is approximately 0.052. An enhanced version of the PA8000 processor that was implemented in 0.25  $\mu$ m technology (and added two on-chip caches) has a 40% better SPECint95/MHz ratio (0.073) [10]. However, the 21264 has an approximately 70% higher operating frequency when normalized to similar technology. The net performance of the 21264 would then be roughly 20% better in comparable technology.

The 21264 is notable as an implementation in which the complexity of out-of-order did not translate into slower cycle time. In fact the designers of the 21264 set their cycle time such that about 10 gate delays could fit in a cycle, along with necessary latch delay [11]. (In contrast, the 21164 design allowed about 12 gate delays in a cycle, with latch

delay.) Achieving this cycle time clearly required trade-offs in the architecture. One such trade-off is exhibited in the less than ideal behavior of the instruction issue arbiter discussed earlier in this chapter. Another interesting compromise was that the execution units were divided into two clusters, and result bypassing between clusters took two cycles while bypassing within a cluster took one cycle. This trade-off improves cycle time at the expense of instructions per cycle throughput, though the architectural loss is minimized because the arbiter algorithm is able to compensate to some degree. See the original article for more detail [3].

## 1.5 TREND IN INTERCONNECT RESISTANCE

The trend in interconnect delay due to resistance is becoming a significant problem for microprocessor designers. As CMOS technology scales from one generation to the next the product of interconnect resistance and load capacitance ( $RC$  delay) is not scaling with technology. This is a serious constraint for architecture that gets relatively worse with each new CMOS technology generation.

In classical CMOS scaling theory, every dimension and every capacitance is reduced by a scaling factor in each generation. For the purposes of this discussion, we assume the scaling factor between successive generations is a constant 0.7. Consider a section of interconnect under this scaling rule. The width scales by 0.7. The height tends to scale by 0.7, and for the identical circuit, the length of the interconnect scales by 0.7 also. Resistance is proportional to length divided by area. So in a single theoretical generation of scaling, interconnect resistance increases by  $\frac{1}{0.7} = 1.43$ . Meanwhile, the capacitance on a given node scales by 0.7, so  $RC$  stays constant. However, a gain in speed of  $\frac{1}{0.7}$  is expected with each process generation.

In practice,  $RC$  is scaling somewhat rather than staying constant over process generations. Several factors account for this. First, interconnect heights have not been scaled as rapidly as widths. The desired result is that resistance doesn't increase by as much as 1.43 in a generation, and the less desirable side effect is that lateral capacitance has become a larger component of total wire capacitance. Second, copper interconnect has been introduced in many processes, providing a one-time decrease in resistance. Third, low K dielectrics are being introduced. This provides an additional reduction in capacitance. Given these factors, process designers are able to provide some scaling in  $RC$  delay, though eventually, fundamental laws will prevent further gains.

A recent article [12] analyzes the interconnect  $RC$  delay trend in terms of the percentage of a chip reachable within one clock cycle. This percentage decreases rapidly as the  $RC$  continues not to scale with technology generations. Regardless of the exact rate of the trend, it is clear interconnect delay will have an increasing impact on architecture.

### 1.5.1 Interconnect Inductance

As later chapters discuss, inductance of on-chip interconnect is becoming an increasingly important problem in on-chip signaling. Analysis of inductance is far less straightforward than that of capacitance and resistance. It becomes more significant both as signals edge rates increase and generally only affects longer signals. Inductance will soon be another source of limitation on the length of on-chip signals, requiring more frequent signal repeaters with a resulting increase in total delay.

### 1.5.2 Architectural Consequences of Interconnect Trends

What will architects do about interconnect delay? In the end, they will have to reverse the present trend toward larger architectural components and longer interconnects (e.g., larger instruction queues, larger memory reorder buffers, more transistors devoted to one processor's logic, larger primary caches). In the shorter term, pipelines will get longer and architects will depend more on prediction to manage pipeline latency. They will also explore parallel, special-purpose architectures that serve parallel applications better.

Applications like graphics and multimedia contain a greater percentage of predictable (therefore parallelizable) computation. Execution of these applications can perform well in deep pipelines or parallel computation units. Applications like web serving, file serving, and transaction processing are composed of many independent threads of execution, and overall throughput is the primary performance measure, not latency in any one thread. These applications should perform well with parallel computation approaches.

## 1.6 TREND IN POWER CONSUMPTION

In classical scaling theory, power consumption should not be that significant a problem, but in actual practice we see an alarmingly rapid upward trend in power consumption for high-performance microprocessors. From one perspective, the main reason is that power supply voltage is not dropping fast enough to support perfect scaling.

The trend of increasing power consumption per chip poses difficult problems in high-end microprocessors. There are several major concerns with high power consumption. The first is cooling. There are physical limits in packaging technology that limit cooling and make cooling high power chips expensive, difficult, and impossible beyond certain limits. Additionally, supply current is growing fast as power consumption increases and  $V_{dd}$  is simultaneously reduced. Larger supply current and rapid rates of change in that current ( $dI/dt$ ) present significant engineering challenges.

Energy consumption per instruction executed is considered the more important metric in portable computers because “sleep” modes can be used to ensure that the processor consumes minimum power when there is no computational task to execute. In other cases, the cost of power supplies and cooling capability in a system limits the acceptable power level for a commercially competitive product.

### 1.6.1 Power Estimation and Scaling

When the 21064 Alpha microprocessor was introduced in 1992 in 0.75  $\mu\text{m}$  CMOS, its power consumption was 30 W at 200 MHz with a  $V_{dd}$  of 3.3 V [13]. This was an astonishingly high number at the time. The power in Alpha processors has been increasing significantly with each generation, and although Alpha tends to be the extreme in a given generation, the same trend is noticeable in other microprocessor families. For example, the designers of the Ultrasparc-III expect that processor will consume 70 W at 600 MHz in a 1.8 V 0.25  $\mu\text{m}$  technology, when fabricated [14].

A simplistic analysis of power consumption in CMOS digital logic is based on the approximation that power consumption is entirely due to charging and discharging of

circuit nodes. An average capacitance  $C_{total}$  charged per cycle is assumed for typical operation. The resulting well-known equation for power is  $Power = C_{total}V_{dd}^2 f$ , where  $V_{dd}$  is the power supply voltage and  $f$  is the operating frequency. The approximation neglects short-circuit current during switching transitions, DC current in unusual circuits or analog components, IO circuits, and leakage current, all of which can be significant. However, the simple formula is a good approximation that allows us to isolate the effects of power supply voltage and operating frequency on power consumption. One can compare the average  $C_{total}$  charged per cycle per unit area (call it  $C_{unit-area}$ ) in various designs and consider the theoretical effect of scaling on this parameter to reach conclusions about power consumption in future designs.

Begin with the theoretical effect of scaling on the parameter  $C_{unit-area}$ . Consider a circuit in a given area in a given technology generation. When scaled to the next technology generation, the capacitance of each node drops by the 0.7 scaling factor and the area of the circuit drops by  $0.7^2 = 0.49$ . An area 0.49 times as large contains 0.7 times the capacitance, so capacitance per unit area increases by 1.43.

In classical scaling,  $V_{dd}$  would be reduced by 0.7 with each technology generation and operating frequency would rise by 1.43. Power per unit area would remain constant with technology scaling. However, actual trends in manufacturing technologies are that  $V_{dd}$  tends to scale more slowly than this [1]. The reasons for this are fundamental problems in semiconductor process engineering. Reducing  $V_{dd}$  is clearly desirable, but it is challenging to design transistors that have acceptable characteristics at a lower  $V_{dd}$ . As a result, power per unit area tends to rise slowly with each process generation.

The problem is exacerbated by two other factors in high-end microprocessors seen in recent generations: increasing die area in high-end microprocessors and increased use of high-performance circuit design approaches that consume comparatively more power per unit area. This helps explain the rapid rise in power dissipation in these designs.

Let us compare the  $C_{unit-area}$  of some processors. The 21164 dissipates 50 W at 300 MHz in a 3.3 V 0.5  $\mu\text{m}$  process, and the die was 16.5 mm by 18.1 mm. The  $C_{unit-area}$  for this chip is 50 pF/mm<sup>2</sup>. A Power-PC chip designed for mobile and high volume desktop applications is reported [15] to dissipate 5 W at 250 MHz in a 2.5 V 0.3  $\mu\text{m}$  process, and the die area was 66.5 mm<sup>2</sup>. Its  $C_{unit-area}$  is also about 50 pF/mm<sup>2</sup>. However, it is roughly two process generations later. The 21164 figure in 0.25  $\mu\text{m}$  CMOS would be predicted by scaling theory to be about 100 pF/mm<sup>2</sup> in a comparable technology. The aggressively high-performance Alpha design is significantly worse than the power-conscious design, though this analysis gives little insight into why that is so.

The Ultrasparc-III designers' expectations are 70 W dissipation at 600 MHz in a 1.8 V 0.25  $\mu\text{m}$  CMOS process for a 360 mm<sup>2</sup> die area [14]. This gives a  $C_{unit-area}$  of 100 pF/mm<sup>2</sup>, identical to the 21164 figure scaled to a similar technology.

The  $C_{unit-area}$  figure is clearly dependent on differences in logic design style and architectural approach. For example, dynamic logic circuits might well consume more power per area than static complementary logic for a typical logic function. At the architecture level, techniques such as pipelining, parallelism, and speculation typically lead to designs in which more logic sections are simultaneously active. An example trade-off is one in which two results are calculated knowing that only one will be needed, but not knowing which until after the calculations begin. This is a direct trade-off between performance and power.

For a direct shrink to new technology or for a consistent design style,  $C_{unit-area}$  is a useful predictor of power consumption. It is straightforward to scale the figure to new technology. It should be quite accurate if the general logic design and architectural approaches are similar. Also, as a technology normalized metric, it could be used to estimate the power consumption with different logic design styles, though care must be taken in the analysis since power consumption is certainly not uniform across the many sections and structures in a chip.

Estimating the effect of architectural decisions on power dissipation generally requires a detailed accounting of the sections that switch simultaneously and an at least rough estimate of the capacitance that is switching. If the sections are large and regular, this can be fairly straightforward. Often though, estimating the effect of an architectural choice on power is quite difficult.

A final point on scaling is to consider what happens as a given processor design is simply implemented in the next generation process with few changes. Nominal scaling theory predicts all capacitance decreasing by a typical 0.7 scaling factor. Nominal operating frequency scales up by the inverse of the scaling factor to 1.43 times. So these two components of the power formula simply cancel. In effect,  $V_{dd}$  scaling determines the power reduction. As mentioned above, however,  $V_{dd}$  tends not to scale as rapidly as scaling theory calls for, so the power reduction will be less than the theoretical 50%. Nonetheless, power decreases quadratically with  $V_{dd}$  and the power savings are typically impressive.

### 1.6.2 Energy-Delay Product

The energy delay has been suggested as a metric [16]. More specifically, the average energy per instruction is multiplied by the average inter-instruction delay. Each term is an interesting metric, and their product reflects the success of the design at optimizing each simultaneously. The authors of that study find that the energy-delay product is quite consistent (within a factor of about 2, normalizing for technology) across a number of general purpose designs, suggesting a roughly inverse relationship between energy efficiency and performance.

The energy-delay metric is certainly interesting and may be useful in roughly estimating power of proposed designs and examining various trade-offs. Nevertheless, examining  $C_{unit-area}$  is the most straightforward way to estimate power trends in an evolving family of products for which trade-offs between die area, power consumption, and performance are made similarly in each product generation.

### 1.6.3 Physical Limits of Heat Dissipation

Packaging technology has managed to keep up with the increased dissipation requirements of recent microprocessors, but there are fundamental limits beyond which cooling becomes very difficult. Both total power and power per unit area present engineering challenges. If present trends continue, expensive active cooling technology may soon be required. Detailed discussion is beyond the scope of this chapter, but packaging and cooling are key issues in today's high-performance microprocessor designs.

### 1.6.4 Active Power Control

The designers of the above-mentioned Power-PC added an active power control strategy. They built an analog temperature sensor with a user programmable threshold. When the temperature on the chip reaches the trigger threshold, a thermal management interrupt is delivered. They also added programmable hardware to limit the rate of instruction fetch to a fraction of the normal processor rate. An interrupt handler, responding to the thermal management interrupt, reduces the instruction fetch rate to a software programmable limit, resulting in a reduction in chip power dissipation.

### 1.6.5 $V_{dd}$ Reduction

Reduction in  $V_{dd}$  is another power reduction technique. The performance of typical CMOS logic circuits drops off more slowly than power consumption does as  $V_{dd}$  is reduced. Power consumption drops linearly with frequency and quadratically with voltage so that the power reduction rate is roughly cubic with voltage drop when the consequential reduction in frequency is included. Designers have used this fact to establish low  $V_{dd}$  operating points with much lower power consumption than at nominal  $V_{dd}$  while maintaining a performance level that is acceptable in the intended application.

$V_{dd}$  reduction is an important tool in microprocessor design for managing power in a flexible way. At the high end, it offers a safety valve in case the chip's power consumption is too large for some application. Even in low-end power-conscious design, it offers an additional trade-off between performance and power consumption that may allow one design to meet the different performance needs of multiple applications with widely varying power budgets.

Of course, not all logic circuit styles slow down equally as  $V_{dd}$  is reduced. There is a point beyond which the circuits simply won't function, and that point may be different for each logic circuit style. Designers expecting to exploit  $V_{dd}$  scaling should understand the effects of reduced supply voltage on the logic circuit styles employed in the design. Good planning early in the design will help make power control by  $V_{dd}$  reduction successful.

### 1.6.6 Ultimate Impact of Power

Present power consumption trends in high-performance microprocessors clearly cannot be supported forever. Both dissipation and electrical integrity of the power source are problematic. Ultimately, power, power per unit area, or supply current delivery will create hard feasibility limits that will not permit present trends in high-end processor architecture and design to continue.

## 1.7 ADVANCED PACKAGING

Flip-chip technology packaging now allows high-performance microprocessors to be built with many more power and I/O connections than in the past. Besides helping solve the power delivery engineering problems that come with high-frequency operation and the signal integrity challenges arising from fast I/O signal edges, the greatly increased number of I/O connections brings new possibilities to system architects.

In a typical flip-chip packaging technology, bond pads are placed in an array over the surface of the die. Solder bump technology is used to mechanically bond the pads on the die to corresponding pads on a multilayer package. The electrical characteristics of this connection are much better than in traditional wire-bonded packaging (particularly for inductance).

Wire-bond connections can only be made at or near the edge of a die, while flip-chip connections are made essentially anywhere on the die. In the wire-bonded technology, I/O signals had to be routed to the edge of the die, usually incurring additional delay as a result. In flip-chip technology it should be possible to reduce this latency by carefully planning the bump locations. Also, with wire bonding, the number of connections is limited by the minimum bond spacing and the length of the chip's periphery, while with flip-chip bonding the limit is proportional to the die area. The net result is that where I/O connections were precious with wire bonding, they are less so with flip-chip bonding.

Given the increased number of transistors available to the designer, and the ever increasing gap between off-chip and on-chip signaling rates, the dominant trend is for all cache memories to be on the microprocessor die. The side effect of this trend is that no I/O signal connections are needed for caches, freeing up I/O for main memory connection and other purposes.

With all these pins, what will designers do? The Alpha 21364 designers offered one strategy in 1998 [17]. They outlined a high-performance microprocessor with an integrated scalable memory subsystem. The design has a high-bandwidth Direct RAMBUS (described below) memory system connected directly to each processor chip. In addition, there are four interprocessor connection ports on each chip. These ports allow unbuffered (glueless) connections to adjacent processors in a system. Processors would typically be arranged in a rectangular mesh with the ends wrapped around to the other side, forming a torus. Bandwidth on the ports is very high, and latency between chips is low. This example suggests that large numbers of pins coupled with high levels of integration will lead to higher levels of multiprocessor performance, scalability, and cost efficiency in high-end computer systems.

Until today, main memory components in computer systems of all types were usually standard DRAM chips. Data and control signals at DRAM components are very slow compared to the signals at the microprocessor's pins. The resulting systems have a hierarchy of signaling with relatively few very fast signals at the top of the hierarchy near the processor and several times as many signals operating at relatively slow signaling rates at the bottom where the DRAMS are connected. Bandwidth is matched at each level, trading signaling rate for parallel signaling. In these systems, memory access time is relatively larger in each new processor generation, since memory components are not getting faster nearly as fast as microprocessors.

Recent technology (Direct RAMBUS being the main example) brings memory components with extremely high signaling rates and changes this picture in an important way. There is no longer any need to fan out to the slower components, the DRAMS can be connected directly to the microprocessor, and there is a consequent reduction in latency and component count. Despite the improvement, main memory latency in Direct RAMBUS technology is still large compared to the processor cycle, so memory latency continues to be a significant problem. Also, bandwidth required by applications will continue to grow, and it is not yet clear how well new DRAM technology will track to these requirements.

## 1.8 CONCLUSION

High-performance CMOS microprocessor designs use a mixture of advanced logic design styles, RAM structures, arrays, datapaths, and customized RAM-like structures to achieve high-speed operation. Architects must be cognizant of these techniques to successfully design a processor. The cost functions and other limitations arising from the technology are neither simple nor obvious from an abstract viewpoint. As technology evolves, successful designs will continue to require architecture that reflects careful consideration of the strengths and limitations of the underlying technology.

Increased integration and packaging advances offer the opportunity for increased integration leading to much better and larger scale multiprocessor systems. This trend is one among several that supports the increased use and importance of parallel computing.

On-chip interconnect technology and power consumption trends are leading to new limits to feasibility. In the end, trends in power consumption, interconnect resistance, and interconnect inductance will likely force an end to the current trends in processor design toward larger, more complex uniprocessors. Architects will have to exploit available parallelism to work around these limitations. The most likely outcome would seem to be increased use of distributed processing.

High-performance processor design will continue on its current dynamic and enormously successful growth trend for the foreseeable future. The application parallelism needed to overcome limits of physical technology is particularly evident in the applications that most demand performance growth. Many variations on distributed processing have been or are now being explored in research and developed as commercial products. An issue continues to be ease of programming such complex systems, though there is clear success in particular applications such as commercial transaction processing. Arguably, solutions to these problems and acceptance of the increased programming difficulty will come as a result of significant reduction in the rate of growth of uniprocessor performance, and that reduction appears to be inevitable. Despite the physical limits that arise, new architectures and continued technology advances will enable designers to continue to deliver increased performance to users in future product generations.

## REFERENCES

- [1] K. Bernstein, et al., *High Speed CMOS Design Styles*. Kluwer, Boston, 1998.
- [2] B. Benschneider, et al., "A 300 MHz 64-b Quad-Issue CMOS RISC Microprocessor," *JSSC*, vol. 30, no. 11, Nov. 1995.
- [3] J. A. Farrell and T. C. Fischer, "Issue Logic for a 600-MHz Out-of-Order Execution Microprocessor," *JSSC*, vol. 33, no. 5, May 1998.
- [4] S. Naffziger, "A Sub-Nanosecond 0.5  $\mu$ m 64 b Adder Design," *ISSCC*, 1996, p. 362.
- [5] J. Silberman, et al., "A 1.0-GHz Single-Issue 64-Bit PowerPc Integer Processor," *JSSC*, vol. 33, no. 11, Nov. 1998.
- [6] N. Vasileghiu, et al., "200-MHz Superscalar RISC Microprocessor," *JSSC*, vol. 31, no. 11, Nov. 1996.
- [7] N. Gaddis and J. Lotz, "A 64-b Quad-Issue CMOS RISC Microprocessor," *JSSC*, vol. 31, no. 11, Nov. 1996.

- [8] B. Gieseke, et al., "A 600 MHz Superscalar RISC Microprocessor with Out-of-Order Execution," *ISSCC*, Feb. 1997, pp. 176-177.
- [9] P. E. Gronowski, et al., "A 433-MHz 64-b Quad-Issue RISC Microprocessor," *JSSC*, vol. 31, no. 11, Nov. 1996.
- [10] P. Barnes, "A 500 MHz 64-b RISC CPU with 1.5 MB On-Chip Cache," *ISSCC*, 1999, pp. 86-87.
- [11] Verbal Communication with Bill Bowhill of Compaq Computer Corporation, May 1999.
- [12] D. Matzke, "Will Physical Scalability Sabotage Performance Gains?," *IEEE Computer Magazine*, vol. 30, no. 9, Sep. 1997.
- [13] D. Dobberpohl, et al., "A 200 MHz 64-b Dual-Issue CMOS Microprocessor," *JSSC*, vol. 27, no. 11, Nov. 1992.
- [14] T. Horel and G. Lauterbach, "Ultrasparc-III: Designing Third-Generation 64-Bit Performance," *IEEE Micro*, May-June, 1999.
- [15] G. Gerosa, et al., "A 250 MHz 5-W PowerPC Microprocessor with On-Chip L2 Cache Controller," *JSSC*, vol. 32, no. 11, Nov. 1997.
- [16] R. Gonzalez and M. Horowitz, "Energy Dissipation in General Purpose Microprocessors," *JSSC*, vol. 31, no. 9, Sep. 1996.
- [17] P. Bannon, "Alpha 21364: A Scalable Single-Chip SMP," *Microprocessor Forum*, Oct. 14, 1998.

PART  
**II**

## **TECHNOLOGY ISSUES**

Yuan Taur  
IBM

## 2.1 MOSFET SCALING THEORY

CMOS technology evolution in the past twenty years has followed the path of device scaling for achieving density, speed, and power improvements. Elementary theory [1] tells us that MOSFET transconductance per device width increases with shorter channel lengths, that is, with reduced source-to-drain spacing. The intrinsic capacitance of a short-channel MOSFET is also lower, which makes it easier to switch. However, for a given process, channel length cannot be arbitrarily reduced even if allowed by lithography because of short-channel effects. For digital applications, the most undesirable short-channel effect is a reduction in the gate threshold voltage at which the device turns on, especially at high drain voltages. Full realization of the benefits of the new high-resolution lithographic techniques therefore requires the development of new device designs, technologies, and structures which can be optimized to keep short-channel effects under control at very small dimensions.

### 2.1.1 Constant-Field Scaling

In constant-field scaling [2], it was proposed that one can keep short-channel effect under control by scaling down the vertical dimensions, for example, gate insulator thickness, junction depth, along with the horizontal dimensions, while also proportionally decreasing the applied voltages and increasing the substrate doping concentration (decreasing the depletion width). This is shown schematically in Fig. 2.1. The principle of constant-field scaling lies in scaling the device voltages and the device dimensions (both horizontal and vertical) by the same factor,  $\kappa (> 1)$ , such that the electric field ( $\mathcal{E}$ ) remains unchanged. This assures that the reliability of the scaled device in terms of hot-carrier injection is not worse than the original device.

#### 2.1.1.1 Rules for Constant-Field Scaling

Table 2.1 shows the scaling rules for various device parameters and circuit performance factors. Doping concentration  $N_a$  must be increased, by the scaling factor,  $\kappa$ , in order to keep Poisson's equation,  $\nabla \cdot \mathcal{E} = -qN_a/\varepsilon_{si}$ , invariant with respect to scaling.

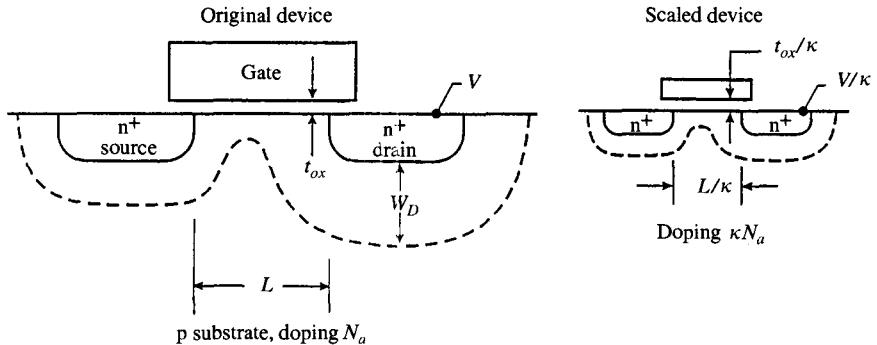


Figure 2.1 Principles of MOSFET constant-electric-field scaling.

**TABLE 2.1** Scaling of MOSFET Device and Circuit Parameters

	MOSFET device and circuit parameters	Multiplicative factor ( $\kappa > 1$ )
Scaling assumptions	Device dimensions ( $t_{ox}, L, W, x_j$ ) Doping concentration ( $N_a, N_d$ ) Voltage ( $V$ )	$1/\kappa$ $\kappa$ $1/\kappa$
Derived scaling behavior of device parameters	Electric field ( $\epsilon$ ) Carrier velocity ( $v$ ) Depletion layer width ( $W_d$ ) Capacitance ( $C = \epsilon A/t$ ) Inversion layer charge density ( $Q_i$ ) Current, drift. ( $I$ ) Channel resistance ( $R$ )	1 1 $1/\kappa$ $1/\kappa$ 1 $1/\kappa$ 1
Derived scaling behavior of circuit parameters	Circuit delay time ( $\tau \sim CV/I$ ) Power dissipation per circuit ( $P \sim VI$ ) Power-delay product per circuit ( $P \times \tau$ ) Circuit density ( $\propto 1/A$ ) Power density ( $P/A$ )	$1/\kappa$ $1/\kappa^2$ $1/\kappa^3$ $\kappa^2$ 1

Here  $\epsilon_{si}$  is the permittivity of silicon and  $q$  is the electronic charge. The maximum drain depletion width,

$$W_D := \sqrt{\frac{2\epsilon_{si}(\psi_{bi} + V_{dd})}{qN_a}} \quad (2.1)$$

scales down approximately by  $\kappa$  provided that the power supply voltage,  $V_{dd}$ , is much greater than the built-in potential,  $\psi_{bi}$ . All capacitances (including wiring load) scale down by  $\kappa$  because they are proportional to area and inversely proportional to thickness. Charge per device ( $\sim C \times V$ ) scales down by  $\kappa^2$  while the inversion-layer charge density (per unit gate area),  $Q_i$ , remains unchanged after scaling. Since the electric field at any given point is unchanged, the carrier velocity ( $v = \mu \times \mathcal{E}$ ) at any given point is also unchanged (mobility is the same for the same vertical field). Therefore, any velocity saturation effects will be similar in both the original and the scaled devices.

The drift current per MOSFET width,

$$\frac{I_{\text{drift}}}{W} = Q_i v = Q_i \mu \varepsilon \quad (2.2)$$

is unchanged with respect to scaling. This means that the drift current scales down by  $\kappa$ , consistent with the behavior of both the linear and the saturation MOSFET currents. A key implicit assumption is that the threshold voltage also scales down by  $\kappa$ . Note that the velocity saturated current also scales the same way because the saturation velocity is a constant, independent of scaling. However, the diffusion current per MOSFET width,

$$\frac{I_{\text{diff}}}{W} = D \frac{dQ_i}{dy} = \mu \frac{kT}{q} \frac{dQ_i}{dy} \quad (2.3)$$

where  $D$  is the diffusivity and  $kT$  the thermal energy, scales up by  $\kappa$  because  $dQ_i/dy$  is inversely proportional to channel length ( $y$ -axis is the current transport direction). Therefore, the diffusion current does not scale down the same way as the drift current. This has significant implications in the nonscaling of MOSFET subthreshold currents, as will be discussed in Section 2.2.1.

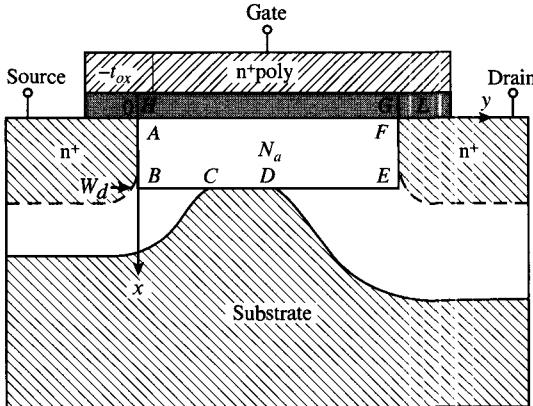
### 2.1.1.2 Effect of Scaling on Circuit Parameters

With both the voltage and the current scaled down by the same factor, it follows that the active channel resistance  $R$  of the scaled-down device remains unchanged. It is further assumed that parasitic resistance is either negligible or unchanged in scaling. The circuit delay, which is proportional to  $R \times C$  or  $C \times V/I$ , then scales down by  $\kappa$ . This is the most important conclusion of constant-field scaling: Once the device dimensions and the power supply voltage are scaled down, the circuit speeds up by the same factor. Moreover, power dissipation per circuit, which is proportional to  $V \times I$ , is reduced by  $\kappa^2$ . Since the circuit density has increased by  $\kappa^2$ , the power density, that is, the active power per chip area, remains unchanged in the scaled-down device. This has important technological implications in that, in contrast to scaled bipolar devices, packaging of the scaled CMOS devices does not require more elaborate heat sinking. The power-delay product of the scaled CMOS circuit shows a dramatic improvement by a factor of  $\kappa^3$  (Table 2.1).

## 2.1.2 Two-Dimensional (2-D) Scale Length Theory

The above simplified picture of 2-D effect scaling becomes inadequate when the power supply voltage is scaled to a level comparable to the built-in potential or the silicon bandgap  $\approx 1$  V, as is the case with sub-0.25  $\mu\text{m}$  CMOS systems. Even though Poisson's equation may be invariant when the doping concentration is increased by the scaling factor,  $\kappa$ , the boundary conditions dictated by the built-in potential of p-n junctions do not scale properly. Figure 2.2 shows the essential 2-D aspects of a short-channel MOSFET [3]. A key parameter is the maximum gate depletion width,  $W_{dm}$ , within which mobile carriers (holes in the case of nMOSFETs) are swept away by the applied gate field. For uniformly doped cases,

$$W_{dm} = \sqrt{\frac{4\varepsilon_{si}kT \ln(N_a/n_i)}{q^2 N_a}} \quad (2.4)$$



**Figure 2.2** Simplified geometry for analyzing 2-D effects in a short-channel MOSFET.

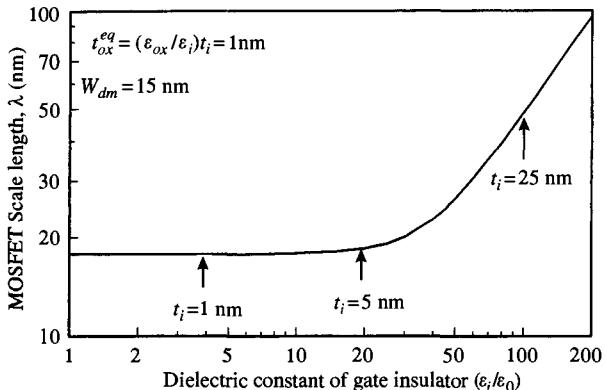
where  $n_i$  is the intrinsic carrier concentration. A rectangle is formed by the boundary of the gate depletion region, the gate electrode, and the source and drain regions, as depicted in Fig. 2.2 [3]. 2-D effects can be characterized by the aspect ratio of this rectangle. When the horizontal dimension, that is, the channel length, is at least twice as long as the vertical dimension, the device behaves like a long-channel MOSFET with good short-channel behavior. For channel lengths shorter than that, 2-D effect kicks in and the source-channel potential barrier critical for setting the threshold condition is increasingly controlled by the drain voltage instead of by the gate voltage alone.

The rectangular box consists of a silicon region of thickness  $W_{dm}$  and an oxide region of thickness  $t_{ox}$ . At the interface, the vertical fields ( $\mathcal{E}_x$ ) obey the boundary condition,  $\varepsilon_{si}\mathcal{E}_{x,si} = \varepsilon_{ox}\mathcal{E}_{x,ox}$ , where  $\varepsilon_{si}$ ,  $\varepsilon_{ox}$  are the permittivity of silicon and oxide, respectively. As far as the vertical field is concerned, the oxide region can be replaced by a silicon region with an equivalent thickness of  $(\varepsilon_{si}/\varepsilon_{ox})t_{ox}$ . Therefore, for thin gate insulators such as in the case with  $\text{SiO}_2$ , the vertical height of the rectangular box can be approximated by  $W_{dm} + (\varepsilon_{si}/\varepsilon_{ox})t_{ox}$ , where  $\varepsilon_{si}/\varepsilon_{ox} \approx 3$  is the ratio between the dielectric constants of silicon and oxide. This means that the minimum channel length is  $L_{min} \approx 2(W_{dm} + 3t_{ox})$  [1]. To scale MOSFETs to shorter channel lengths, then, it is important to scale both  $W_{dm}$  and  $t_{ox}$  proportionally.

The above approximation assumes that the vertical field is the dominant component at the interface between the silicon and the gate insulator. This may not be the case for thick oxides or for high- $\kappa$  (high dielectric constant) gate insulators which may be needed when the tunneling current through thin oxides becomes prohibitively high (Sect. 2.2.2). In principle, high- $\kappa$  gate insulators can be physically thicker, thus alleviating the tunneling problem, while still delivering the same field effect or gate capacitance measured by the ratio  $\varepsilon_i/t_i$ , where  $\varepsilon_i$  and  $t_i$  are the permittivity and the thickness of the insulator film. The above 1-D boundary condition for vertical fields is sufficient for describing a long-channel device. In short-channel devices, however, 2-D effects are important and one must take both the lateral and the vertical fields into account.

For lateral fields ( $\mathcal{E}_y$ ) tangential to the interface, the boundary condition is  $\mathcal{E}_{y,si} = \mathcal{E}_{y,ox}$ , independent of the dielectric constants. By properly matching the boundary conditions of both components of electric fields at the silicon-insulator interface, one can derive a scale length  $\lambda$  that is a solution to the following equation [4]:

$$\varepsilon_{si} \tan(\pi t_i/\lambda) + \varepsilon_i \tan(\pi W_{dm}/\lambda) = 0 \quad (2.5)$$



**Figure 2.3** Plot of scale length versus dielectric constant ( $\epsilon_i/\epsilon_0$ ) for an equivalent oxide thickness ( $\epsilon_{ox}/\epsilon_i$ ) of 1 nm and a silicon depletion depth  $W_{dm}$  of 15 nm, where  $\epsilon_0$  is the permittivity of free space, and  $\epsilon_{ox}$  is that of  $\text{SiO}_2$ .

Here  $W_{dm}$  again is the depth of the depletion region in silicon.  $\lambda$  also goes into the length-dependent term of the maximum potential barrier in a MOSFET of channel length  $L$ :  $\propto \exp(-\pi L/2\lambda)$  [3]. The ratio  $L/\lambda$  is a good measure of the severity of the 2-D effect. For the short-channel  $V_t$  roll-off and the drain-induced barrier lowering (DIBL) to be acceptable, the above exponential factor must be much less than one which means the minimum useful channel length is about 1.5–2.0 times  $\lambda$ .

The lowest-order solution  $\lambda$  to the above equation is plotted in Fig. 2.3 versus the dielectric constant of the gate insulator [4]. The curve is calculated for a constant  $\epsilon_i/t_i$ , that is, for the same  $t_{ox}^{eq} \equiv (\epsilon_{ox}/\epsilon_i)t_i$ . When  $\epsilon_i < \epsilon_{si}$  and  $t_i \ll W_{dm}$ , an approximate solution is  $\lambda \approx W_{dm} + (\epsilon_{si}/\epsilon_i)t_i$ , dominated by  $t_{si}$ . When  $\epsilon_i \gg \epsilon_{si}$ , however,  $\lambda$  starts to increase for a given  $t_{ox}^{eq}$ , indicating that the lateral field is becoming important. In the extreme limit,  $\lambda \approx t_i + (\epsilon_i/\epsilon_{si})W_{dm}$ , dominated by the physical thickness of the gate insulator, independent of its dielectric constant [5]. This shows that even for ultra high- $k$  gate insulators, the physical film thickness must be kept less than half the channel length to be useful.

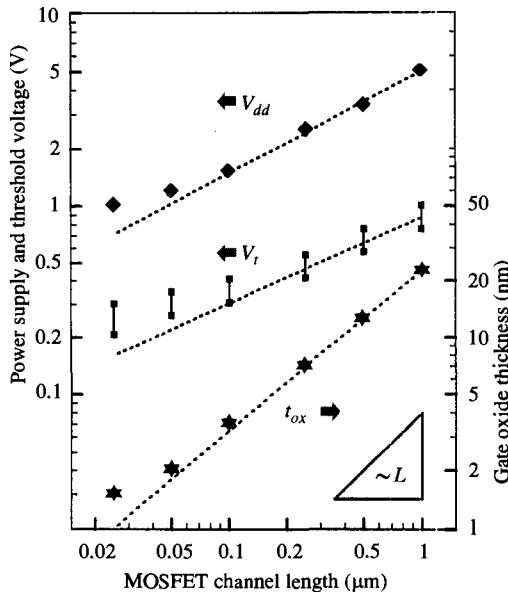
## 2.2 CMOS SCALING ISSUES BELOW 0.25 μm

Even though constant-field scaling provides a basic guideline to the design of scaled MOSFETs, the requirement of reducing the voltage by the same factor as the device physical dimension is too restrictive. Because of subthreshold (diffusion) current nonscaling and reluctance to depart from the standardized voltage levels of the previous generation, the power supply voltage was seldom scaled in proportion to channel length. In fact, the field has been gradually rising over the generations rather than staying constant.

### 2.2.1 Power Supply and Threshold Voltage

This trend can be clearly seen in Fig. 2.4 where power supply voltage ( $V_{dd}$ ), threshold voltage ( $V_t$ ), and gate oxide thickness ( $t_{ox}$ ) are plotted as a function of MOSFET channel length [5]. A high electric field can lead to a number of deleterious effects, such as hot-carrier injection into gate oxide and electromigration resulting from the increased current density, that could impact chip reliability. More importantly, higher than scaled voltage level drives up the active power of a CMOS chip, which is given by (excluding crossover currents)

$$P_{ac} = C_{sw} V_{dd}^2 f \quad (2.6)$$



**Figure 2.4** History and trends of power supply voltage ( $V_{dd}$ ), threshold voltage ( $V_t$ ), and gate oxide thickness ( $t_{ox}$ ) versus channel length for CMOS logic technologies.

where  $C_{sw}$  is the total node capacitance being charged and discharged in a clock cycle, and  $f$  is the clock frequency. As CMOS technology advances, clock frequency goes up. The total switching capacitance is likely to increase as well, as one tries to integrate more circuits into the same or even larger chip area. The active power of today's microprocessors is already in the 10–50 W range. Besides power management systems with architectural innovation, the most effective way to curb the growth of active power is to reduce the power supply voltage.

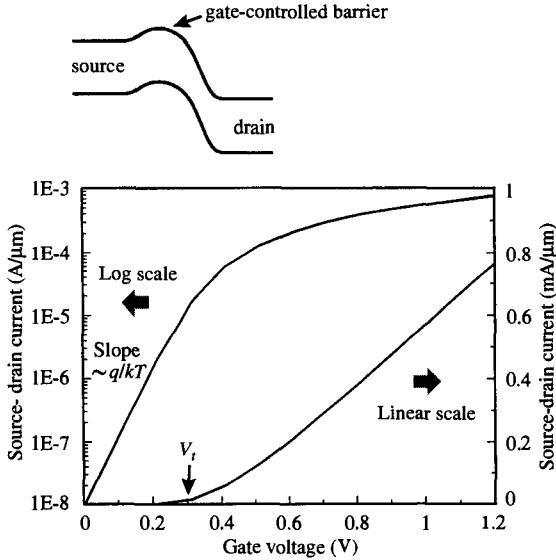
### 2.2.1.1 Threshold Voltage Nonscaling

Threshold voltage deviates even further away from the ideal scaling behavior than the power supply voltage, as seen in Fig. 2.4. MOSFET threshold voltage is defined as the gate voltage at which significant current starts to flow from the source to the drain (Fig. 2.5). Below the threshold voltage, the current does not drop immediately to zero. Rather, it decreases exponentially with a slope on the logarithmic scale inversely proportional to the thermal energy  $kT$ . This is because some of the thermally distributed electrons at the source of the transistor have high enough energy to overcome the potential barrier controlled by the gate voltage and flow to the drain (Fig. 2.5 inset). Such a subthreshold behavior follows directly from fundamental thermodynamics and is independent of power supply voltage and channel length.

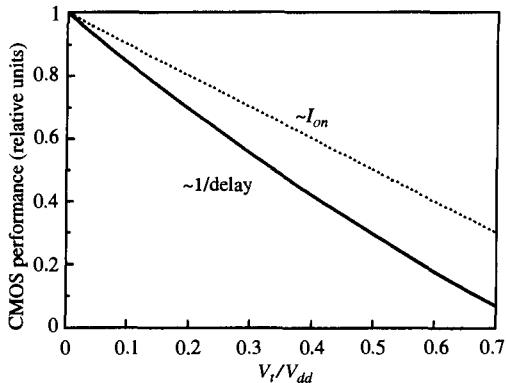
The standby power of a CMOS chip is given by [1]

$$P_{off} = W_{tot} V_{dd} I_{off} = W_{tot} V_{dd} I_0 \exp\left(-\frac{qV_t}{mkT}\right) \quad (2.7)$$

where  $W_{tot}$  is the total width of turned-off devices and  $I_{off}$  is the off current per width at 100°C (worst-case temperature).  $I_{off}$  depends exponentially on the threshold voltage at 100°C,  $V_t$ , where  $m$  is a dimensionless body-effect coefficient typically  $\approx 1.2$ . Even if  $V_t$  is kept constant, the leakage current of turned-off devices would increase in proportion



**Figure 2.5** MOSFET current in both logarithmic (left) and linear (right) scales versus gate voltage. Inset shows the band diagram of an nMOSFET.



**Figure 2.6** Relative CMOS performance ( $\propto 1/\text{delay}$ ) as a function of  $V_t/V_{dd}$ . The dependence can be fitted by a function,  $0.7 - V_t/V_{dd}$ , which is stronger than that of the on current ( $\propto 1 - V_t/V_{dd}$ ).

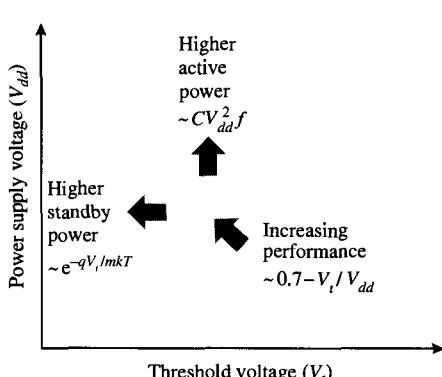
to  $1/t_{ox}$  and  $(W_{tot}/L)$  because the current at threshold condition  $I_0$  (of the order of 1–10 μA/μm for 0.1 μm devices) is proportional to the inversion charge density at threshold,  $Q_i \sim (1-2)(kT/q)C_{ox}$ , where  $C_{ox} = \epsilon_{ox}/t_{ox}$  is the gate oxide capacitance per unit area. The off-state leakage current would further increase by about 10× for every 0.1 V reduction of  $V_t$ . Fortunately, the leakage current and therefore the standby power is quite low in today's CMOS chips. This allows some room for a slightly downward trend of  $V_t$  as shown in Fig. 2.4. For a chip with an integration level of 100 million transistors, the average leakage current of turned-off devices should not exceed a few times  $10^{-8}$  A. This constraint bounds the threshold voltage to a minimum of about  $\sim 0.2$  V at the operating temperature (100°C worst case).

### 2.2.1.2 Power vs. Performance Trade-Off

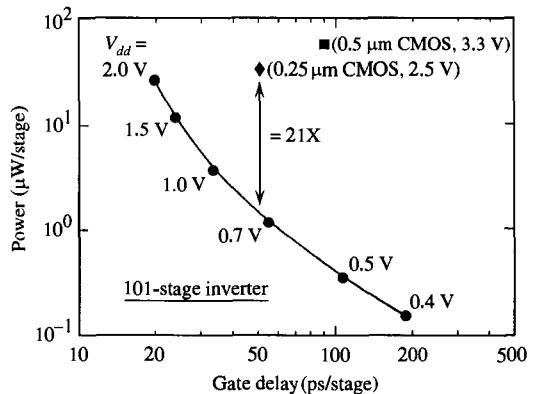
The increasing  $V_t/V_{dd}$  ratio in Fig. 2.4 means a loss of gate overdrive ( $V_{dd} - V_t$ ) which degrades CMOS circuit performance gained from scaling. Figure 2.6 plots the

relative CMOS performance ( $\propto 1/\text{delay}$ ) as a function of  $V_t/V_{dd}$ . The dependence can be fitted by a function,  $0.7 - V_t/V_{dd}$ , which is stronger than that of the on current ( $\propto 1 - V_t/V_{dd}$ ) because of the finite rise time of the driving stage [1]. Since  $V_t$  cannot be reduced below 0.2 V and since CMOS performance is a sensitive function of  $V_t/V_{dd}$  ratio, there is very little performance to gain by scaling the power supply voltage to significantly below 1 V.

The above considerations can be represented in general as trade-offs between power and performance in a power supply voltage-threshold voltage design plane (Fig. 2.7). Performance is gained at the expense of higher active power or higher standby power or both. For high-performance CMOS, power supply voltage will be decreased more slowly than the historical trend (Fig. 2.4), leading to higher chip power and more aggressive device designs at higher electric fields [6]. For low power applications, like in handheld wireless units, threshold voltage will not be reduced as aggressively and the supply voltage can be further scaled down with significant savings in power-delay products. This is demonstrated in Fig. 2.8 where the power per device of a 0.1  $\mu\text{m}$  CMOS ring oscillator is plotted versus gate delay by varying the power supply voltage [7]. Near the high-performance design point discussed above, a 10% decrease in performance usually brings in 30–40% saving in active power.



**Figure 2.7** Relative trends of CMOS performance, active power, and standby power in a  $V_{dd} - V_t$  design plane.



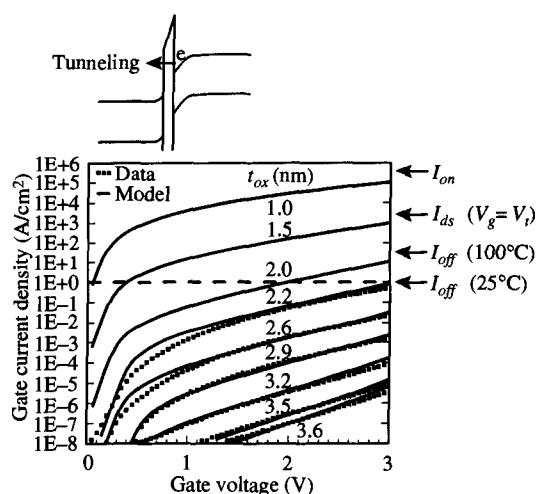
**Figure 2.8** Measured power per stage versus gate delay of 0.1  $\mu\text{m}$  CMOS ring oscillator with  $V_{dd}$  as a parameter. The device widths are  $W_n = 3 \mu\text{m}$ ,  $W_p = 4 \mu\text{m}$ .

There are other schemes for meeting leakage power requirements. For example, one can fabricate multiple-threshold-voltage devices on a chip. Low-threshold devices could be used in critical logic paths for speed, while high-threshold devices would be used everywhere else, including memory arrays, for low standby power. One can also sense the circuit activity and cut off the power supply to logic blocks that are not switching—an approach known as *sleep mode*. Other possibilities include dynamic-threshold devices for which the threshold voltage is controlled by a backgate bias voltage in either bulk or silicon-on-insulator device structures. Yet another option is low-temperature CMOS. Low-temperature operation not only steepens the subthreshold slope and improves mobility (Section 2.4), but also reduces wire resistance. However, all of these solutions will generally carry a cost in complexity and/or density.

## 2.2.2 Gate Oxides

To keep adverse 2-D electrostatic effects on threshold voltage (short-channel effect) under control, gate oxide thickness is reduced nearly in proportion to channel length as shown in Fig. 2.4. This is necessary in order for the gate to retain more control over the channel than the drain. A simple rule is that gate oxide thickness needs to be about 1/50 to 1/25 of channel length [1]. For sub-0.25 μm CMOS systems with channel lengths of 0.1 μm or shorter, an oxide thickness of < 3 nm is needed. This thickness consists of only a few layers of atoms and is approaching fundamental limits. While it is amazing that SiO<sub>2</sub> can carry us this far without being limited by extrinsic factors such as defect density, surface roughness, or large-scale thickness and uniformity control, oxide films this thin are subject to quantum mechanical tunneling, giving rise to a gate leakage current that increases exponentially as the oxide thickness is scaled down. Tunneling currents for oxide thicknesses ranging from 3.6 nm to 1.0 nm are plotted versus gate voltage in Fig. 2.9 [8]. In the direct tunneling regime, the current is rather insensitive to the applied voltage or field across the oxide, so reduced voltage operation will not offer much relief. Although the gate leakage current may be at a level negligible compared with the on-state current of a device, it will first have an effect on the chip standby power. Note that the leakage power will be dominated by turned-on nMOSFETs in which electrons tunnel from the silicon inversion layer to the positively biased gate (Fig. 2.9 inset). Edge tunneling in the gate-to-drain overlap region of turned-off devices should not be a fundamental issue as one can always build up the corner oxide thickness by additional oxidation of polysilicon after gate patterning. pMOSFETs have a much lower leakage than nMOSFETs because there are very few electrons in the p<sup>+</sup> poly gate available for tunneling to the substrate and hole tunneling has a much lower probability. If one assumes that the total active gate area per chip is of the order of 0.1 cm<sup>2</sup>, the maximum tolerable gate leakage current would be of the order of 1–10 A/cm<sup>2</sup>. This sets a lower limit of 1.5–2.0 nm for the physical thickness of gate oxide. Dynamic memory devices have a more stringent leakage requirement and therefore a thicker limit on gate oxide.

Another issue with the thin gate oxide is the loss of inversion charge and therefore current and transconductance due to inversion layer quantization [9] and polysilicon-gate



**Figure 2.9** Measured and calculated oxide tunneling currents versus gate voltage for different oxide thicknesses. The labels on the right indicate the various current magnitudes of a 0.1 μm channel length MOSFET. The inset shows the band diagram for tunneling in a turned-on nMOSFET.

depletion effects. In the quantum mechanical model, the density of inversion electrons peaks at approximately 1 nm below the silicon surface, which effectively reduces the gate capacitance and therefore the inversion charge to those of an equivalent oxide 0.3–0.4 nm thicker than the physical oxide. Similarly, depletion effects occur in polysilicon in the form of a thin space charge layer near the gate oxide interface which acts to reduce the gate capacitance and inversion charge density for a given gate drive [10]. The percentage of gate capacitance attenuation becomes more significant as the oxide thickness is scaled down. For a polysilicon doping of  $10^{20} \text{ cm}^{-3}$ , a 2 nm oxide loses about 20% of the inversion charge at a 1.5 V gate voltage due to the combined polysilicon gate depletion and inversion layer quantization effects [5].

### 2.2.3 Channel Profile Design

A schematic cross section of CMOS devices for sub-0.25 μm systems is shown in Fig. 2.10. To achieve low-threshold voltages for both nMOS and pMOS devices in a CMOS chip, dual n<sup>+</sup> and p<sup>+</sup> polysilicon gates are still required, in spite of their shortcomings in conductivity and depletion effects. This is because threshold voltages are dictated by the gate work functions. A midgap work function metal gate, while free of depletion effects, would either result in threshold voltage magnitudes too high for both devices or require compensating doping of the channel leading to buried channel operation with poor short-channel characteristics [1].

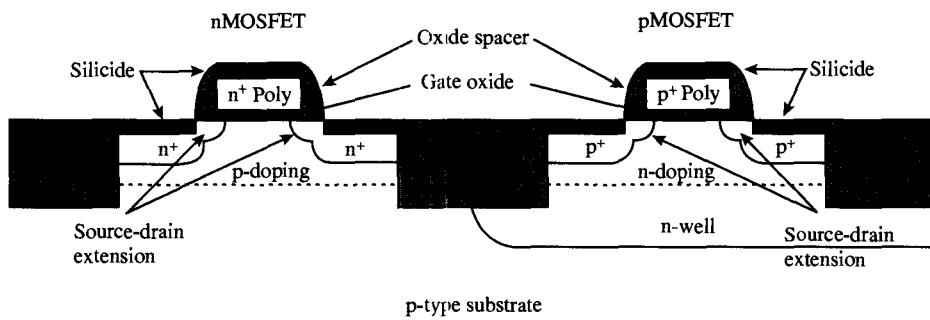


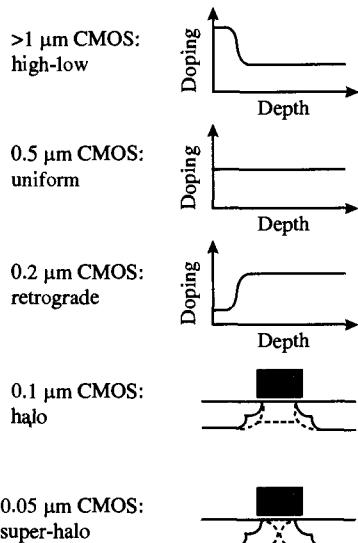
Figure 2.10 Schematic cross section of CMOS devices in sub-0.25 μm systems.

#### 2.2.3.1 Retrograde Doping

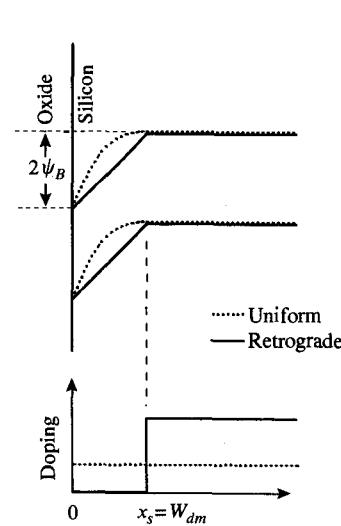
Various vertically nonuniform doping profiles, such as high-low and low-high, have been employed in previous CMOS generations to optimize both the gate depletion depth and the threshold magnitude simultaneously (top three schematics in Fig. 2.11). As discussed in Section 2.1.2, to scale down MOSFET channel length without excessive short-channel effect, both the oxide thickness and the gate-controlled depletion width in silicon ( $W_{dm}$ ) must be reduced in proportion to  $L$ . The latter requires increased channel doping concentration, which, for a uniformly doped ( $N_a$ ) channel, leads to higher depletion charge and electric field at the silicon surface. These in turn cause the potential across the oxide and therefore the threshold voltage,

$$V_t = V_{fb} + 2\psi_B + \frac{\sqrt{4\epsilon_{si}qN_a\psi_B}}{C_{ox}} \quad (2.8)$$

to go up. Here  $V_{fb}$  is the flatband voltage,  $C_{ox}$  is the gate oxide capacitance per unit area, and  $\psi_B = (kT/q) \ln(N_a/n_i)$  is the Fermi potential with respect to the midgap. To reduce the gate-controlled depletion width while fulfilling the  $V_t$ -reduction trend depicted in Fig. 2.4, a retrograde, that is, low-high, channel doping is needed below 0.25 μm channel length [1]. Figure 2.12 shows a schematic band-bending diagram at the threshold condition of an extreme retrograde profile with an undoped surface layer of thickness  $x_s$ . For the same gate depletion width ( $W_{dm}$ ), the surface electric field and the total depletion charge of an extreme retrograde channel is one-half that of a uniformly doped channel. This reduces threshold voltage and improves mobility.

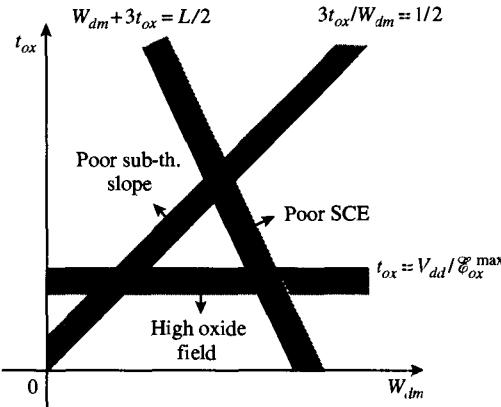


**Figure 2.11** Evolution of CMOS channel doping profiles.

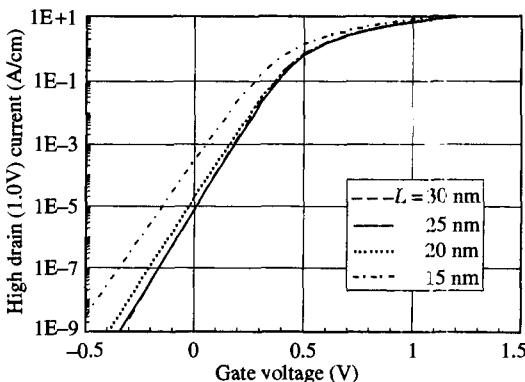


**Figure 2.12** Band diagrams at the threshold condition of a uniformly doped and an extreme retrograde-doped channel (profiles shown below).

Retrograde channel doping represents a vertically nonuniform profile that allows the threshold voltage to be decoupled from the gate-controlled depletion width. However, the body-effect coefficient,  $m = 1 + (\epsilon_{si}/W_{dm})/(\epsilon_{ox}/t_{ox}) \approx 1 + (3t_{ox}/W_{dm})$ , and the inverse subthreshold slope,  $(\ln 10)(mkT/q)$ , are still coupled to the gate depletion width  $W_{dm}$ . The sensitivity of threshold voltage to substrate bias, measured by  $m - 1$ , is usually referred to as the body effect. Body effect tends to degrade MOSFET currents when the source-to-substrate junction is reverse biased. For a given  $t_{ox}$ , reduction in  $W_{dm}$  improves short-channel effect, but compromises substrate sensitivity and subthreshold slope. Since both the subthreshold slope,  $2.3mkT/q$ , and the substrate sensitivity,  $dV_t/dV_{bs} = m - 1$ , degrade with higher  $m$ ,  $m$  should be kept close to 1. A larger  $m$  also results in a lower saturation current in the long-channel limit. Typically, one requires  $m < 1.5$ , or  $3t_{ox}/W_{dm} < 1/2$ . These design considerations are illustrated in Fig. 2.13 [1]. The lower limit of  $t_{ox}$  is imposed by technology constraints to  $V_{dd}/\epsilon_{ox}^{\max}$ , where  $\epsilon_{ox}^{\max}$  is the maximum oxide field. For a given  $L$  and  $V_{dd}$ , the allowable parameter space in a  $t_{ox}$ - $W_{dm}$  design plane is a triangular area bounded by SCE, oxide field, and subthreshold slope (also substrate sensitivity) requirements.



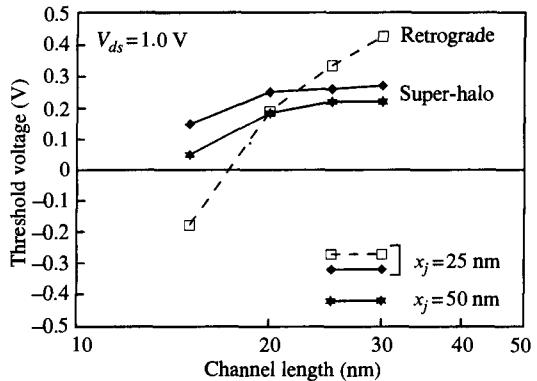
**Figure 2.13** A  $t_{ox}$ - $W_{dm}$  design plane. Some trade-off among the various factors can be made within the parameter space bounded by SCE, body effect, and oxide field considerations.



**Figure 2.14** Simulated subthreshold currents for channel lengths from 30 to 15 nm with a super-halo profile.  $I_{off} = 10^{-5}$  A/cm (1 nA/ $\mu$ m) for 20, 25, and 30 nm devices.

### 2.2.3.2 Halo Doping

Halo doping or nonuniform channel profile in the lateral direction was introduced below 0.25 μm to provide yet another degree of freedom which can be tailored to further minimize  $V_t$ -tolerances due to short-channel effect (Fig. 2.11). With the higher fields and nonscaled gate oxide below the 0.1 μm generation, an optimally tailored profile that is both vertically and laterally nonuniform (super-halo) is needed to contain the short-channel effect [11]. Such a profile can be realized by ion implantation self-aligned to the gate with a very restricted amount of diffusion. The highly nonuniform profile sets up a higher effective doping concentration toward shorter devices, which counteracts short-channel effects. This results in off currents insensitive to channel length variations and allows CMOS scaling to the shortest channel length possible. In a 25 nm CMOS design shown in Fig. 2.14,  $I_{off}$  is independent of channel length variations between 20 and 30 nm [11]. The superior short-channel effect obtained with the super-halo is shown in Fig. 2.15, compared with a non-halo retrograde profile. Because of the flat  $V_t$  dependence on channel length, super-halo allows a nominal device to operate at a lower threshold voltage, thereby gaining significant performance benefit: 30–40% over non-halo devices for a 1.0 V power supply. The halo doping level is ultimately limited by band-to-band tunneling currents in the high-field region near the drain [11].



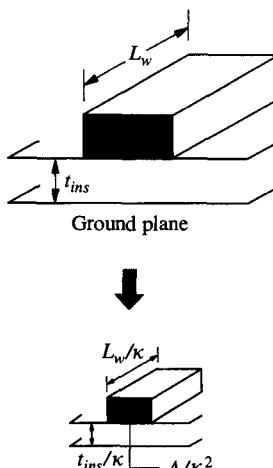
**Figure 2.15** Short-channel threshold roll-off for super-halo and retrograde (non-halo) doping profiles. Threshold voltage is defined as the gate voltage where  $I_{ds} = 1 \mu\text{A}/\mu\text{m}$ .

## 2.3 INTERCONNECT *RC* DELAY

Interconnect capacitance and resistance have negligible effects on the delay of local circuits such as CMOS inverters or NAND gates. However, they play a major role on VLSI system performance, especially in standard-cell designs where wire capacitance dominates circuit delay. While interconnect *RC* issues cannot be separated from circuit and system design methodology, this section summarizes the fundamental principles of wire delay from a technology perspective.

### 2.3.1 Interconnect Scaling

A straightforward strategy on interconnect scaling similar to that of MOSFET scaling is shown schematically in Fig. 2.16 [2]. All linear dimensions—wire length, width, thickness, spacing, and insulator thickness—are scaled down by the same factor,  $\kappa$ , as the device scaling factor. Wire lengths ( $L_w$ ) are reduced by  $\kappa$  since the linear dimension of the devices and circuits that they connect to is reduced by  $\kappa$ . Both the wire and the insulator thicknesses are scaled down along with the lateral dimension for otherwise the fringe capacitance and wire-to-wire coupling (crosstalk) would increase



**Figure 2.16** Scaling of interconnect lines and insulator thicknesses.

**TABLE 2.2** Scaling of Local Interconnect Parameters

	Interconnect parameters	Scaling factor ( $\kappa > 1$ )
Scaling assumptions	Interconnect dimensions ( $t_w, L_w, W_w, t_{ins}, W_{sp}$ )	$1/\kappa$
	Resistivity of conductor ( $\rho_w$ )	1
	Insulator permittivity ( $\epsilon_{ins}$ )	1
Derived wire scaling behavior	Wire capacitance per unit length ( $C_w$ )	1
	Wire resistance per unit length ( $R_w$ )	$\kappa^2$
	Wire $RC$ delay ( $\tau_w$ )	1
	Wire current density ( $J/W_w t_w$ )	$\kappa$

disproportionately. Table 2.2 summarizes the rules for interconnect scaling. All material parameters, such as metal resistivity  $\rho_w$  and dielectric constant  $\epsilon_{ins}$ , are assumed to remain the same. Wire capacitance then scales down by  $\kappa$  the same way as the device capacitance (Table 2.1), while wire capacitance per unit length,  $C_w$ , remains unchanged (approximately 2 pF/cm for silicon dioxide insulation). Wire resistance, on the other hand, scales up by  $\kappa$ , in contrast to the device resistance which does not change with scaling (Table 2.1). Wire resistance per unit length,  $R_w$ , then scales up by  $\kappa^2$ , as indicated in Table 2.2. It is also noted that the current density of interconnects increases with  $\kappa$ , which implies that reliability issues such as electromigration may become more serious as the wire dimension is scaled down. Fortunately, a few material and process advances in metallurgy have taken place over the generations to keep electromigration in check in VLSI technologies.

### 2.3.1.1 Interconnect Resistance

Interconnect  $RC$  delay ( $\tau_w$ ) of an interconnect line with resistance per unit length  $R_w$ , capacitance per unit length  $C_w$ , and length  $L_w$  is given by [1]

$$\tau_w = \frac{1}{2} R_w C_w L_w^2 \quad (2.9)$$

Using  $R_w = \rho_w / W_w t_w$  and  $C_w \approx 2\pi\epsilon_{ins}$  (approximated by concentric cylinders with a radii ratio of  $\approx e$ ), one can express Eq. (2.9) as

$$\tau_w \approx \tau \epsilon_{ins} \rho_w \frac{L_w^2}{W_w t_w} \quad (2.10)$$

where  $W_w$  and  $t_w$  are the wire width and thickness, respectively. One of the key conclusions of interconnect scaling is that the wire  $RC$  delay  $\tau_w$  does not change as the device dimension and intrinsic delay are scaled down. Eventually, this will impose a limit on VLSI performance. Fortunately, for conventional aluminum metallurgy with silicon dioxide insulation,  $\rho_w \approx 3 \times 10^{-6} \Omega\text{-cm}$  and

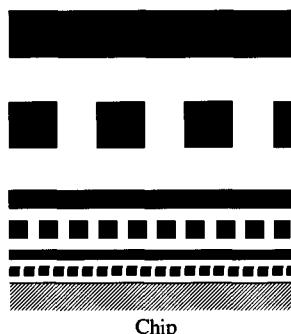
$$\tau_w \approx (3 \times 10^{-18} \text{ s}) \frac{L_w^2}{W_w t_w} \quad (2.11)$$

It is easy to see that the  $RC$  delay of local wires is negligible as long as  $L_w^2 / W_w t_w < 3 \times 10^5$ . For example, a  $0.25 \mu\text{m} \times 0.25 \mu\text{m}$  size wire  $100 \mu\text{m}$  long has an  $RC$  delay of  $0.5 \text{ ps}$ , which is quite negligible even when compared with the intrinsic delay ( $\approx 20 \text{ ps}$ ) of a  $0.1 \mu\text{m}$  CMOS inverter [5]. Therefore, a local circuit macro can be scaled down with all  $W_w$ ,  $t_w$ , and  $L_w$  reduced by the same factor without running into serious  $RC$  problems.

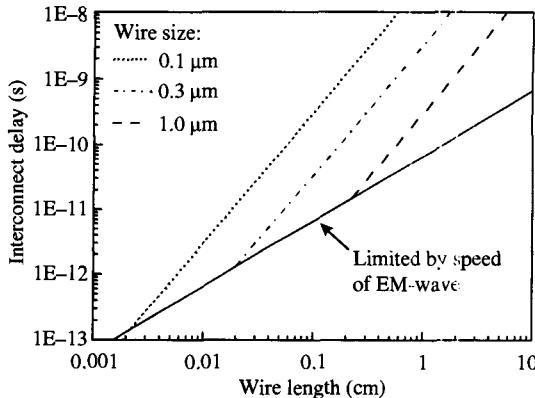
### 2.3.2 Global Wire Issues

Based on the above discussion,  $RC$  delay of local wires will not limit circuit speed even though it cannot be reduced through scaling. The  $RC$  delay of global wires, on the other hand, is an entirely different matter. Unlike local wires, the length of global wires, on the order of the chip dimension, does not scale down because chip size actually increases slightly for advanced technologies with better yield/defect density to accommodate a much higher number of circuit counts. Even if we assume chip size does not change, the  $RC$  delay of global wires scales up by  $\kappa^2$  from Eq. (2.11). It is clear that one quickly runs into trouble if the cross-sectional area of global wires is scaled down the same way as the local wires. For example, in a  $0.5\text{ }\mu\text{m}$  CMOS technology,  $L_w^2/W_w t_w$  is in the range  $10^8\text{--}10^9$ , and  $\tau_w$  is approximately 1 ns, severely impacting system performance. The use of copper wires instead of aluminum would reduce the numerical factor in Eq. (2.11) by a factor of about 1.5 and provide some relief. A number of solutions have been proposed to deal with the problem. The most obvious one is to minimize the number of cross-chip global interconnects in the critical paths as much as possible through custom layout/design and use of sophisticated design tools. One can also use repeaters to reduce the dependence of  $RC$  delay on wire length from a quadratic one to a linear one [12]. A more fundamental solution is to increase or not to scale the cross-sectional area of global wires. However, just increasing the width and thickness of global wires is not enough since wire capacitance will then increase significantly, which degrades both performance and power. Intermetal dielectric thickness must be increased in proportion to keep the wire capacitance per unit length constant. Of course, there is a technology price to pay to build such low- $RC$  global wires. It also means more levels of interconnects since one still needs several levels of thin, dense local wires to make the chip “wirable.”

The best strategy for interconnect scaling is then to scale down the size and spacing of lower levels in step with device scaling for local wiring, and to use unscaled or even scaled-up levels on top for global wiring, as shown schematically in Fig. 2.17 [13]. Unscaled wires allow the global  $RC$  delay to remain essentially unchanged as seen from Eq. (2.11). Scaled-up (together with the insulator thickness) wires allow the global  $RC$  delay to scale down together with the device delay. This is even more necessary if chip size increases with every generation. Ultimately, the scaled-up global wires would approach the transmission-line limit when the inductive effect becomes more important than the resistive effect. This happens when the signal rise time is shorter than the time of flight over the length of the line. Signal propagation is then limited by the speed of electromagnetic wave,  $c/(\epsilon_{ins}/\epsilon_0)^{1/2}$ , instead of by  $RC$  delay. Here  $c = 3 \times 10^{10}\text{ cm/s}$  is



**Figure 2.17** Schematic cross section of wiring hierarchy that addresses both the density and the global  $RC$  delay in a high-performance CMOS processor.



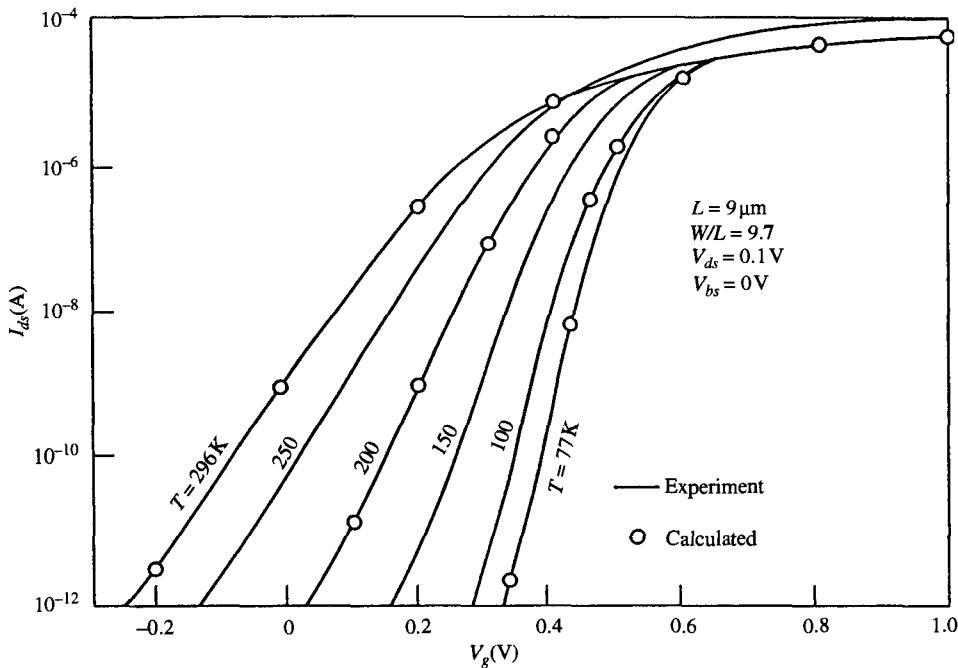
**Figure 2.18** RC delay versus wire length for three different wire sizes (assuming square wire cross sections). Wires become electro-magnetic-wave-propagation limited when the RC delay equals the time of flight over the line length. An oxide insulator is assumed here.

the velocity of light in vacuum. For oxide insulators,  $(\epsilon_{ins}/\epsilon_0)^{1/2} \approx 2$ , the time of flight is approximately 70 ps/cm. Figure 2.18 shows the interconnect delay versus wire length calculated from Eq. (2.11) for three different wire cross sections [5]. For a longer global wire to reach the speed-of-light limit, a larger wire cross section is needed. The transmission-line situation is more often encountered in packaging wires [12].

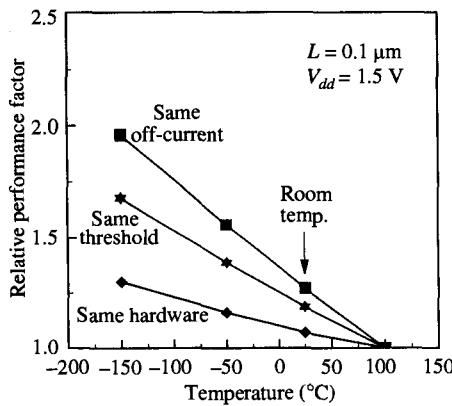
## 2.4 LOW-TEMPERATURE CMOS

In Sect. 2.2, we discussed why CMOS performance trend will slow below 0.1 μm channel length due to fundamental factors of oxide tunneling and voltage nonscaling [6]. One option that offers opportunities for further gains in performance-driven systems is cooled CMOS. The benefit is mainly derived from two aspects of MOSFET characteristics at low temperature: higher carrier mobilities and steeper subthreshold slope. Electron mobility improves by a factor of 2–5 from 300 K to 77 K, depending on the magnitude of the vertical field [14]. Similarly, hole mobility also improves by a factor of 1.7–4 for the same temperature range. In addition, MOSFET subthreshold slope steepens by a factor proportional to the absolute temperature, making it much easier to turn off a device at low temperature than at room temperature (Fig. 2.19) [15]. This allows the threshold voltage  $V_t$ , and therefore the power supply voltage  $V_{dd}$ , to scale down further below their permissible values at room temperature. It is important to also tighten the short-channel threshold-voltage tolerances through the use of optimized doping profiles, that is, super-halo.

To project the performance gain of CMOS circuits at low temperatures, we consider an example of 0.1 μm CMOS devices operated at 1.5 V. Figure 2.20 plots the relative performance factor, defined as inversely proportional to the two-way NAND delay, versus temperature [1]. Three different scenarios are considered, depending on the assumption of the threshold voltage. In each case, the performance factor is normalized to the value at 100°C, the worst-case temperature for most IC products. The performance gained from the higher mobilities is represented by the *same threshold* (middle) curve, for which the threshold voltages are held constant as the temperature is varied. The improvement factor is less than that of the mobilities because of the velocity saturation effect in short-channel devices. In reality, however, for the *same hardware*, the magnitude of threshold voltage increases toward lower temperatures, as shown in Fig. 2.19. This is because the Fermi levels move toward the band edges as the



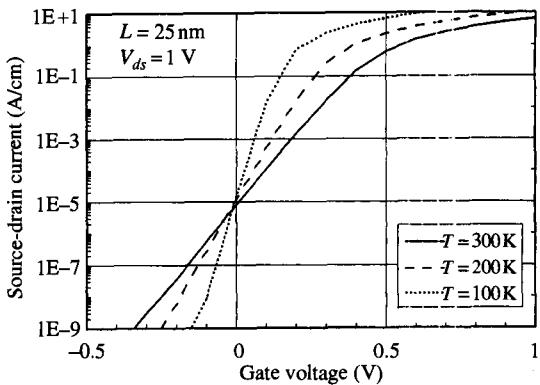
**Figure 2.19** Subthreshold  $I$ - $V$  characteristics of an nMOSFET as a function of temperature. The gate oxide is  $200\text{ \AA}$  thick.



**Figure 2.20** Relative performance factors (with respect to the  $100^\circ\text{C}$  value) of  $1.5\text{ V}$  CMOS circuits as a function of temperature. Threshold voltages are adjusted differently with temperature in each of the three scenarios shown.

temperature is lowered [1]. While it dramatically (and needlessly) reduces the off-current at zero gate voltage (Fig. 2.19), the rise of threshold voltage offsets a major part of the performance gained from the higher mobilities such that only a slight net improvement is obtained at low temperature, as shown by the bottom curve in Fig. 2.20.

To gain the most performance out of low-temperature CMOS, one should turn the threshold voltage trend around and take advantage of the steeper subthreshold slope. This is represented by the *same off-current* case in Fig. 2.20, in which the threshold voltages are adjusted to lower values as temperature decreases such that the off-current is maintained at the same level as the product specification at  $100^\circ\text{C}$  (Fig. 2.21). In



**Figure 2.21** Simulated subthreshold currents of 25 nm MOSFETs at three different temperatures.  $V_t$  is adjusted to maintain the same  $I_{off}$ .

principle, this can be accomplished using a retrograde channel profile without degrading the short-channel effect (Fig. 2.12). For a given depletion width in silicon, an ideal low-high step doping profile reduces the surface field, and therefore the potential drop across the gate oxide, by a factor of 2 from the uniformly doped case [1]. This makes it possible to achieve a  $V_t$  of  $\approx 0.2\text{ V}$  at 100 K (Fig. 2.21). Beyond that, a forward bias on the substrate would be necessary to further reduce the threshold voltage, assuming the work functions of  $n^+$  and  $p^+$  polysilicon gates. But there is a limit on the forward bias because a proper barrier must be maintained to prevent high leakage currents. It would be difficult to design for  $V_t < 0.1\text{ V}$  at very low temperatures. Given the poorer thermal efficiency and the steep rise in cooling system costs beyond  $-50^\circ\text{C}$ , this temperature is probably the sweet spot for cooled CMOS, where 50% higher performance can be achieved using optimized devices. With the ample gate overdrive for  $V_t = \pm 0.2\text{ V}$  and  $V_{dd} = 1.5\text{ V}$ , it may be desirable at this point to trade performance for lower power [4] by operating the CMOS devices at a lower supply voltage, for example, at 0.8 V. A  $4\times$  power reduction can be achieved with only a slight loss in performance compared with the 1.5 V case.

While the performance factors in Fig. 2.20 are derived at the device level, they are expected to translate directly to the chip level without extensive design modifications. This is because the delay improvement comes from the device currents and is therefore independent of loading conditions (as opposed to SOI which mainly benefits from lower parasitic capacitances). Another key factor is that the conductivity of metal interconnects and therefore  $RC$  delays also improve by at least as much as the devices at low temperatures.

## REFERENCES

- [1] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*. Cambridge University Press, New York, 1998.
- [2] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions," *IEEE J. Solid-State Circuits*, SC-9, 256 (1974).
- [3] T. N. Nguyen, "Small-Geometry MOS Transistors: Physics and Modeling of Surface- and Buried-Channel MOSFETs," *Ph.D. Thesis*, Stanford University, 1984.

- [4] D. J. Frank, Y. Taur, and H. -S. Wong, "Generalized Scale Length for Two-Dimensional Effects in MOSFET's," *IEEE Electron Device Lett.*, vol. 19, p. 385 (1998).
- [5] Y. Taur et al., "CMOS Scaling into the Nanometer Regime," *IEEE Proceedings*, p. 486 (1997).
- [6] Y. Taur and E. J. Nowak, "CMOS Devices below 0.1  $\mu\text{m}$ : How High Will Performance Go?" *1997 IEDM Technical Digest*, p. 215.
- [7] Y. Mii, S. Wind, Y. Taur, Y. Lii, D. Klaus, and J. Buccignano, "An Ultra-Low Power 0.1  $\mu\text{m}$  CMOS," *1994 VLSI Technology Symp. Technical Digest*, pp. 9–10.
- [8] S. -H. Lo, D. A. Buchanan, Y. Taur, and W. Wang, "Quantum-Mechanical Modeling of Electron Tunneling Current from the Inversion Layer of Ultra-Thin-Oxide nMOSFETs," *IEEE Electron Device Lett.*, vol. 18, pp. 209–211 (1997).
- [9] F. Stern, "Self-Consistent Results for *n*-Type Si Inversion Layers," *Physical Review B*, vol. 5, no. 12, pp. 4891–4899, 1972.
- [10] C. Y. Wong, J. Y. -C. Sun, Y. Taur, C. S. Oh, R. Angelucci, and B. Davari, "Doping of  $n^+$  and  $p^+$  Polysilicon in a Dual-Gate CMOS Process," *1988 IEDM Technical Digest*, pp. 238–241.
- [11] Y. Taur, C. H. Wann, and D. J. Frank, "25 nm CMOS Design Considerations," *1998 IEDM Technical Digest*, p. 789.
- [12] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, Reading, MA, 1990.
- [13] G. A. Sai-Halasz, "Performance Trends in High-End Processors," *IEEE Proc.*, vol. 83, pp. 20–36 (1995).
- [14] S. Takagi, M. Iwase, and A. Toriumi, "On Universality of Inversion-Layer Mobility in *n*- and *p*-Channel MOSFETs," *1988 IEDM Technical Digest*, pp. 398–401.
- [15] F. H. Gaenslen, V. L. Rideout, E. J. Walker, and J. J. Walker, "Very Small MOSFETs for Low Temperature Operation," *IEEE Trans. Electron Devices*, ED-24, p. 218 (1977).

Vivek De, Yibin Ye, Ali Keshavarzi, Siva Narendra, James Kao,  
Dinesh Somasekhar, Raj Nair, Shekhar Borkar  
*Intel Corporation*

### 3.1 INTRODUCTION

Supply voltage ( $V_{cc}$ ) must continue to scale down at the historical rate of 30% per technology generation in order to keep power dissipation and power delivery costs under control in future high-performance microprocessor designs. To improve transistor and circuit performance by at least 30% per technology generation, transistor threshold voltage ( $V_t$ ) must also reduce at the same rate so that a sufficiently large gate overdrive ( $V_{cc}/V_t$ ) is maintained. However, reduction in  $V_t$  causes transistor subthreshold leakage current ( $I_{off}$ ) to increase exponentially. Large leakage can (1) severely degrade noise immunity of dynamic logic circuits, (2) compromise stability of 6T SRAM cells, and (3) increase leakage power consumption of the chip to an unacceptably large value. In addition, degradation of short-channel effects, such as  $V_t$  roll-off and *drain-induced barrier lowering* (DIBL), in conventional bulk MOSFETs with low  $V_t$  can pose serious obstacles to producing high-performance, manufacturable transistors at low cost in sub-100 nm  $L_{eff}$  technology generations and beyond. To further compound the technology scaling problems, within-die and across-wafer device parameter variations are becoming increasingly untenable. This nonscalability of process tolerances is also a barrier to  $V_{cc}$  and technology scaling.

To illustrate the barrier associated with excessive leakage power, one can estimate the subthreshold leakage power of future chips, starting with the 0.25  $\mu\text{m}$  technology described in [1], and projecting subthreshold leakage currents for 0.18  $\mu\text{m}$ , 0.13  $\mu\text{m}$ , and 0.1  $\mu\text{m}$  technologies. Because subthreshold leakage is increasingly the dominant component of transistor leakage, it can be used to illustrate the excessive leakage power barrier. Assume that 0.25  $\mu\text{m}$  technology has  $V_t$  of 450 mV, and  $I_{off}$  is around 1 nA/ $\mu\text{m}$  at 30°C. Also assume that subthreshold slopes are 80 and 100 mV/decade at 30°C and 100°C, respectively,  $V_t$  scales by 15% per generation, and  $I_{off}$  increases by 5X each technology generation. Because  $I_{off}$  increases exponentially with temperature, it is important to consider leakage currents and leakage power as a function of temperature. Figure 3.1 shows projected  $I_{off}$  (as a function of temperature) for the four different technologies.

Next, we use these projected  $I_{off}$  values to estimate the active leakage power of a 15 mm die (small die), and compare the leakage power with the active power. The total transistor width on the die increases by  $\sim 50\%$  each technology generation; hence the

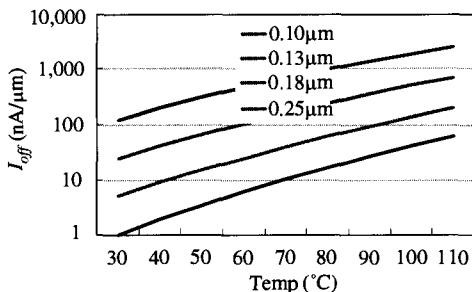


Figure 3.1 Projected off currents.

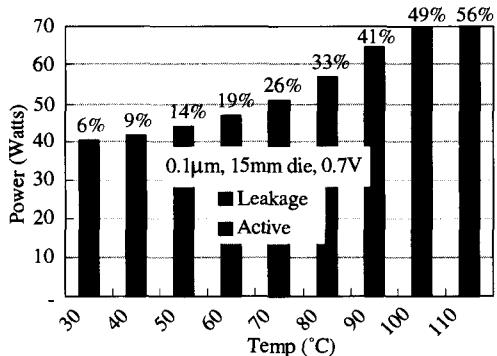


Figure 3.2 Projected leakage power in 0.1 μm technology.

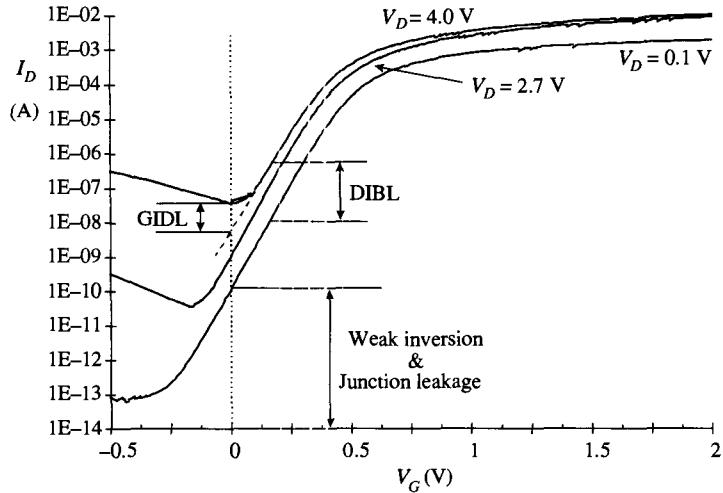
total leakage current increases by  $\sim 7.5X$ . This results in leakage power of the chip increasing by  $\sim 5X$  each technology generation. Since the active power remains constant (per scaling theory) for constant die size, the leakage power will become a significant portion of the total power as shown in Fig. 3.2.

This chapter explores the various components of leakage currents at the transistor level, and also describes the effect of leakage currents at the circuit level. Finally, it will conclude with a few techniques that can be used to help control subthreshold leakage currents in both sleep and active circuit modes.

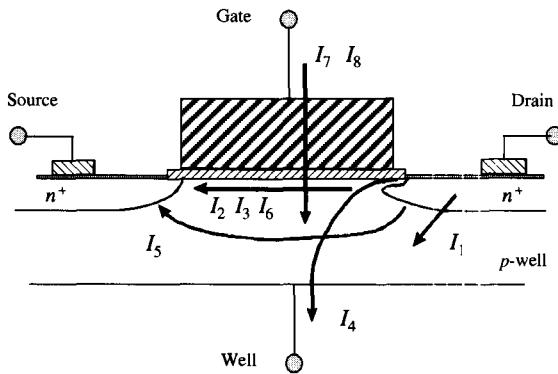
## 3.2 TRANSISTOR LEAKAGE CURRENT COMPONENTS

Transistor off-state leakage current ( $I_{OFF}$ ) is the drain current when the gate-to-source voltage is zero.  $I_{OFF}$  is influenced by threshold voltage, channel physical and effective dimensions, channel/surface doping profile, drain/source junction depth, gate oxide thickness,  $V_{DD}$ , and temperature. However,  $I_{OFF}$  as defined above is not the only leakage mechanism of importance for a deep submicron transistor. Log( $I_D$ ) versus  $V_G$  is an important transistor curve in the saturated and linear bias states (Fig. 3.3). It allows measurement of many device parameters such as  $I_{OFF}$ ,  $V_T$ ,  $I_D(\text{SAT})$ ,  $I_D(\text{LIN})$ ,  $g_m(\text{SAT})$ ,  $g_m(\text{LIN})$ , and slope ( $S_i$ ) of  $V_G$  versus  $I_D$  in the weak inversion state.  $I_{OFF}$  is measured at the  $V_G = 0$  V intercept. Measurements will illustrate all leakage current mechanisms and their properties in deep submicron transistors. The transistors in this study were from a  $0.35\text{ }\mu\text{m}$  CMOS process technology with  $L_{eff} \ll 0.25\text{ }\mu\text{m}$  and nominal  $V_{DD} \approx 2.5\text{ V}$  [2].

We describe eight short-channel leakage mechanisms illustrating certain properties with measurements (Fig. 3.4).  $I_1$  is reverse-bias p-n junction leakage caused by barrier emission combined with minority carrier diffusion and band-to-band tunneling away from the oxide-silicon interface,  $I_2$  is weak inversion,  $I_3$  is *drain-induced barrier lowering (DIBL)*,  $I_4$  is *gate-induced drain leakage (GIDL)*,  $I_5$  is channel punchthrough,  $I_6$  is channel surface current due to narrow width effect,  $I_7$  is oxide leakage, and  $I_8$  is gate current due to hot carrier injection. Currents  $I_1$ – $I_6$  are off-state leakage mechanisms, while  $I_7$  (oxide tunneling) occurs when the transistor is on.  $I_8$  can occur in the off state, but more typically occurs when the transistor bias states are in transition.



**Figure 3.3** n-channel  $I_D$  vs.  $V_G$  showing DIBL, GIDL, weak inversion, and p-n junction reverse bias leakage components in a  $0.35\text{ }\mu\text{m}$  technology.



**Figure 3.4** Summary of leakage current mechanisms of deep submicron transistors [10].

### 3.2.1 p-n Junction Reverse Bias Current ( $I_1$ )

The reverse bias p-n junction leakage ( $I_1$ ) has two main components: One is minority carrier diffusion/drift near the edge of the depletion region, and the other is due to electron-hole pair generation in the depletion region of the reverse bias junction [3]. If both n- and p- regions are heavily doped (this will be the case for advanced MOSFETs using heavily doped shallow junctions and halo doping for better short-channel effects), Zener and band-to-band tunneling may also be present. For a MOS transistor, additional leakage can occur between the drain and well junction from gated diode device action (overlap and vicinity of gate to the drain to well p-n junctions) or carrier generation in drain to well depletion regions with influences of the gate on these current components [4]. The p-n reverse bias leakage ( $I_{REV}$ ) is a function of junction area and doping concentration [3], [5].  $I_{REV}$  for pure diode structures in our technology [2] was a minimal contributor to total transistor  $I_{OFF}$ . The p-n junction breakdown voltage was  $> 8$  V.

### 3.2.2 Weak Inversion ( $I_2$ )

Weak inversion or subthreshold conduction current between source and drain in a MOS transistor occurs when gate voltage is below  $V_T$  [3], [6]. The weak inversion region is seen in Fig. 3.3 as the linear portion of the curve. The carriers move by diffusion along the surface similar to charge transport across the base of bipolar transistors. The exponential relation between driving voltage on the gate and the drain current is a straight line in a semi-log plot. Weak inversion typically dominates modern device off-state leakage due to the low  $V_T$  that is used.

### 3.2.3 DIBL (Drain-Induced Barrier Lowering) ( $I_3$ )

DIBL occurs when a high voltage is applied to the drain where the depletion region of the drain interacts with the source near the channel surface to lower the source potential barrier. The source then injects carriers into the channel surface without the gate playing a role. DIBL is enhanced at higher drain voltage and shorter  $L_{eff}$ . Surface DIBL typically happens before deep bulk punchthrough. Ideally, DIBL does not change the slope,  $S_t$ , but does lower  $V_T$ . Higher surface and channel doping and shallow source/drain junction depths reduce the DIBL leakage current mechanism [6], [7]. Figure 3.3 illustrates the DIBL effect as it moves the curve up and to the left as  $V_D$  increases. DIBL can be measured at constant  $V_G$  as the change in  $I_D$  for a change in  $V_D$ . For  $V_D$  between 0.1 and 2.7 V,  $I_D$  changed 1.68 decades giving a DIBL of 1.55 V/decade change of  $I_D$ . DIBL may also be quantified in units of mV/V for at a constant drain current value.

The subthreshold leakage of a MOS device including weak inversion and DIBL can be modeled according to the following equation.

$$I_{subth} = A \times e^{\frac{1}{nv_T}(V_G - V_S - V_{TH0} - \gamma' \times V_S + \eta \times V_{DS})} \times \left(1 - e^{-\frac{V_{DS}}{v_T}}\right) \quad (3.1)$$

where

$$A = \mu_0 C_{OX}' \frac{W}{L_{eff}} (v_T)^2 e^{1.8} e^{-\Delta V_{TH}}$$

$V_{TH0}$  is the zero bias threshold voltage,  $v_T = kT/q$  is the thermal voltage. The body effect for small values of source to bulk voltages is very nearly linear and is represented by the term  $\gamma' V_S$ , where  $\gamma'$  is the linearized body effect coefficient.  $\eta$  is the DIBL coefficient,  $C_{OX}'$  is the gate oxide capacitance,  $\mu_0$  is the zero bias mobility and  $n$  is the subthreshold swing coefficient for the transistor.  $\Delta V_{TH}$  is a term introduced to account for transistor-to-transistor leakage variations.

### 3.2.4 GIDL (Gate-Induced Drain Leakage) ( $I_4$ )

GIDL current arises in the high electric field under the gate/drain overlap region causing deep depletion [7] and effectively thins out the depletion width of drain to well p-n junction. The high electric field between gate and drain (a negative  $V_G$  and high positive  $V_D$  bias for NMOS transistor) generates carriers into the substrate and drain from direct band-to-band tunneling, trap-assisted tunneling, or a combination of thermal emission and tunneling. It is localized along the channel width between the gate and

drain. GIDL is at times referred to as surface band-to-band tunneling leakage. GIDL current is seen as the “hook” in the waveform of Fig. 3.3 that shows increasing current for negative values of  $V_G$  (gate bias dependent specially observed at high  $V_D$  curves). Thinner  $T_{ox}$  and higher  $V_{DD}$  (higher potential between gate and drain) enhance the electric field dependent GIDL. The impact of drain (and well) doping on GIDL is rather complicated. At low drain doping values, we do not have high electric field for tunneling to occur. For very high drain doping, the depletion volume for tunneling will be limited. Hence, GIDL is worse for drain doping values in between the above extremes. Very high and abrupt drain doping is preferred for minimizing GIDL as it provides lower series resistance required for high transistor drive current. GIDL is a major obstacle in  $I_{OFF}$  reduction. As it was discussed, a junction related bulk band-to-band tunneling component in  $I_1$  may also contribute to GIDL current, but this current will not be gate bias dependent. It will only increase baseline value of  $I_4$  current component.

We isolated  $I_{GIDL}$  by measuring source current  $\log(I_s)$  versus  $V_G$ . It is seen as the dotted line extension of the  $V_D = 4.0$  V curve in Fig. 3.3.  $I_{GIDL}$  is removed because it uses the drain and substrate (well) terminals, not the source terminal. The GIDL contribution to  $I_{OFF}$  is small at 2.7 V, but as the drain voltage rises to 4.0 V (close to burn-in voltage), the off-state current on the  $V_D = 4.0$  V curve increases from 6 nA (at the dotted line intersection with  $V_G = 0$  V) to 42 nA, for a GIDL of 36 nA. The pure weak inversion and reverse bias p-n junction current of 99 pA is approximated from the  $V_D = 0.1$  V curve.

### 3.2.5 Punchthrough ( $I_5$ )

Punchthrough occurs when the drain and source depletion regions approach each other and electrically “touch” deep in the channel. Punchthrough is a space-charge condition that allows channel current to exist deep in the subgate region causing the gate to lose control of the subgate channel region. Punchthrough current varies quadratically with drain voltage and  $S_t$  increases reflecting the increase in drain leakage [8, p. 134]. Punchthrough is regarded as a subsurface version of DIBL.

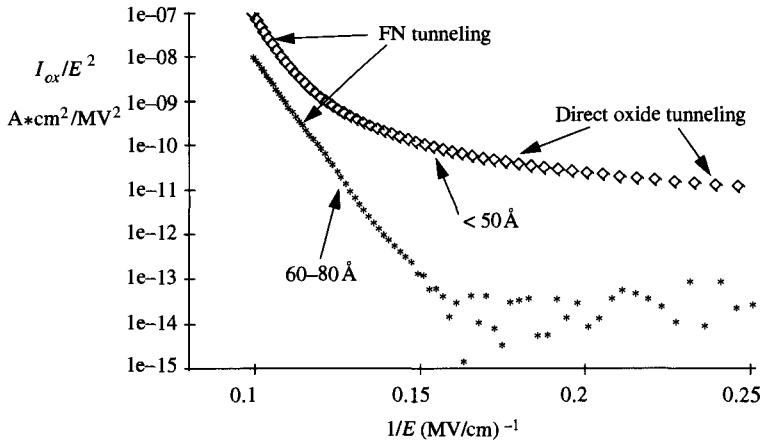
### 3.2.6 Narrow Width Effect ( $I_6$ )

Transistor  $V_T$  in nontrench isolated technologies increases for geometric gate widths on the order of  $\leq 0.5$   $\mu\text{m}$ . An opposite and more complex effect is seen for trench isolated technologies that show decrease in  $V_T$  for effective channel widths on the order of  $W \leq 0.5$   $\mu\text{m}$  [9]. No narrow width effect was observed in our transistor sizes with  $W \gg 0.5$   $\mu\text{m}$ .

### 3.2.7 Gate Oxide Tunneling ( $I_7$ )

Gate oxide tunneling current  $I_{ox}$  which is a function of electric field ( $E_{ox}$ ) can cause direct tunneling through the gate or Fowler–Nordheim (FN) tunneling through the oxide bands [Eq. (3.1)] [8]. FN tunneling typically lies at a higher field strength than found at product use conditions and will probably remain so. FN tunneling has a constant slope for  $E_{ox} > 6.5$  MV/cm (Fig. 3.5). Figure 3.5 shows significant direct oxide tunneling at lower  $E_{ox}$  for thin oxides.

$$I_{ox} = A \cdot E_{ox}^2 \cdot e^{-\frac{B}{E_{ox}}} \quad (3.2)$$



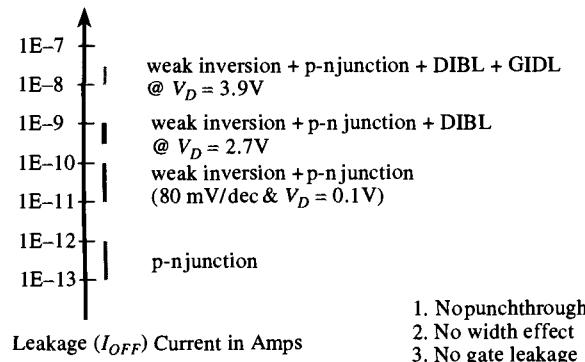
**Figure 3.5** Fowler–Nordheim and direct tunneling in *n*-channel transistor oxide. The 60–80 Å curve shows dominance of FN tunneling while the < 50 Å curve shows FN at high  $E_{ox}$ , but significant direct tunneling at low electric fields.

Oxide tunneling current is presently not an issue for devices in production, but could surpass weak inversion and DIBL as a dominant leakage mechanism in the future as oxides get thinner.

### 3.2.8 Hot Carrier Injection ( $I_8$ )

Short-channel transistors are more susceptible to injection of hot carriers (holes and electrons) into the oxide. These charges are a reliability risk and are measurable as gate and substrate currents. While past and present transistor technologies have controlled this component, it increases in amplitude as  $L_{eff}$  is reduced unless  $V_{DD}$  is scaled accordingly.

Figure 3.6 summarizes relative contributions of main components of intrinsic leakage for a typical 0.35 micron CMOS technology. We can see that for a nominal drain voltage of 2.7 V (consistent with typical power supply voltage of the technology),



**Figure 3.6** Components of  $I_{OFF}$  for a 0.35  $\mu$ m technology for a 20- $\mu$ m wide transistor. Currents from various leakage mechanisms accumulate resulting in a total measured transistor  $I_{OFF}$  for a given drain bias.

DIBL is the dominant component of leakage (it elevates the amount of weak inversion subthreshold leakage current). At elevated burn-in voltage of 3.9 V, GIDL dominates. However, at low  $V_D$ , weak inversion is the primary leakage mechanism.

### 3.3 CIRCUIT SUBTHRESHOLD LEAKAGE CURRENT

Subthreshold leakage currents for a single device can be modeled as illustrated in Eq. (3.1). However, in a CMOS circuit that contains multiple devices connected together, the net leakage effect will be highly dependent on the applied input vectors [11], [12]. The underlying mechanisms are related to (1) transistor stack effect and (2) total effective width of NMOS and PMOS devices that are turned off [13], [14].

#### 3.3.1 Transistor Stack Effect

The “stack effect” refers to the leakage reduction effect in a transistor stack when more than one transistor is turned off. The dynamics of the stack effect can be best understood by considering the two-input NAND gate in Fig. 3.7. When both M1 and M2 are turned off, the voltage  $V_M$  at the intermediate node is positive due to the small drain current. Thus, the gate-to-source voltage of the upper transistor M1 is negative, that is,  $V_{gs1} < 0$ . The exponential characteristic of the subthreshold conductance on  $V_{gs}$  greatly reduces the leakage. In addition, the body effect of M1 due to  $V_M > 0$  further reduces the leakage current as  $V_t$  increases.

The internal node voltage  $V_M$  is determined by the cross point of the drain currents in M1 and M2. Since leakage current strongly depends on the temperature and the transistor threshold voltage, we consider two cases: (1) High  $V_t$  and room temperature at 30°C and (2) low  $V_t$  and high temperature at 110°C. Figure 3.8 shows the DC solution of nMOS subthreshold current characteristics from SPICE simulations. The leakage current of a single nMOS transistor at  $V_g = 0$  is determined by the drain current of M1 at  $V_M = 0$ . It is clear that the leakage current through a two-transistor stack is approximately an order of magnitude smaller than the leakage of a single transistor. The voltage  $V_M$  of the internal node converges to  $\sim 100$  mV, as shown in Fig. 3.8. The small  $V_M$  ( $=$  drain-to-source voltage of M2) reduces the DIBL (drain-induced barrier lowering), and hence increases  $V_t$  of M2, which also contributes to the leakage reduction.

The subthreshold swing is proportional to  $kT$ . The slope decreases when the temperature  $T$  increases, which moves the cross point (Fig. 3.8) upwards. Thus, the amount of reduction will be smaller at higher temperature. The amount of reduction is also dependent of the threshold voltage  $V_t$ , which is larger for higher  $V_t$ . For three- or four-transistor stacks, the leakage reduction is found to be 2–3X larger in both nMOS

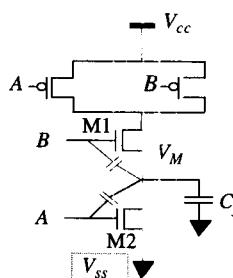


Figure 3.7 Two-nMOS stack in a two-input NAND gate.

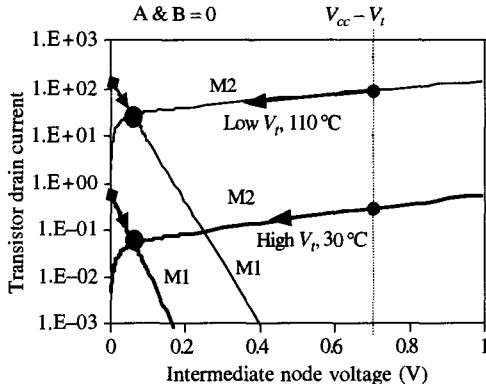


Figure 3.8 DC solution of 2-NMOS stack.

	High $V_t$	Low $V_t$
2 NMOS	10.7X	9.96X
3 NMOS	21.1X	18.8X
4 NMOS	31.5X	26.7X
2 PMOS	8.6X	7.9X
3 PMOS	16.1X	13.7X
4 PMOS	23.1X	18.7X

Figure 3.9 Leakage current reduction by two-, three-, and four-transistor stacks at room temperature  $T = 30^\circ\text{C}$ .

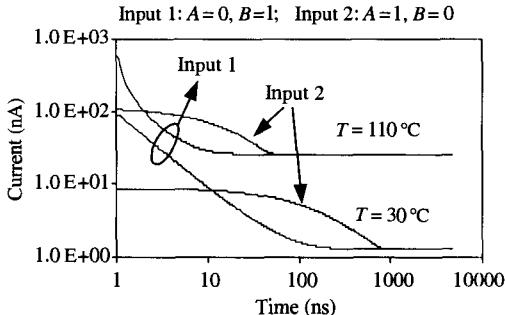
and pMOS. Results are summarized in Fig. 3.9. Note that reductions are obtained at the room temperature, as we are only interested in standby mode.

The reduced standby stack leakage current is obtained under steady-state condition. After a logic gate has a transition, the leakage current does not immediately converge to its steady-state value. Let us again consider the NAND gate in Fig. 3.7. Assume that the inputs switches from  $A = 0, B = 1$  to  $A = 0, B = 0$  to turn off both transistors, the voltage  $V_M$  of the internal node is approximately  $V_{DD} - V_t$  after the transition. Due to the junction capacitance  $C_j$ ,  $V_M$  will “slowly” go down as it is discharged by the subthreshold drain current of M2. In the other case, when the inputs switches from  $A = 1, B = 0$  to  $A = 0, B = 0$ ,  $V_M$  is negative after the transition due to the coupling capacitance between the gate and drain of M2 (as shown in Fig. 3.7). It also takes a certain amount of time for  $V_M$  to converge to its final value as determined by the DC stack solution.

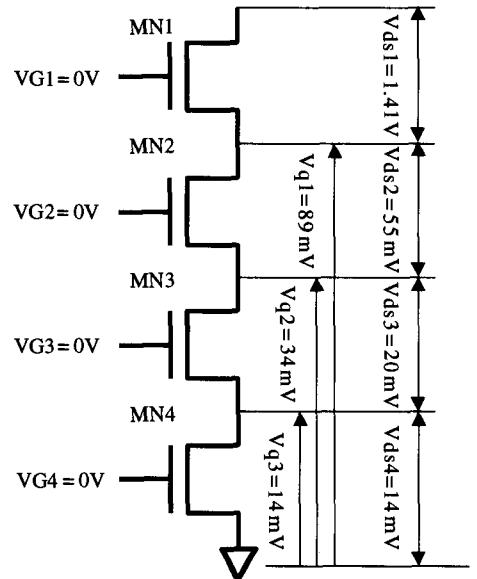
The time required for the leakage current in transistor stacks to converge to its final value is dictated by the rate of charging or discharging of the capacitance at the intermediate node by the subthreshold drain current of M1 or M2. The convergence of leakage current is shown in Fig. 3.10. We define the *time constant* as the amount of time required to converge to twice its final stack leakage value. This time constant is, therefore, determined by (1) drain-body junction and gate-overlap capacitances per unit width, (2) the input conditions immediately before the stack transistors are turned “off”, and (3) transistor subthreshold leakage current, which depends strongly on temperature and  $V_t$ . Therefore, the convergence rate of leakage current in transistor stacks increases rapidly with  $V_t$  reduction and temperature increase. For  $V_t = 200\text{ mV}$  devices in a sub-1V, 0.1  $\mu\text{m}$  technology, this time constant in 2-NMOS stacks at  $110^\circ\text{C}$  ranges from 5 to 50 ns.

### 3.3.2 Steady-State Leakage Model of Transistor Stacks

To investigate the leakage behavior of a stack of transistors, consider a stack of four NMOS devices. Such a structure would occur in the NMOS pull-down tree of a four-input NAND tree as shown in Fig. 3.11. Let us assume that all four devices are OFF with the applied gate voltage  $VG_1$  through  $VG_4$  being zero. Additionally the full



**Figure 3.10** Temporal behavior of leakage current in transistor stacks for different temperatures and initial input conditions.



**Figure 3.11** Four-input NAND NMOS stack.

supply voltage—1.5 V in the figure—is impressed across the stack. After a sufficiently long time, the voltage at each of the internal nodes will reach a steady-state value. With the assumption that subthreshold leakage from the drain to the source of the MOS device is the dominant leakage component and source drain junction leakage is negligible, application of Kirchoff's current law (KCL) yields the current through each transistor being the same and being identically equal to the overall stack current.

To calculate the overall leakage of the stack we use Eq. (3.1) to determine the leakage through a transistor as a function of the drain to source voltage. This yields the voltage across second transistor from the top as

$$V_{DS2} = \frac{nv_T}{(1 + 2\eta + \gamma')} \ln \left( \left( \frac{A_1}{A_2} \right) e^{\frac{nV_{DD}}{nv_T}} + 1 \right)$$

Additionally the voltage of the rest of the transistors can be expressed in a recursive fashion. The drain to source voltage of the  $i$ th transistor can be expressed in terms of the  $(i - 1)$ th transistor.

$$V_{DSi} = \frac{nv_T}{(1 + \gamma')} \ln \left( 1 + \frac{A_{i-1}}{A_i} \left( 1 - e^{\frac{-1}{v_T} V_{DS(i-1)}} \right) \right)$$

With the drain to source transistor voltages known, the leakage current through the stack can be computed by finding the leakage of the bottom transistor of the stack from the subthreshold equation. An identical method applies to the solution of leakage current for PMOS stacks. An estimate of leakage in transistor stacks was first presented by Gu and Elmasry [13]. The above analysis for transistor stack leakage was first presented by M. C. Johnson [14]. An early implementation of this idea for actively reducing leakage in word decoder-driver circuits for RAMs is presented by Kawahara

[15]. The term *self-reverse biasing* used in this paper, gives a clear indication of the mechanism by which leakage of a stack of transistors is reduced.

### 3.3.3 Transient Model of Transistor Stack Leakage

When stacked devices are turned off, the time required for the leakage currents to settle to the previously computed steady-state leakage levels can be large and can vary widely, ranging from microseconds to milliseconds. The settling time is important for determining if the quiescent leakage current model is applicable. The worst-case settling time for a stack occurs when all the internal nodes are charged to the maximum possible value  $V_{DD} - V_T$  just before every transistor of the stack is turned off. We notice that a strong reverse gate bias will now be present for all devices except for the bottom-most device. In the figure, MN1 to MN3 will have a reverse gate bias and the leakage through them is small. Hence, we approximate the discharge current of the drain node of MN4 as being the leakage current of MN4 alone. Once the drain voltage of MN4 is sufficiently small, MN3 discharges its drain node with a discharge current, which is the leakage current of the two stacked devices MN3 and MN4. The discharge time of each internal node of the stack— $t_{disi}$ —is sequential and the overall settling time is the sum of the discharge times. To derive a closed-form solution for the discharge time, it is assumed that the capacitance of the internal nodes is not dependent on voltage. Additionally it is assumed that we know the internal node voltages after the instant the devices are cut off—this requires a determination of the voltage to which internal nodes are bootstrapped. By using the capacitor discharge equation for the internal nodes, the discharge time can be written as

$$t_{disi} = \frac{nC_i L_{eff}}{\mu_0 C_{OX} W v_T e^{1.8\eta}} \times e^{\frac{1}{nv_T}} ((1 + \gamma' + \eta) V_{qi+1} + V_{TH0}) \times \left( e^{\frac{-\eta V_{qi}}{nv_T}} - e^{\frac{-\eta V_{booti}}{nv_T}} \right)$$

## 3.4 LEAKAGE CONTROL TECHNIQUES

Many techniques have been reported in the literature to reduce leakage power during standby condition. Examples of such techniques are: (1) reverse body biasing, as discussed in another chapter of this book; (2) MTCMOS sleep transistor and variations of sleep transistor-based techniques; and (3) embedded multiple- $V_t$  CMOS design where low- $V_t$  devices are used only in the critical paths for maximizing performance while high- $V_t$  devices are used in noncritical paths to minimize leakage power. In this section we will discuss two standby leakage reduction techniques: one through stack effect vector manipulation, and the other through embedded dual- $V_t$  design applied to domino circuits. We will also discuss the applicability of reverse body biasing for improving performance and leakage power distribution of multiple die samples during active modes, as well as the impact of technology scaling on the effectiveness of this technique.

### 3.4.1 Standby Leakage Control by Input Vector Activation

For any static CMOS gate other than the inverter, there are stacked transistors in NMOS or PMOS tree (e.g., PMOS stack in NOR, NMOS stack in NAND). Typically, a large circuit block contains a high percentage of logic gates where transistor “stacks” are already present. Note that leakage reduction in a transistor stack can be

achieved only when more than one device is turned off. Thus, the leakage current of a logic gate depends on its inputs. For a circuit block consisting of a large number of logic gates, the leakage current in standby mode is therefore determined by the vector at its inputs, which is fed from latches. For different vectors, the leakage is different. An input vector, which gives as small leakage current as possible, needs to be determined. One of the following three methods can be used to select the input vector: (1) Examining the circuit topology, makes it possible to find a “very good” input vector which maximizes the stack effect, and hence minimizes the leakage current. (2) An algorithm can be developed for efficiently searching for the “best” vector. (3) Testing or simulating a large number of randomly generated input vectors, the one with the smallest leakage can also be selected and used in the standby mode. Method 1 is suitable for datapath circuits (e.g., adders, multipliers, comparators) due to their regular structure. For random logic, an algorithm in method 2 is required to find a vector with good quality. In [11], the input dependence of the leakage has been empirically observed and random search is used to determine an input vector. However, the fact that the dependence is due to the transistor stack effect has not been addressed.

Figure 3.12 shows the distribution of the standby leakage current of a 32-bit static CMOS Kogg-Stone adder generated by 1000 random input vectors, with two threshold voltages. The standby leakage power varies by 30–40%, depending on the input vector, which determines the magnitude of the transistor stack effect in the design. The best input vector for minimum leakage can be easily determined by examining the circuit structure. This predetermined vector needs to be loaded into the circuit during the standby mode. Figure 3.13 shows an implementation of the new leakage reduction technique where a “standby” control signal, derived from the “clock gating” signal, is used to generate and store the predetermined vector in the static input latches of the adder during “standby” mode so as to maximize the stack effect (the number of nMOS and pMOS stacks with “more than one ‘off’ device”). Since the desired input vector for

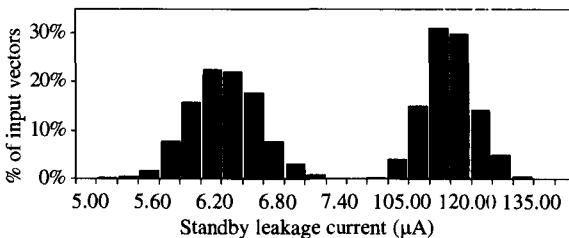


Figure 3.12 Distribution of standby leakage current in the 32-bit static CMOS adder for a large number of input vectors.

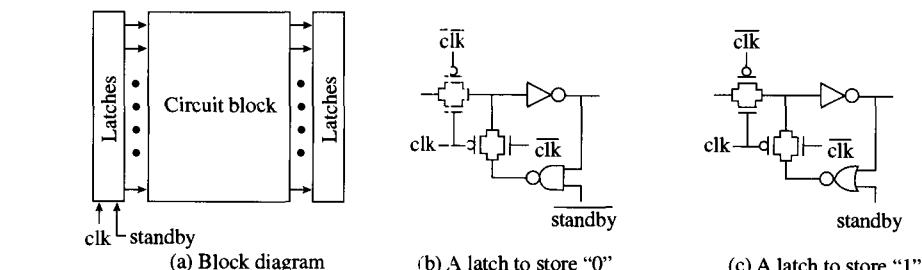


Figure 3.13 An implementation of the standby leakage control scheme through input activation.

		% Reduction
	Average	35.4%
High $V_t$	Worst	60.7%
	Average	33.3%
Low $V_t$	Worst	56.5%

**Figure 3.14** Adder leakage current reduction by the “best” input vector activation compared to average and worst standby leakage.

leakage minimization is encoded by using a NAND or NOR gate in the feedback loop of the static latch, minimal penalty is incurred in adder performance. As shown in Fig. 3.14, up to 2X reduction in standby leakage can be achieved by this technique. Note that the vector found by examining the design results in significantly smaller leakage than that obtained by any of the 1000 random vectors. In order that the additional switching energy dissipated by the adder and latches, during entry into and exit from “standby mode,” be less than 10% of the total leakage energy saved by this technique during standby, the adder must remain in standby mode for at least 5  $\mu$ s.

### 3.4.2 Embedded Dual- $V_t$ Design for Domino Circuits

A promising technique to control subthreshold leakage currents during standby modes, while still maintaining performance, is to utilize dual  $V_t$  devices. As previously described, two main dual  $V_t$  circuit styles are common in the literature. MTCMOS, or Multiple-Threshold CMOS, involves using high  $V_t$  sleep transistors to gate the power supplies for a low  $V_t$  block [16]. Leakage currents will thus be reduced during sleep modes, but the circuit will require large areas for the sleep transistors, and active performance will be affected. Furthermore, optimal sizing of the sleep transistors is complex for larger circuits, and will be affected by the discharge and data patterns encountered [17], [18].

The second family of dual  $V_t$  circuits is one in which individual devices are partitioned to be either high  $V_t$  or low  $V_t$ , depending on their timing requirements. For example, gates in the critical path would be chosen to have low  $V_t$ , while noncritical gates would have high  $V_t$ 's, with correspondingly lower leakage currents [19]. This technique in general is only effective up to a certain point (diminishes with more critical paths in the circuit), and determining which paths can be made high  $V_t$  is a complex CAD problem [20].

An alternative application of dual  $V_t$  technology that can be very useful in microprocessor design is dual  $V_t$  domino logic [21]. In this style, individual gates utilize both high  $V_t$  and low  $V_t$  transistors, and the overall circuit will exhibit extremely low leakage in the standby mode, yet suffer no reduction in performance. This is achieved by exploiting the fixed transition directions in domino logic, and assigning a priori low-threshold voltages to only those devices in the critical charging/discharging paths. In effect, devices that can switch during the evaluate mode should be low  $V_t$  devices, while those devices that switch during precharge modes should be high  $V_t$  devices. Figure 3.15 shows a typical dual  $V_t$  domino stage used in a clock-delayed domino methodology, consisting of a pull-down network, inverter (I1), leaker device (P1), and clock drivers (I2, I3), with the low  $V_t$  devices shaded.

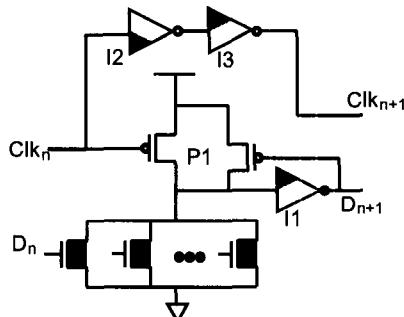


Figure 3.15 Dual threshold voltage domino logic gate.

During normal circuit operation, critical gate transitions occur only through low  $V_t$  devices, so high-performance operation is maintained. On the other hand, precharge transitions occur only through high  $V_t$  devices, but since precharge times in domino circuits are not in the critical path, slower transition times are acceptable. By having high  $V_t$  precharge transistors, it is possible to place the dual  $V_t$  domino gate into a very low leakage standby state merely by placing the clock in the evaluate mode and asserting the inputs. In a cascaded design with several levels of domino logic, the standby condition remains the same, requiring only an assertion of the first-level inputs. The correct polarity signal will then propagate throughout the logic to strongly turn off all high  $V_t$  devices and ensure low subthreshold leakage currents. In summary, dual  $V_t$  domino logic allows one to trade off slower precharge time for improved standby leakage currents. As a result, using dual  $V_t$  domino logic can achieve the performance of an all low  $V_t$  design, while maintaining the low standby leakage current of an all high  $V_t$  design.

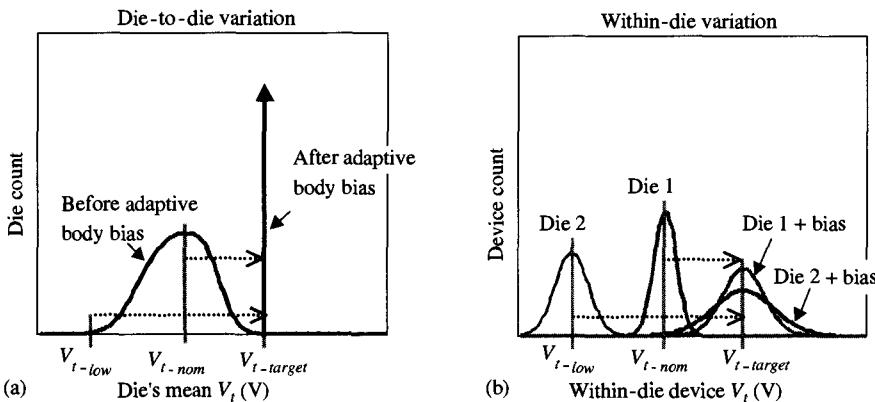
### 3.4.3 Adaptive Body Biasing (ABB)

Another technique to control subthreshold leakage is to modulate transistor  $V_t$ 's directly through body biasing. With application of maximum reverse body bias to transistors, threshold voltage increases, resulting in lower subthreshold leakage currents during standby mode. However, because the threshold voltage can be set dynamically, this technique can also be used to adaptively bias a circuit block during the active mode. Adaptive body biasing can be used to help compensate for large inter-die and within-die parameter variations by tuning the threshold voltage so that a common target frequency is reached. By applying reverse body bias to unnecessarily fast circuits, subthreshold leakage in the active mode can then be reduced as well. In order to use this technique, the initial process  $V_t$  should be targeted to a lower value than desired, and then reverse body bias can be applied to achieve a higher threshold voltage mean with lower variation.

Adaptive body biasing can easily be applied to a die as a whole (single PMOS body and NMOS body bias values for the whole chip), which will tighten the distribution of chip delays and leakage currents for a collection of dies. However, since die sizes and parameter variations are becoming larger with future scaling, within-die variation becomes a problem as well. ABB can be applied aggressively at the block level, where individual functional blocks within a chip, such as a multiplier or ALU, can be independently modulated to meet a common performance target. The following section, however, focuses on the effectiveness of adaptive body biasing applied at the die level, and further explores the limitations of technology scaling on this technique.

### 3.4.3.1 Impact of ABB on Die-to-Die and Within-Die Variations

As illustrated in Fig. 3.16a, adaptive body biasing technique matches the mean  $V_t$  of all the die samples close to the target threshold voltage, when they were all smaller than the target to begin with. Hence, to use adaptive body bias we first need to retarget the threshold voltage of the technology to be lower than what it would have been if adaptive body bias weren't used. Also, short-channel effects of a MOS transistor degrades with body bias [6]. So as technology is scaled, this adverse effect of body biasing poses an increasingly serious challenge to controlling short-channel effects and results in (1) reduction in effectiveness of adaptive body bias to control die-to-die mean  $V_t$  variation and (2) increase in within-die  $V_t$  variation. As illustrated in Fig. 3.16b, the die sample that requires larger body bias to match its mean  $V_t$  to the target threshold voltage will end up with higher within-die  $V_t$  variation. This increase in within-die  $V_t$  variation due to adaptive body bias can impact clock skew, worst-case gate delay, worst-case device leakage, and analog circuits.



**Figure 3.16** (a) Adaptive body bias reduces die-to-die variation in mean  $V_t$ .  
(b) Within-die  $V_t$  variation increases for die samples that require body bias to match their mean  $V_t$  to the target  $V_t$ .  $V_{t-target}$  is the target saturation threshold voltage for a given technology.  $V_{t-low}$  and  $V_{t-nom}$  are the minimum and mean threshold voltages of the die-to-die distribution.

### 3.4.3.2 Short-Channel Effects

In this section we describe short-channel effects, namely,  $V_t$ -roll-off and drain-induced barrier lowering (DIBL) that are affected by body bias. In a long-channel MOS, the channel charge is controlled primarily by the gate. As MOS channel length is scaled down, the source-body and drain-body reverse-biased diode junction depletion regions contribute a larger portion of the channel charge. This diminishes the control that gate and body terminals have on the channel, resulting in  $V_t$ -roll-off and body effect reduction [22]. Another short-channel effect of interest is reduction of the barrier for inversion charge to enter the channel from the source terminal with increase in

drain voltage. This dependence of MOS threshold voltage on drain voltage is DIBL. Both  $V_t$ -roll-off and DIBL degrade further with body bias because of widening diode depletions. The threshold voltage equation for short-channel MOS that captures the two short-channel effects is given as

$$V_t = V_{fb} + |2\phi_p| + \frac{\lambda_b}{C_{ox}} \sqrt{2qN\varepsilon_s(|2\phi_p| + V_{sb})} - \lambda_d V_{ds}$$

$$\lambda_b = 1 - \left( \sqrt{1 + \frac{2W}{X_j}} - 1 \right) \frac{X_j}{L}$$

$$\lambda_d = \left[ \frac{L}{2.2 \mu\text{m}^{-2}(T_{ox} + 0.012 \mu\text{m})(W_{sd} + 0.15 \mu\text{m})(X_j + 2.9 \mu\text{m})} \right]^{-2.7}$$

$\lambda_b$  models the  $V_t$ -roll-off and body effect degradation with channel length reduction, and  $\lambda_d$  models DIBL. This parameter is based on empirical fitting of device parameters and has been verified to be accurate down to 0.1  $\mu\text{m}$  channel length [23].

### 3.4.3.3 Effectiveness of ABB

We know that adaptive body bias requires (1) lower  $V_t$  devices and (2) body bias to reduce die-to-die mean  $V_t$  variation. We also know that as technology is scaled, body terminal's control on the channel charge diminishes. This is further aggravated if  $V_t$  has to be reduced and/or if body bias has to be applied since both result in increased diode depletions. Figure 3.17 illustrates the shift in threshold voltage of two 0.25  $\mu\text{m}$  MOS transistors. The two MOS transistors are identical in all aspects except in their threshold voltage values. The linear threshold voltages of the high- $V_t$  and the low- $V_t$  devices are 400 mV and 250 mV, respectively. It is clear from Fig. 3.17 that for 600 mV of body bias, the increase in threshold voltage for the high- $V_t$  device is significantly more than that of the low- $V_t$  device. The reasons for the reduced effectiveness of body bias for the low- $V_t$  device are (1) reduced channel doping required for  $V_t$  reduction means these devices will have lower body effect to begin with, (2) low- $V_t$  devices have more diode depletion charge degrading body effect, and (3) body bias increases diode depletion even more resulting in added body effect degradation. It has been shown in [24] that with aggressive 30%  $V_t$  scaling it will not be possible to match the mean  $V_t$  of all the die samples for 0.13  $\mu\text{m}$  technology.

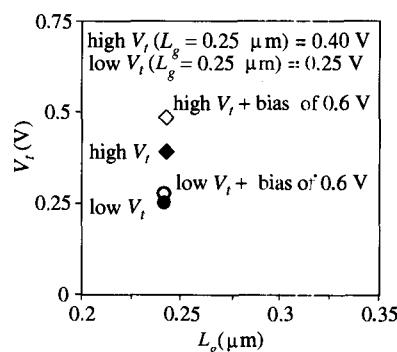
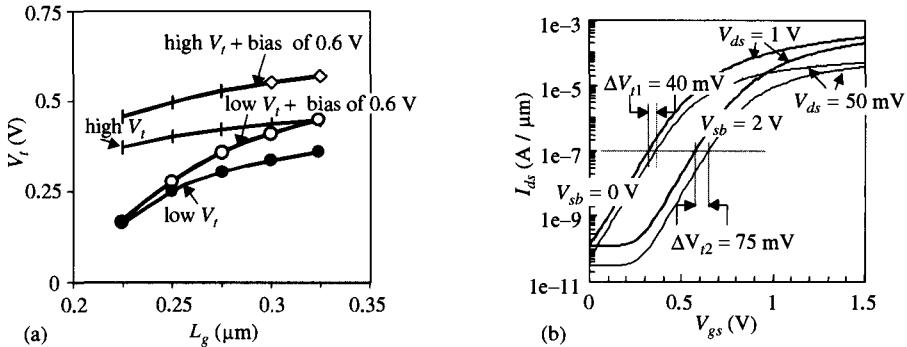


Figure 3.17 Body effect reduction for low- $V_t$  0.25  $\mu\text{m}$  device compared to a high- $V_t$ , 0.25  $\mu\text{m}$  device.



**Figure 3.18** (a) Increase in  $V_t$ -roll-off due to  $V_t$  lowering and body bias. (b) Increase in DIBL ( $\Delta V_t/\Delta V_{ds}$ ) due to body bias, for a  $0.25 \mu\text{m}$  NMOS.

### 3.4.3.4 Impact of ABB on Within-Die $V_t$ Variation

Low- $V_t$  devices that are required for adaptive body bias schemes have worse short-channel effects, and these effects degrade with body bias. As Fig. 3.18(a) illustrates,  $V_t$ -roll-off behavior is larger for a low- $V_t$  device compared to a high- $V_t$  device, and  $V_t$  roll-off increases further with body bias, as expected. Also, body bias increases DIBL ( $\Delta V_t/\Delta V_{ds}$ ) as expected, and this is depicted in Fig. 3.18(b).

Within-die  $V_t$  variation due to within-die variation in the critical dimension ( $\Delta L$ ) will depend on  $V_t$ -roll-off ( $\lambda_b$ ) and DIBL ( $\lambda_d$ ). So, increase in  $V_t$ -roll-off and DIBL due to adaptive body bias will result in a larger within-die  $V_t$  variation. It has been shown in [24] that this increase in within-die  $V_t$  variation due to adaptive body bias worsens with scaling and is more pronounced under aggressive  $V_t$  scaling. So, for effective use of adaptive body bias one has to consider the maximum within-die  $V_t$  variation increase that can be tolerated. It should also be noted that adaptive body bias will become less effective with technology scaling due to increasing transistor threshold voltage variation and degrading body effect.

## ACKNOWLEDGMENTS

We thank our colleagues K. Soumyanath, Kevin Zhang, Ian Young, and several others for providing insight into topics discussed in this paper. Bill Holt, Ricardo Suarez, Fred Pollack, and Richard Wirt provided continued support and encouragement.

## REFERENCES

- [1] M. Bohr, et al., "A High Performance  $0.25 \mu\text{m}$  Logic Technology Optimized for 1.8 V Operation," *IEDM Tech. Dig.*, p. 847, Dec. 1996.
- [2] M. Bohr, et al., "A High Performance  $0.35 \mu\text{m}$  Logic Technology for 3.3 V and 2.5 V Operation," *IEDM Tech. Dig.*, p. 273, Dec. 1994.
- [3] A. Keshavarzi, S. Narendra, S. Borkar, C. Hawkins, K. Roy, and V. De, "Technology Scaling Behavior of Optimum Reverse Body Bias for Standby Leakage Power Reduction in CMOS IC's," *1999 Int. Symp. On Low Power Electronics and Design*, p. 252, Aug. 1999.

- [4] A. S. Grove, *Physics and Technology of Semiconductor Devices*. John Wiley & Sons, New York, 1967.
- [5] A. W. Righter, J. M. Soden, and R. W. Beegle, "High Resolution  $I_{DDQ}$  Characterization and Testing-Practical Issues," *Int. Test Conf.*, pp. 259–268, Oct. 1996.
- [6] Y. P. Tsividis, *Operation and Modeling of the MOS Transistor*. McGraw-Hill, New York, 1987.
- [7] J. R. Brews, "The Submicron MOSFET," Chapter 3 in S.M. Sze, editor, *High Speed Semiconductor Devices*. John Wiley & Sons, New York, 1990, pp. 155–159.
- [8] R. F. Pierret, *Semiconductor Device Fundamentals*. Addison-Wesley, Reading, MA, 1996.
- [9] J. A. Mandelman and J. Alsmeyer, "Anomalous Narrow Channel Effect in Trench-Isolated Buried-Channel  $p$ -MOSFET's," *IEEE Elec. Dev. Ltr.*, vol. 15, no. 12, Dec. 1994.
- [10] A. Keshavarzi, K. Roy, and C. Hawkins, "Intrinsic Leakage in Low Power Deep Submicron ICs," *Int. Test Conf.*, p. 146, Nov. 1997.
- [11] J. P. Halter and F. Najm, "A Gate-level Leakage Power Reduction Method for Ultra-low-power CMOS Circuits," *Proc. IEEE CICC* 1997, pp. 475–478.
- [12] Y. Ye, S. Borkar, and V. De, "A New Technique for Standby Leakage Reduction in High-Performance Circuits," *1998 Symposium on VLSI Circuits*, June 1998, pp. 40–41.
- [13] R. X. Gu and M. I. Elmasry, "Power Dissipation Analysis and Optimization of Deep Submicron CMOS Digital Circuits," *IEEE Journal on Solid-State Circuits*, vol. 31, no. 5, pp. 707–713, May 1996.
- [14] M. C. Johnson, D. Somasekhar, and K. Roy, "Deterministic Estimation of Minimum and Maximum Leakage Conditions in CMOS Logic," *IEEE Transactions on Computer-Aided Design of IC's*, June 1999, pp. 714–725.
- [15] T. Kawahara, et al., "Subthreshold Current Reduction for Decoded-driver by Self-reverse Biasing," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 11, pp. 1136–1143, Nov. 1993.
- [16] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *IEEE JSSC*, vol. 30, no. 8, pp. 847–854, Aug. 1995.
- [17] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor Sizing Issues and Tool For Multi-Threshold CMOS Technology," *34th Design Automation Conference*, pp. 409–414, June 1997.
- [18] J. Kao, S. Narendra, and A. Chandrakasan, "MTCMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Patterns," *35th Design Automation Conference*, pp. 495–500, June 1998.
- [19] W. Lee, et al., "A 1V DSP for Wireless Communications," *ISSCC*, pp. 92–93, Feb., 1997.
- [20] L. Wei, Z. Chen, M. Johnson, and K. Roy, "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits," *35th Design Automation Conference*, pp. 489–494, June 1998.
- [21] J. Kao, "Dual Threshold Voltage Domino Logic," *25th European Solid State Circuits Conference*, pp. 118–121, Sep. 1999.
- [22] H. C. Poon, L. D. Yau, and R. L. Johnston, "DC Model for Short-Channel IGFETs," *Intl. Electron Device Meeting*, pp. 156–159, 1973.
- [23] K. K. Ng, S. A. Eshraghi, and T. D. Stanik, "An Improved Generalized Guide for MOSFET Scaling," *IEEE Trans. Electron Device*, vol. 40, no. 10, pp. 1895–1897, Oct. 1993.
- [24] S. Narendra, D. Antoniadis, and V. De, "Impact of Using Adaptive Body Bias to Compensate Die-to-die  $V_t$  Variation on Within-die  $V_t$  Variation," *Intl. Symp. Low Power Electronics and Design*, pp. 229–232, Aug. 1999.

Tadahiro Kuroda  
*Keio University*  
Takayasu Sakurai  
*University of Tokyo*

## 4.1 LOW-VOLTAGE LOW-THRESHOLD-VOLTAGE CIRCUIT DESIGN

### 4.1.1 Power, Energy, and Speed

Besides operating speed, energy is also an important metric for mobile applications, since the battery lifetime depends on energy. On the other hand, instantaneous power is also important for nonmobile applications since package type and the pertinent cooling device are determined by power dissipation.

CMOS power dissipation,  $P$ , and propagation delay time,  $T_{pd}$ , are approximately given by

$$P = p_t \cdot f_{CLK} \cdot C_L \cdot V_{DD}^2 + \frac{I_0}{W_0} W_T \cdot 10^{-V_{TH}/S} \cdot V_{DD} \quad (4.1)$$

$$T_{pd} = \frac{\beta \cdot C_L \cdot V_{DD}}{(V_{DD} - V_{TH})^\alpha} \quad (4.2)$$

where  $p_t$  is the switching probability,  $f_{CLK}$  is the clock frequency,  $C_L$  is the load capacitance,  $V_{DD}$  is the power supply voltage,  $V_{TH}$  is the threshold voltage,  $S$  is the sub-threshold slope (typically 0.1 V/decade),  $\alpha$  represents the velocity saturation effect (typically 1.4), and the other parameters are constants determined by circuit, layout, and device parameters. Power and delay dependence on  $V_{DD}$ ,  $V_{TH}$  are calculated by these equations [1].

If the leakage current is ignored, the power-delay (PD) product can be interpreted as the amount of energy (E) expected in each switching event, which is independent of the circuit speed and  $V_{TH}$ . It is, therefore, desirable to operate a circuit at the slowest possible speed at the lowest possible  $V_{DD}$  in order to save energy. However, most of the interesting applications require near-peak throughput. The energy-delay (ED) product, which is equivalent to  $PD^2$ , can be an important index for those who care about speed as much as energy. The calculation result suggests that  $V_{TH}$  should be lowered to about 0.1 V to minimize the ED product [1].

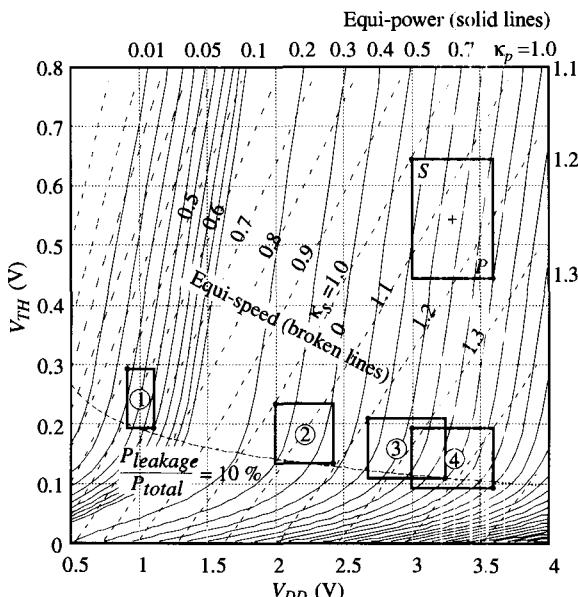
General guidelines for power reduction are observed from Eq. (4.1) as threefold: (1) lower  $V_{DD}$ , (2) reduce  $C_L$ , and (3) lower  $p_t$ . Lowering  $V_{DD}$  is the most attractive choice due to the quadratic dependence. However, as  $V_{DD}$  becomes lower, circuit delay increases and chip throughput degrades. In order to maintain the chip throughput at

low  $V_{DD}$ , there are four different approaches: (1) lower  $V_{TH}$  to recover the circuit speed, (2) employ dual- $V_{TH}$  process and lower  $V_{TH}$  only for critical circuits, (3) employ multiple supply voltages and lower  $V_{DD}$  only for noncritical circuits, and (4) utilize parallel and/or pipeline architectures to compensate for the degraded circuit speed. This chapter focuses on the first approach. To probe further into the other approaches, a recent edition of *Low-Power CMOS Design* from the IEEE Press [2] is a good reference.

#### 4.1.2 Low- $V_{DD}$ , Low- $V_{TH}$ Design Space

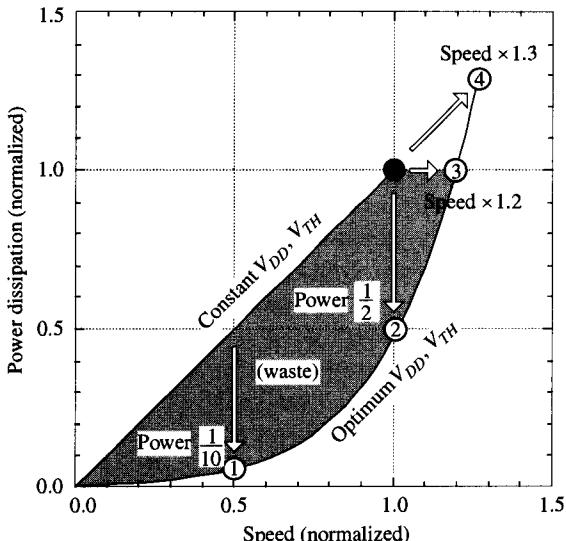
The calculated equi-power and equi-speed curves on the  $V_{DD}$ - $V_{TH}$  plane are depicted in Fig. 4.1. A rectangle in the figure illustrates ranges of  $V_{DD}$  change and  $V_{TH}$  fluctuation. Circuits are typically designed at  $V_{DD}$  of  $3.3\text{ V} \pm 10\%$  and  $V_{TH}$  of  $0.55\text{ V} \pm 0.1\text{ V}$  in a  $0.3\text{ }\mu\text{m}$  CMOS device. This rectangle is a design window because all the circuit specifications should be satisfied within the rectangle for yield-conscious design. In the design window, the circuit speed becomes the slowest at the upper-left corner  $S$ , while, at the lower-right corner  $P$ , the power dissipation becomes the highest. The equi-speed and equi-power curves are normalized at the  $S$  and  $P$  corners as designated by normalized factors  $\kappa_s$  and  $\kappa_p$ , so that how much speed and power are improved or degraded can be found, compared to those in a typical condition, by sliding and sizing the design window on the  $V_{DD}$ - $V_{TH}$  plane.

When the design window is moved toward lower  $V_{DD}$  and lower  $V_{TH}$  along the equi-speed curve, power dissipation is reduced. Since the subthreshold leakage current increases rapidly as  $V_{TH}$  is lowered, the power dissipation will be increased again at the point where the leakage current dominates the power dissipation. It is found from Fig. 4.1 that the power dissipation is minimum at the point around which power dissipation due to the subthreshold leakage current makes up 10% of the total power dissipation. This condition is also depicted as a broken line in the figure to indicate the optimum  $V_{TH}$ 's.



**Figure 4.1** Exploring low- $V_{DD}$ , low- $V_{TH}$  design space. Contour lines in terms of speed (broken lines) and power (solid lines) are drawn.

The low- $V_{DD}$ , low- $V_{TH}$  design space is explored in this way, and the results are summarized in Fig. 4.2. The optimum  $V_{DD}$  and  $V_{TH}$  according to speed requirements can prevent the waste of power and speed caused by the constant  $V_{DD}$  and  $V_{TH}$ . In this way, optimizing  $V_{DD}$  and  $V_{TH}$  is essential in low-power, high-speed CMOS design, whereas they are given as constant and common parameters in the conventional CMOS design.



**Figure 4.2** Speed and power saving by optimum  $V_{DD}$  and  $V_{TH}$ .

#### 4.1.3 Design Issues at Low $V_{DD}$ and Low $V_{TH}$

Lowering both  $V_{DD}$  and  $V_{TH}$  enables high-speed, low-power CMOS circuit operation. This approach, however, raises three problems: (1) increase in standby power dissipation in low  $V_{TH}$ , (2) inability to sort out defective chips by monitoring the quiescent power supply current (IDQ testing), and (3) degradation of worst-case speed due to  $V_{TH}$  fluctuation in low  $V_{DD}$ .

Standby power is important not only for portable equipment but also for a microprocessor, since such a processor is in standby most of the time. Without the IDQ testing, it is more difficult to develop test vectors for high test coverage as integration level improves. For these reasons it is very difficult to lower  $V_{TH}$  below 0.5 V.

In order to keep the delay variation percentage due to  $V_{TH}$  fluctuation ( $\Delta V_{TH}$ ) constant in low  $V_{DD}$ ,  $\Delta V_{TH}$  should be scaled by [6]

$$\frac{\Delta V'_{TH}}{\Delta V_{TH}} = \left( \frac{T_{pd}}{T'_{pd}} \cdot \frac{V'_{DD}}{V_{DD}} \right)^{\frac{1}{\alpha}} \quad (4.3)$$

For example,  $V_{DD} = 3.0$  V,  $V'_{DD} = 2.0$  V, and  $T_{pd} = T'_{pd}$ ,  $\Delta V_{TH}$  should be reduced by 25%. However, it is difficult to reduce  $\Delta V_{TH}$  by means of process and device refinement.

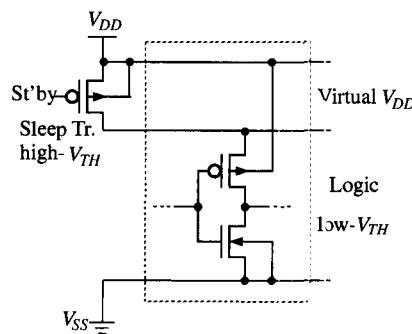
There are two approaches to solving these three problems. Conventional power-down schemes either on a board or in a chip can solve the battery life problem. On the other hand, controlling  $V_{TH}$  through substrate bias can solve all three problems.

## 4.2 POWER-DOWN SCHEME

Power-down scheme is a simple and straightforward approach to cut the leakage current and save battery life, but it may increase design complexity and pose drawbacks and limitations. For example, in a cellular phone, radio condition and base station information should be checked once per second in a cell-waiting mode. Therefore, circuits should be appropriately partitioned into two groups, depending on whether or not the power supply can be cut. Another example of an increase in design complexity is found in an application that requires data saving and loading between on-chip registers and off-chip memory devices, since data can be lost when power is down. Furthermore, when power is down, output voltages become unstable in the  $0 \pm 0.7$  V range due to drain-well junctions. In order to prevent leakage in the succeeding circuit a pull-down resistor or a gate keeper is required; however, this increases power dissipation and propagation delay. The limitation in this approach comes from the fact that frequent power switching is difficult, since it requires a longer than 100  $\mu$ s settling time before the supply voltage is stable. For instance, in the Windows-CE operating system, return to the ON state from the IDLE state should be accomplished within 1  $\mu$ s. Therefore, the power supply cannot be turned off in the IDLE mode.

The power-down scheme on a board brings up further design considerations. All the input signals to a chip should be set to the low level in advance to keep current from flowing into the chip through an ESD protection device in an input Pad. When turning on the power, the procedure should be in the reverse order, except for low-active inputs such as chip-enable-bar ( $CE$ ) and output-enable-bar ( $OE$ ). They should be reset to the high level in advance, to prevent erroneous operation in the circuits during power-up, but current injection from these inputs may cause latch-up.

The power-down scheme in a chip addresses another design issue. In a Multiple-Threshold CMOS (MTCMOS) technology [3] depicted in Fig. 4.3, internal power is cut in the standby mode by a high- $V_{TH}$  sleep transistor. Since the substrate of PMOS should be tied to  $V_{DD}$ , both  $V_{DD}$  and virtual  $V_{DD}$  lines should be incorporated in a cell, which causes area penalty.



**Figure 4.3** Multiple-Threshold CMOS (MTCMOS).

Correct sleep transistor sizing is a key parameter that affects the performance of MTCMOS circuits [4]. If sized too large, silicon area would be wasted and switching energy overhead would be increased. If sized too small, on the other hand, the circuit speed would be degraded due to the virtual  $V_{DD}$  bounce. Actually, the worst-case delay in MTCMOS is strongly dependent on the discharge patterns of internal gates, which

**TABLE 4.1** Simulated Delay of 8 Bit Multiplier in CMOS and MTCMOS

Input vector (Initial) → (Final)	CMOS Delay	MTCMOS	
		W/L = 60 Degradation (%)	W/L = 170 Degradation (%)
A: (X = 00, Y = 00) → (X = FF, Y = 81)	8.96 ns	18.1%	5.0%
B: (X = 7F, Y = 81) → (X = FF, Y = 81)	8.93 ns	4.8%	1.7%

causes the virtual  $V_{DD}$  line to fluctuate depending on discharge pattern through the sleep transistor. The worst-case input vector is difficult to predict and can even be different from a vector that exercises a critical path in an ordinary CMOS implementation. For example, Table 4.1 shows SPICE simulation results of an  $8 \times 8$  bit carry save multiplier in CMOS and MTCMOS [4]. If one wished to size the sleep transistor to provide less than 5% speed penalty for vector A, then one must size the sleep transistor greater than W/L = 170. On the other hand, if one were to examine the vector B, the same analysis could lead one to erroneously size the sleep transistor to be only W/L = 60, which would actually correspond to an 18% degradation in speed for the previous case. In this way, it is very difficult to correctly size the sleep transistor.

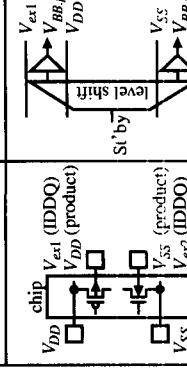
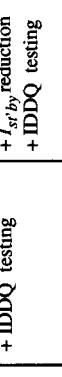
As one continues to scale  $V_{DD}$  to lower voltages, the effective resistance of the sleep transistor increases dramatically, requiring an even larger size sleep transistor. Finally, at lower than 0.7 V, MTCMOS will not work, because the high- $V_{TH}$  transistor will not turn on. A super cutoff CMOS (SCCMOS) circuit overcomes this problem [5]. A low- $V_{TH}$  sleep transistor is employed for low-voltage operation, and its gate is driven to about  $V_{DD} + 0.4$  V in the standby mode to fully cut off the leakage current. Supply voltage scaling below 0.7 V is possible, and at the same time pico-ampere standby current per logic gate can be achieved.

## 4.3 CONTROLLING $V_{TH}$ THROUGH SUBSTRATE BIAS

### 4.3.1 Variable Threshold-Voltage CMOS (VTCMOS) Technology

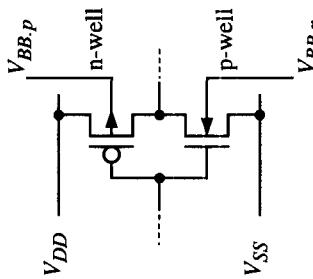
A variable threshold-voltage CMOS (VTCMOS) technology [6]–[10] controls  $V_{TH}$  by means of substrate bias control, as depicted in Fig. 4.4. The  $V_{TH}$  variation can be compensated for by feedback controlling the substrate bias,  $V_{BB}$ , using an on-chip self-substrate bias circuit (SSB) and a leakage current monitor (LCM), such that monitored leakage current is set to a target value. Three schemes are developed. A self-adjusting threshold-voltage scheme (SAT) compensates for the  $V_{TH}$  variation [7]. A standby power reduction scheme (SPR) reduces the subthreshold leakage current [7],[8]. An SAT + SPR scheme solves all the three problems [6],[9],[12]. The three schemes with their variants are summarized in Fig. 4.4.

In VTCMOS, it is considered that the usage of a transistor is expanded from the conventional three-terminal device, consisting of the drain, the gate, and the source, to a four-terminal device by adding the substrate. Except for the unique property wherein  $V_{TH}$  is variable, VTCMOS holds all the properties of CMOS. Consequently, the conventional design methodologies and design infrastructures can be employed as before, and no such penalties as in the conventional power-down schemes are induced.

	SAT	SPR (standby power reduction)	SAT + SPR
$V_{BB,p}$ (active) (standby)	$V_{DD} + 0.3\text{ V}$ —	$V_{DD}$ $V_{DD} + 1.2\text{ V}$ $V_{SS}$ $V_{SS} - 2.0\text{ V}$	$V_{DD} + 0.3$ $V_{DD} + 1.7\text{ V}$ $V_{SS} - 0.5\text{ V}$ $V_{SS} - 3.0\text{ V}$
$V_{BB,n}$ (active) (standby)	$V_{SS} - 0.5\text{ V}$ —		
$V_{TH}$ (process) (active) (standby)	$0.1\text{ V} \pm 0.10\text{ V}$ $0.2\text{ V} \pm 0.05\text{ V}$ —	$0.2\text{ V} \pm 0.10\text{ V}$ $0.2\text{ V} \pm 0.10\text{ V}$ $0.5\text{ V} \pm 0.05\text{ V}$	$0.1\text{ V} \pm 0.10\text{ V}$ $0.2\text{ V} \pm 0.05\text{ V}$ $0.5\text{ V} \pm 0.05\text{ V}$
$V_{BB,p}$			
		Sub separation	SPR w/SSB
			
	Circuit		
			
	Active St'by	$V_{DD}$	$V_{DD}$
			$V_{DD}$
	Transition time	—	short to $V_{DD}, V_{SS}$
		—	$V_{SS}$
	+ $\Delta V_{TH}$ compensation	+ IDDQ testing	$0.1\mu\text{s}$ (st'by to active) $0.1\mu\text{s}$ (active to st'by)
			$0.1\mu\text{s}$ (st'by to active) $100\mu\text{s}$ (active to st'by)
	Effect	$+ I_{st'by}$ reduction + IDDQ testing	$+ I_{st'by}$ reduction + IDDQ testing
			$+ \Delta V_{TH}$ compensation + $I_{st'by}$ reduction + IDDQ testing
	Penalty	-300 $\mu\text{m}\text{-sq}$ macro	-70 $\mu\text{m}\text{-sq}$ macro -2 additional supplies -triple-well
	Fabricated chip	testchip [7]	testchip [7]
			188k-gate GA [8]
			DCT [6] RISC [12]
			MPEG-4 [9]

SAT: Self-adjusting threshold-voltage, LCM: Leakage current monitor, SSB: Self-substrate bias, SCI: Substrate charge injector

Figure 4.4 VT CMOS and variants.



Compared to MTCMOS, circuit overhead is much smaller, since much smaller current flows in the substrate than in the power supply lines. In VTCMOS, it takes  $100\ \mu s$  for SSB to pump current out from the substrate for transitioning to the standby mode, whereas it takes  $0.1\ \mu s$  for a transistor to inject current to the substrate for going to the active mode. This “slow falling asleep but quick awakening” feature is effective in many applications. In the example of the Windows-CE operating system, every time an MPU moves to the IDLE state, VTCMOS can raise  $V_{TH}$  to reduce the leakage current, since it can lower  $V_{TH}$  for the ON state in  $0.1\ \mu s$ .

### 4.3.2 VTCMOS Circuit Techniques

Two circuit techniques are essential: (1) feedback control of substrate bias by SSB and LCM for  $\Delta V_{TH}$  compensation, and (2) a combination of a switch transistor, which connects the substrate to  $V_{DD}/V_{SS}$  in the active mode, and a substrate bias control by SSB and LCM, which will reverse bias the substrate in the standby mode.

A circuit schematic of the SSB + LCM feedback control is depicted in Fig. 4.5 [6]. LCM activates SSB when the monitored leakage current in LCM,  $I_{leak.LCM}$ , is larger than a target preset value,  $I_{ref}$ . SSB lowers  $V_{BB}$  by pumping out current from the substrate. Accordingly,  $V_{TH}$  is raised and  $I_{leak.LCM}$  is reduced. When  $I_{leak.LCM}$  becomes smaller than  $I_{ref}$ , LCM stops SSB. However, the substrate current due to impact ionization and junction leakage raises  $V_{BB}$  gradually again. Accordingly,  $V_{TH}$  is lowered gradually and  $I_{leak.LCM}$  increases. When  $I_{leak.LCM}$  becomes larger than  $I_{ref}$ , LCM activates SSB again. By activating SSB intermittently in this way,  $V_{TH}$  can be set to the target value, and consequently, its process-induced fluctuation can be compensated.

The ratio of  $I_{leak.LCM}$  to the total leakage current in a chip,  $I_{leak.chip}$ , or the leakage current detection ratio,  $X_{LCM}$ , is given by

$$X_{LCM} \equiv \frac{I_{leak.LCM}}{I_{leak.chip}} = \frac{W_{LCM} 10^{(V_b - V_{TH})/S}}{W_{chip} 10^{-V_{TH}/S}} = \frac{W_{LCM}}{W_{chip}} \cdot 10^{\frac{V_b}{S}} \quad (4.4)$$

where  $W_{chip}$  is effective total channel width corresponding to the total leakage current in the chip,  $W_{LCM}$  is channel width of a monitor transistor in LCM, and  $V_b$  is its gate potential. Since  $I_{leak.LCM}$  leads to a power penalty of LCM, it should be as small as possible. Too small an  $I_{leak.LCM}$  value, however, slows LCM response speed, which enlarges unevenness in  $V_{BB}$  caused by the on-off control of SSB, resulting in larger

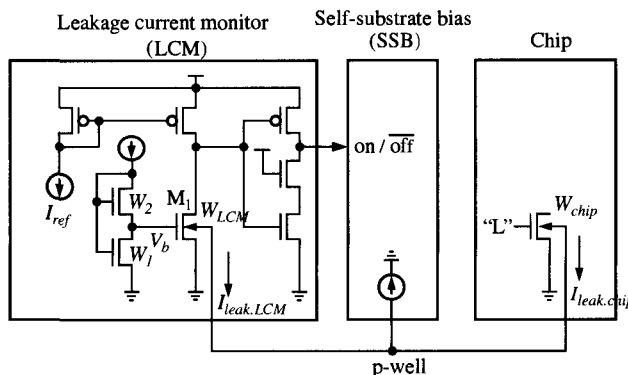


Figure 4.5 SSB + LCM feedback control circuit.

dynamic error of  $V_{TH}$ . In this design,  $I_{leak,LCM}$  is set to  $1\text{ }\mu\text{A}$ . When chip leakage current in the active mode is  $1\text{ mA}$ ,  $X_{LCM}$  is  $0.1\%$ . Given  $V_b = 2S$ , which is approximately  $0.2\text{ V}$ , the size of the monitor transistor can be designed as small as approximately  $0.001\%$  of the effective total transistors in the chip.

A bias circuit for  $V_b$  is implemented by connecting two transistors whose channel widths are  $W_1$  and  $W_2$ , as depicted in Fig. 4.5, and by designing a current source such that the two transistors are operated in the subthreshold region. As the drain currents of the two transistors are equal,

$$W_2 \cdot 10^{(V_1 - V_b - V_{TH})/S} = W_1 \cdot 10^{(V_1 - V_{TH})/S}$$

$$\therefore V_b = S \cdot \log \frac{W_2}{W_1} \quad (4.5)$$

Substituting Eq. (4.5) into Eq. (4.4),

$$X_{LCM} = \frac{W_{LCM}}{W_{chip}} \cdot \frac{W_2}{W_1} \quad (4.6)$$

$X_{LCM}$  can be designed only by transistor size ratio and can be independent of the power supply voltage, temperature, and process fluctuation. If  $V_b$  is generated by dividing voltages between  $V_{DD}$  and  $V_{SS}$  by registers,  $V_b = \lambda \cdot V_{DD}$ , and consequently,  $X_{LCM}$  is a function of  $V_{DD}$  and  $S$ . Since  $S$  is a function of temperature,  $X_{LCM}$  depends on  $V_{DD}$  and temperature, which is not desirable. Variation in  $X_{LCM}$ , analyzed by SPICE simulation, is within  $15\%$ , which results in less than  $1\%$  error in  $V_{TH}$  controllability.

A circuit implementation of the SPR w/ SSB scheme is depicted in Fig. 4.6 [8]. The substrate is connected to  $V_{DD}/V_{SS}$  by a switch transistor  $M_1$  in the active mode and is reverse biased by SSB and LCM in the standby mode. In order to turn off the switch transistor completely in the standby mode, a voltage lower than  $V_{BB}$  should be applied to the gate. For this purpose, a diode  $D_1$  is inserted between the gate and the source of  $M_1$ , and SSB is connected to the gate. When SSB pumps current out and the gate potential for  $M_1$  reaches  $-0.7\text{ V}$ , diode  $D_1$  turns on. Thereafter, SSB keeps the gate-source voltage at  $-0.7\text{ V}$  to turn  $M_1$  off completely, and pumps the substrate current out to lower the substrate potential. At this point, to prevent the transistor  $M_2$  from receiving over-voltage, a transistor  $M_3$ , whose gate is connected to  $V_{SS}$ , is inserted between the drain for  $M_2$  and the gate for  $M_1$  to clamp any  $M_2$  drain potential higher than  $V_{SS} + V_{TH}$ . When transitioning from the standby to the active modes, on the other hand, both LCM and SSB are disabled,  $M_2$  is turned on, and so is  $M_1$ . At this point, the

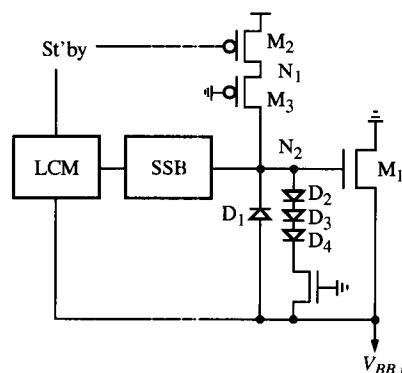


Figure 4.6 SPR w/ SSB circuit.

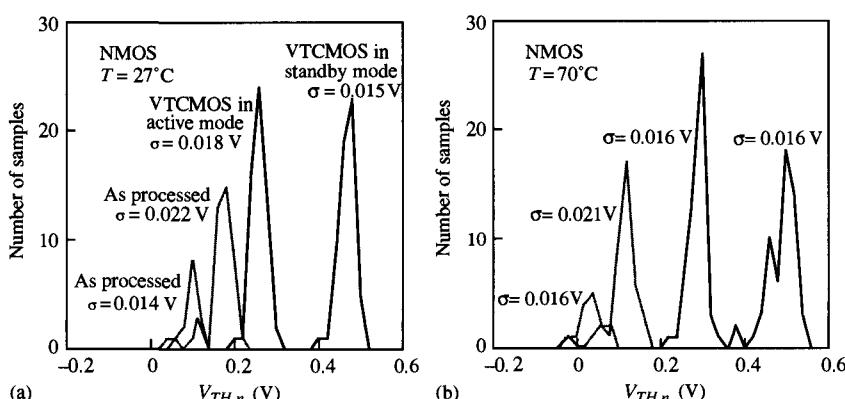
$M_1$  gate voltage should be raised gradually in accordance with the rise in the substrate potential for reliability. By inserting three diodes,  $D_2$ ,  $D_3$ , and  $D_4$ , between the gate and the source, the gate-source voltage for  $M_1$  is clamped to 1.8 V for protection. When the substrate potential rises to  $V_{SS} - V_{TH}$ , this clamp circuit is released, and  $V_{DD}$  is provided to the gate to turn on  $M_1$  strongly.

### 4.3.3 VTCMOS Performance and Penalty

Two experimental chips were fabricated based on a 0.3  $\mu\text{m}$  CMOS technology. One is an MPEG-4 video codec, which employs the SAT + SPR scheme [9]. Three million transistors are integrated in a 9  $\text{mm}^2$  die.  $V_{TH}$  controllability, power penalty, substrate current dependence, and latch-up immunity were investigated. The other chip is a gate array which employs the SPR w/ SSB scheme [8]. The  $M_1$  transistor in the SAT + SPR circuit is distributed by utilizing unused transistors in I/O cells. The chip includes available gate counts of 188,000 gates in an 8 mm die, where a Delay-Locked Loop (DLL), a 4K-bit SRAM, and 64 kinds of gate chains are implemented. Substrate noise influence is investigated by evaluating these noise-sensitive circuits [10]. Wafers were fabricated for several different  $V_{TH}$  values ranging from  $-0.1$  to  $0.2$  V, by changing conditions in ion implantation. External supply voltage is 3.3 V for the I/O Pads and lower supply voltages are provided to internal circuits.

#### 4.3.3.1 $V_{TH}$ Controllability

Die-to-die  $V_{TH}$  variation is investigated by measuring  $V_{TH}$  of a monitor transistor in the MPEG-4 chip. About 40 chips were measured for each  $V_{TH}$  condition in the following three ways: (1)  $V_{TH}$  as processed when the VTCMOS control is disabled and the n-well and the p-substrate are tied to  $V_{DD}$  and  $V_{SS}$ , respectively; (2)  $V_{TH}$  controlled by VTCMOS in the active mode; and (3)  $V_{TH}$  controlled by VTCMOS in the standby mode. In (2) the MPEG-4 chip is operated with test vector inputs so that the measurements include dynamic errors, such as those due to substrate noise influence. The measured NMOS  $V_{TH}$  values at 27°C and 70°C are plotted in Figs. 4.7(a) and (b). PMOS exhibits a similar trend. Four distributions are shown in each figure: (1) as



**Figure 4.7** Measured  $V_{TH}$ . (a)  $V_{TH,n}$  at 27°C and (b)  $V_{TH,n}$  at 70°C.  $\sigma$  is the standard deviation for each distribution.

processed, with target  $V_{TH}$  of 0.05 V; (2) as processed, with target  $V_{TH}$  of 0.15 V; (3) VTCMOS in the active mode, measured from all the chips in both (1) and (2); and (4) VTCMOS in the standby mode, measured from all the chips in both (1) and (2). Putting (1) and (2) together is considered to represent the typical  $V_{TH}$  variation of  $\pm 0.1$  V. It is found that the VTCMOS technology reduces  $V_{TH}$  variation from  $\pm 0.1$  V to  $\pm 0.05$  V in both the active and the standby modes, and raises  $V_{TH}$  by 0.25 V in the standby mode.

Measured temperature dependence for  $V_{TH}$  is 0.7 mV/ $^{\circ}$ C for an NMOS and is  $-0.7$  mV/ $^{\circ}$ C for a PMOS under the VTCMOS control, whereas the values are  $-1.3$  mV/ $^{\circ}$ C and 2.0 mV/ $^{\circ}$ C, respectively, in the conventional CMOS device. In the conventional CMOS, as the temperature is raised, the effective carrier mobility decreases, so that the drain current is reduced and the circuit delay is increased.

For sub 1 V, especially around 0.5 V, the drain current shows positive temperature dependence, since the increase in the drain current by  $V_{TH}$  decrease surmounts the mobility degradation [11]. This may result in thermal runaway in an asynchronous circuit, since increased current raises operating frequency to increase power dissipation and raise operating temperature. Although the frequency does not change in a synchronous circuit, if the subthreshold leakage becomes the dominant component in power dissipation at low  $V_{TH}$ , the power dissipation exhibits highly positive temperature dependence, and the thermal runaway may occur. Thus, in a scaled device with low  $V_{DD}$  and low  $V_{TH}$ , temperature dependence control becomes indispensable. The temperature dependence for  $V_{TH}$  in VTCMOS can be controlled by circuit design, since it is determined by the temperature dependence of the target leakage current source in LCM.

Even though VTCMOS can compensate for die-to-die  $V_{TH}$  variation, it may increase within-die  $V_{TH}$  variation [12]. In deep submicron devices, dimension variation in gate poly-Si is the dominant reason for the within-die  $V_{TH}$  variation. Since the reverse substrate bias extends the drain substrate depletion layer, the short-channel effect becomes worse. Furthermore, as the channel length is shorter, channel potential is more influenced by the drain than by the substrate, and the body effect coefficient,  $\gamma$ , is further reduced. As a result,  $V_{TH}$  increase, caused by the reverse substrate bias, becomes smaller in a shorter channel transistor. Coupled with the short-channel effect,  $V_{TH}$  variation, due to the channel length variation, is enlarged, when the reverse substrate bias is applied.

From the  $V_{TH}$  measurement, this effect is not found. The standard deviation is not more increased in the standby mode than in the active mode, even though a deeper substrate bias is applied. Since chip leakage current reflects  $V_{TH}$  for all the transistors in the chip, within-die  $V_{TH}$  variation should be better observed. Chip leakage current was measured at 27  $^{\circ}$ C and 70  $^{\circ}$ C, and the results are plotted in Figs. 4.8(a) and (b). The horizontal axes in both figures are the average of  $|V_{TH,p}| + V_{TH,n}$ , measured as processed at 27  $^{\circ}$ C. The VTCMOS technology sets the leakage current below 10 mA in the active mode and below 10  $\mu$ A in the standby mode, independently from processed  $V_{TH}$  and temperature. The lower  $V_{TH}$  is, the deeper the bias which is applied to the substrate. However, the deviation in the leakage current does not increase. It seems that impact resulting from using VTCMOS on within-die  $V_{TH}$  variation is negligible in this device technology.

#### **4.3.3.2 Power and Area Penalty**

Power penalty for a VTCMOS circuit, such as SSB and LCM, is very small. Measured power current for the VTCMOS control circuit ( $I_{VTCMOS}$ ), as well as power current ( $I_{DD}$ ) and substrate current ( $I_{p\text{-well}}, I_{n\text{-well}}$ ) for the MPEG-4 chip, are depicted in

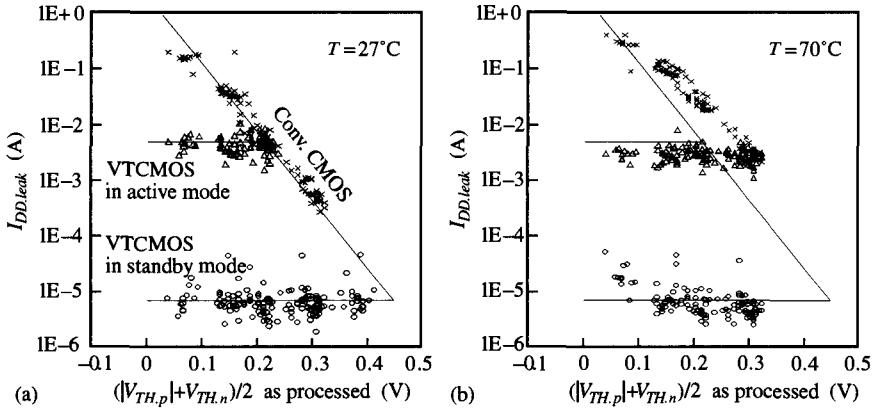


Figure 4.8 Measured chip leakage current (a) at 27°C and (b) at 70°C.

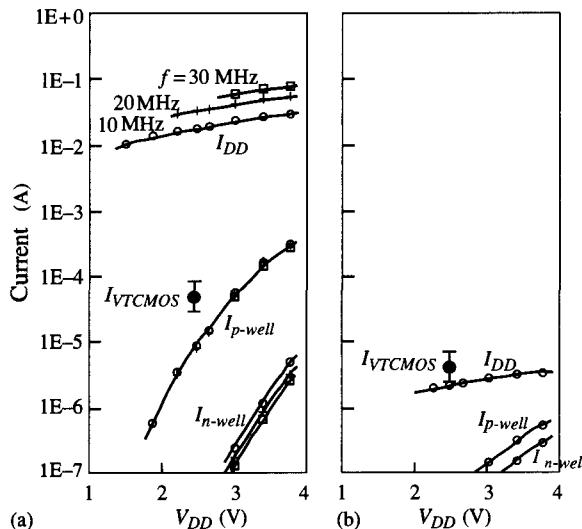
Figure 4.9 Measured power current of VTCMOS circuit ( $I_{VTCMOS}$ ) as well as power current ( $I_{DD}$ ) and substrate current ( $I_{p\text{-well}}$ ,  $I_{n\text{-well}}$ ) of MPEG-4 chip at 27°C in (a) active mode, when  $V_{TH}$  is set at 0.2 V, and in (b) standby mode, when  $V_{TH}$  is set at 0.5 V.  $I_{VTCMOS}$  is excluded from  $I_{DD}$ .

Fig. 4.9. Substrate current due to impact ionization is approximately only a few tenths of a percent of the power current at 2.5 V. SSB in its equilibrium control pumps out the same amount of the substrate current as is generated, for which SSB consumes several times more current from the power supply. Consequently, SSB power dissipation is less than 1% of the chip power current, when the substrate potential is under equilibrium control. LCM, on the other hand, consumes power all the time to monitor the leakage current. The smaller the power for LCM, the slower the response time, and the larger is the control error for  $V_{BB}$ . Currently, LCM is designed to consume 3  $\mu$ A current. In the standby mode, this static power is larger than the dynamic power for SSB. In most applications, however, smaller than 10  $\mu$ A standby current is considered negligible.

From the fact that the substrate current is almost independent from the operating frequency in the active mode, it is inferred that current due to the impact ionization at nonswitching transistors dominates the substrate current at low  $V_{TH}$ . Design knowledge is derived from the fact that SSB pumping capability should be determined only by  $V_{DD}$  and the total transistor width of a chip, and independently from the operating frequency.

Switching between the active mode and the standby mode, by charging and discharging substrate capacitance, also consumes power. Suppose, for instance, that the substrate capacitance for  $10 \text{ mm}^2$  is  $5 \text{ nF}$ , and that the substrate potential is changed by  $3 \text{ V}$ . Energy required for charging and discharging the substrate each time is  $5 \text{ nF} \times (3 \text{ V})^2$ , that is  $0.05 \mu\text{joule}$ . For example, in the Windows-CE operating system, when an MPU goes back and forth between the IDLE mode and the ON mode every  $25 \text{ ms}$ , due to a timer interruption, the power penalty is as small as  $2 \mu\text{W}$ .

Area penalty of the VTCMOS circuit itself is less than 1%, but area penalty for separation of the substrate contacts may be 6% at most. This penalty can be reduced if the number of the substrate contacts is reduced, which raises concerns about the substrate noise influence and the latch-up immunity.

#### **4.3.3.3 Substrate Noise Influence**

Substrate noise influences on circuit performance were investigated in the gate array chip [10]. Tracking jitter for DLL and a shmoo plot for SRAM, both of which are very sensitive to substrate noise, were measured under noise generated by the 64 gate chains. The p-well and the n-well alternately run vertically in the chip, so that DLL and SRAM share the same p-wells and n-wells with the gate chains. Two variants are designed in terms of the number of the substrate contacts per well strip: (1) only one substrate contact at the bottom; and (2) 400 substrate contacts in equal density. In (1), all the 64 gate chains are operated at  $50 \text{ MHz}$ , whereas, in (2), none of them are activated. No distinguishable difference is found between the two variants in the measured DLL tracking jitter and measured SRAM shmoo plot. Several reasons are considered. Noise in one phase is canceled out by that in the opposite phase. Only a small number of gates are propagating signals at any given moment, while all the rest are static and stabilizing the substrate potential with their junction capacitance. At lower supply voltages, the smaller substrate noise is expected, due to the impact ionization and capacitance coupling between the drain and the substrate. Simultaneous switching for a large amount of data at a local area, however, should be treated with care.

#### **4.3.3.4 Latch-Up Immunity**

Generally the following three triggers are considered to initiate latch-up: (1) forward bias of the substrate with respect to the source diffusions during power-up due to capacitance coupling between them, (2) substrate current due to impact ionization in the active mode, and (3) current injection from the I/O circuits due to overshoot and undershoot of input and output signals.

While SSB is not sufficiently operated during power-up, the potential rise in the p-substrate with respect to the n-channel transistor source,  $\Delta V_{BB,psub}$ , is given by

$$\Delta V_{BB,psub} = \frac{C_p \cdot C_{well}}{C_p \cdot C_{well} + C_n(C_p + C_{well})} \Delta V_{DD} \quad (4.7)$$

where  $C_p$  is capacitance between the p-channel transistor source and the n-well,  $C_{well}$  is capacitance between the n-well and the p-substrate,  $C_n$  is capacitance between the p-substrate and the n-channel transistor source, and  $\Delta V_{DD}$  is the power supply voltage change. For example, for  $C_p = C_n = 1 \text{ nF}$  and  $C_{well} = 4 \text{ nF}$ ,  $\Delta V_{BB,psub} = 0.44\Delta V_{DD}$ . When  $\Delta V_{DD} > 1.6 \text{ V}$ ,  $\Delta V_{BB,psub} > 0.7 \text{ V}$ , and a parasitic lateral npn-bipolar transistor,  $Q_L$ , is turned on. The collector current of  $Q_L$  in turn lowers the potential of the n-well against the p-channel transistor source, and turns on a parasitic vertical pnp-bipolar transistor,  $Q_V$ , to initiate latch-up. Actually, latch-up occurs during power-up to 4 V in the 0.3  $\mu\text{m}$  device, which disables a burn-in test at 4.4 V. A power-on-shunt circuit can prevent latch-up by shunting the p-substrate to  $V_{SS}$  and the n-well to  $V_{DD}$  during power-up.

Internal latch-up in the active mode can be prevented by designing a sufficiently high current pumping capability for SSB using the measured data of substrate current in Fig. 4.9.

I/O latch-up is reduced, since the substrate is not tied to the power lines. For example, when undershoot signal of  $-V_{in}$  ( $< -0.7 \text{ V}$ ) is received at an input, current flows out from the p-substrate through an input ESD protection device. In this case, the p-substrate is biased to  $-V_{in} + 0.7 \text{ V}$  ( $< 0 \text{ V}$ ). Therefore,  $Q_L$  is unlikely to turn on. When an overshoot signal is received,  $Q_V$  is unlikely to turn on in the same way. However, when another n-well exists close by,  $Q_V$  may eventually turn on there. One way to prevent it is to apply high bias to the n-well for the input protection device to keep  $Q_V$  from being turned on by the overshoot signal. Another sure method is to employ a triple-well structure to bias only the internal p-well and n-well, which are completely separated from the I/O portion by a deep n-well. In this approach, high  $V_{TH}$  is also necessary for the I/O circuits.

In any case, when  $V_{DD}$  becomes lower than the holding voltage (typically around 1 V), latch-up never occurs. As long as  $V_{DD}$  for the burn-in test is higher than the holding voltage, latch-up prevention should be considered in circuit and layout designs. Eventually, however, low-voltage CMOS will be free from latch-up.

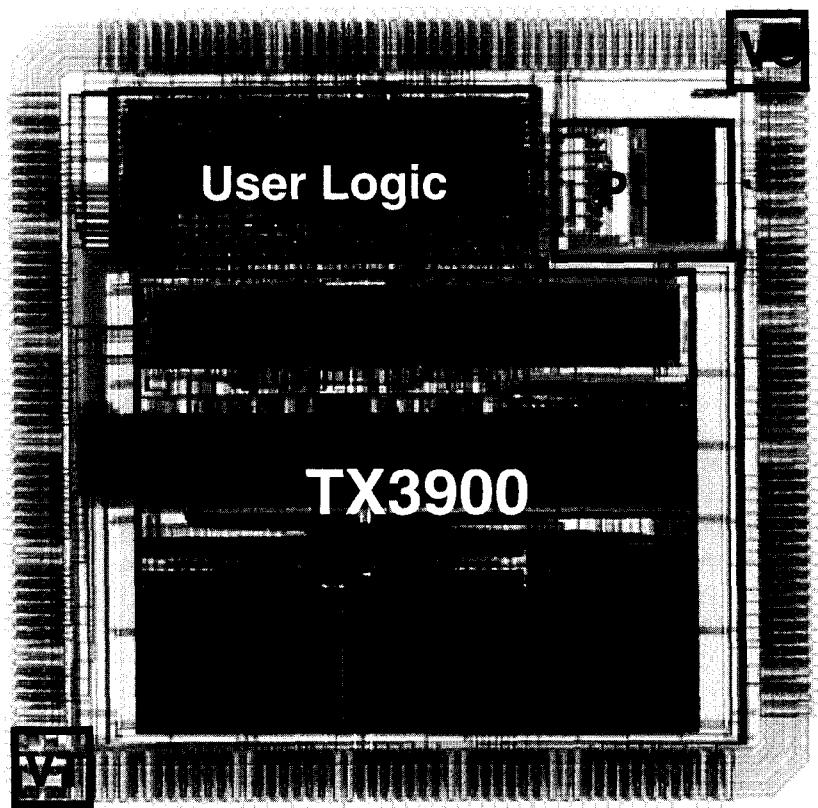
## 4.4 PROCESSOR DESIGN EXAMPLES

### 4.4.1 TX3900

A 32-bit RISC core processor, TX3900, is designed in VT CMOS which employs a compatible instruction set architecture with the MIPS R3900 [13]. The internal supply voltage is generated by a variable supply-voltage (VS) scheme [13], which is an adaptive  $V_{DD}$  control with an on-chip DC-DC converter.  $V_{TH}$  is controlled by the SAT + SPR scheme.

The chip is implemented by about 440,000 transistors, including 32-bit MAC, 4 kB direct mapped instruction cache, and 1 kB two-way set-associative data cache. The chip is fabricated in a 0.4  $\mu\text{m}$  CMOS n-well/p-sub double-metal technology. A chip micro-photograph appears in Fig. 4.10. Main features are summarized in Table 4.2. A VS macro and a VT macro are small enough to be added at the corners of the chip. Substrate contacts are connected to the VT macro. The RISC core operates at 40 MHz at 1.9 V, and at 10 MHz at 1.3 V.

Figure 4.11 shows a measured power dissipation of the RISC core without I/O. White circles and black squares in this figure represent power dissipation at 3.3 V and  $V_{DDL}$  determined by the VS scheme, respectively. Power dissipation is reduced by an



**Figure 4.10** Chip microphotograph of 32-bit RISC core processor, TX3900, with VS scheme and VTCMOS technology.

**TABLE 4.2** TX3900 Features

Technology	0.4 $\mu$ m CMOS, n-well/p-sub, double-metal, $V_{TH} = 0.05$ V (controlled to 0.2 V)
External $V_{DD}$	3.3 V (generate internally 0.8–2.9 V)
Clock frequency	40 MHz
Power dissipation	140 mW
Standby current	10 $\mu$ A
Chip size	8.0 mm $\times$ 8.0 mm
Macro size	0.49 mm $\times$ 0.72 mm (VT), 0.45 mm $\times$ 0.59 mm (VS)

amount larger than that which can be achieved by reducing clock frequency. The power dissipation at  $f_{ext} = 0$  is about 20 mW, which comes from the DC-DC converter. The DC-DC converter efficiency was 85% in this design, which is now raised to 95% by advanced circuit techniques [14].

Measured performance in MIPS/W is 320 MIPS/W at 33 MHz, and 480 MIPS/W at 20 MHz, which is improved by a factor of more than 2 compared with that of the conventional CMOS design, 150 MIPS/W.

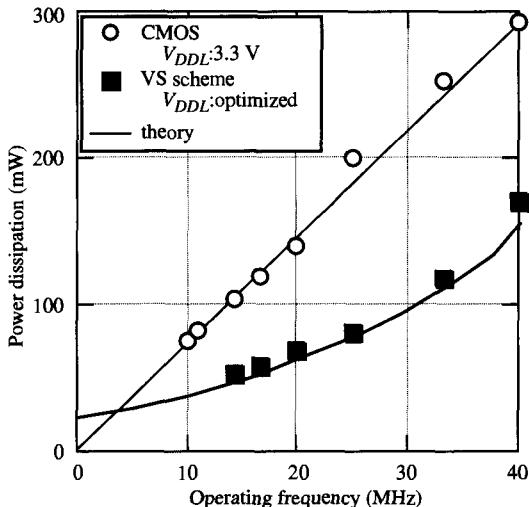


Figure 4.11 Measured power dissipation vs. operating frequency.

#### 4.4.2 SH-4

A 200 MHz SH-4 microprocessor is designed using a switched substrate-impedance scheme [15]. It offers a battery backup capability in a self substrate-biased data retention mode, in which it consumes only  $17.8 \mu\text{A}$  from a  $1.0 \text{ V}$  supply. Main features are summarized in Table 4.3.

TABLE 4.3 SH-4 Features

Technology	0.2 $\mu\text{m}$ CMOS, n-well/p-sub, triple-metal, $V_{TH} = 0.15 \text{ V}, 0.45 \text{ V}$
External $V_{DD}$	3.3 V, 1.8 V, 1.0 V (data retention mode)
Clock frequency	200 MHz
Power dissipation	1.0 W
Standby current	$46.5 \mu\text{A}$ (standby mode), $17.85 \mu\text{A}$ (data retention mode)
Chip size	$6.84 \text{ mm} \times 6.84 \text{ mm}$
Macro size	$0.21 \text{ mm} \times 0.65 \text{ mm}$ (VBC)

The switched substrate-impedance scheme is basically the same as the SPR w/ SSB scheme. The scheme is illustrated in Fig. 4.12. The substrates are connected to power/ground lines in the active mode by the switch transistors, which correspond to the  $M_1$  transistor in the SPR w/ SSB scheme. About 10,000 switch cells are distributed over the chip so that substrate impedance is low enough for high-speed circuit operation in the active mode. Two supply voltages are required:  $3.3 \text{ V}$  for mainly  $n$ -well bias and  $1.8 \text{ V}$  for internal power supply. Substrate bias of  $-1.5 \text{ V}$  is applied to PMOS transistors. The same amount of substrate bias for NMOS transistors is generated by SSB.

In the data retention mode, the internal power supply is lowered to  $1.0 \text{ V}$ , as illustrated in Fig. 4.13, which reduces not only the subthreshold leakage current by deeper substrate bias ( $-2.3 \text{ V}$ ), but also gate-induced drain leakage current by lower drain-gate voltage ( $1.0 \text{ V}$ ). Chip leakage current is reduced from  $1.3 \text{ mA}$  in the active mode to  $46.5 \mu\text{A}$  in the standby mode, and is further reduced to  $17.8 \mu\text{A}$  in the data retention mode.

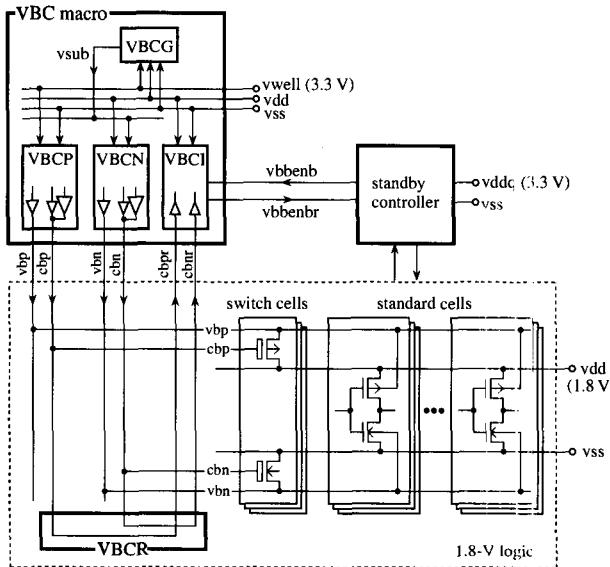


Figure 4.12 Switched substrate-impedance scheme.

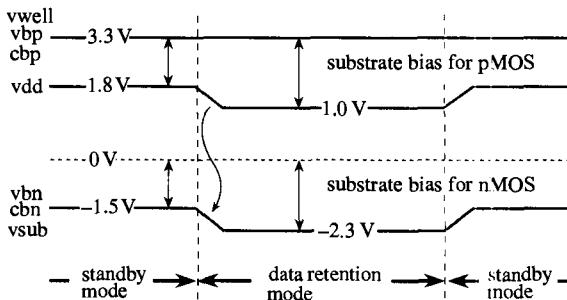


Figure 4.13 Substrate bias at standby and data retention modes.

## 4.5 CONCLUSION

Low-power, high-speed CMOS circuit design by means of  $V_{DD}$ ,  $V_{TH}$  control is presented. Low- $V_{DD}$ , low- $V_{TH}$  design space is explored. Design considerations in power-down schemes are discussed. Issues in sleep transistor sizing in MTCMOS are described, and its circuit modification for low-voltage operation is introduced.

The VTCMOS technology is ready for practical use. Design issues are discussed in depth. VTCMOS compensates for die-to-die  $V_{TH}$  variation within  $\pm 0.05$  V and raises  $V_{TH}$  by 0.25 V to reduce the subthreshold leakage current below 10  $\mu$ A in the standby mode. The impact of using VTCMOS on the short-channel effect and the within-die  $V_{TH}$  variation is negligible. VTCMOS temperature dependence can be controlled by the LCM circuit design. Power and area penalty values in VTCMOS become less than 1% and 6%, respectively. Circuit performance fluctuation, due to the substrate noise, is too small to be measured. Two design examples, TX3900 and SH-4, are presented, which demonstrate practical use of the  $V_{TH}$  control through substrate bias.

It is essential to optimize  $V_{DD}$  and  $V_{TH}$  for low-power, high-speed CMOS design, while they have been given to designers as fixed parameters in the conventional CMOS

design. Circuit designers can now control them as objectives of design optimization. As a result, active power can be reduced to below half, while static power and chip throughput are maintained.

## REFERENCES

- [1] T. Kuroda and T. Sakurai, "Overview of Low-power ULSI Circuit Techniques," *IEICE Trans. Electron.*, vol. E78-C, no. 4, pp. 334–344, April 1995.
- [2] A. Chandrakasan and R. Brodersen, *Low-Power CMOS Design*. IEEE Press, Piscataway, NJ, 1998.
- [3] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V Power Supply High-speed Digital Circuit Technology with Multithreshold-voltage CMOS," *IEEE J. Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, Aug. 1995.
- [4] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor Sizing Issues and Tool for Multi-Threshold CMOS Technology," *Proc. Design Automation Conference '97*, pp. 409–414, June 1997.
- [5] H. Kawaguchi, K. Nose, and T. Sakurai, "A CMOS Scheme for 0.5 V Supply Voltage with Pico-ampere Standby Current," *ISSCC Dig. Tech. Papers*, pp. 192–193, Feb. 1998.
- [6] T. Kuroda, T. Fujita, S. Mita, T. Nagamatu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai, "A 0.9 V 150 MHz 10 mW 4 mm<sup>2</sup> 2-D Discrete Cosine Transform Core Processor with Variable Threshold-voltage (VT) Scheme," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1770–1779, Nov. 1996.
- [7] T. Kuroda and T. Sakurai, "Threshold-voltage Control Schemes Through Substrate-Bias for Low-power High-speed CMOS LSI Design," *J. VLSI Signal Processing Systems*, vol. 13, no. 2/3, pp. 191–201, Aug. 1996.
- [8] T. Kuroda, T. Fujita, T. Nagamatu, S. Yoshioka, T. Sei, K. Matsuo, Y. Hamura, T. Mori, M. Murota, M. Kakumu, and T. Sakurai, "A High-speed Low-power 0.3 μm CMOS Gate Array with Variable Threshold Voltage (VT) Scheme," *Proc. CICC'96*, pp. 53–56, May 1996.
- [9] M. Takahashi, M. Hamada, T. Nishikawa, H. Arakida, Y. Tsuboi, T. Fujita, F. Hatori, S. Mita, K. Suzuki, A. Chiba, T. Terasawa, F. Sano, Y. Watanabe, H. Momose, K. Usami, M. Igarashi, T. Ishikawa, M. Kanazawa, T. Kuroda, and T. Furuyama, "A 60 mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-voltage Scheme," *ISSCC Dig. Tech. Papers*, pp. 34–35, Feb. 1998.
- [10] T. Kuroda, T. Fujita, S. Mita, T. Mori, K. Matsuo, M. Kakumu, and T. Sakurai, "Substrate Noise Influence on Circuit Performance in Variable Threshold-voltage Scheme," *Proc. ISLPED'96*, pp. 309–312, Aug. 1996.
- [11] K. Kanda, K. Nose, H. Kawaguchi, and T. Sakurai, "Design Impact of Positive Temperature Dependence of Drain Current in Sub 1 V CMOS VLSI's," *Proc. CICC'99*, pp. 563–566, May 1999.
- [12] S. Narendra, D. Antoniadis, and V. De, "Impact of Using Adaptive Body Bias to Compensate Die-to-die  $V_t$  Variation on Within-die  $V_t$  Variation," *Proc. ISLPED'99*, pp. 229–232, Aug. 1999.
- [13] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama, "Variable Supply-voltage Scheme for Low-power High-speed CMOS Digital Design," *IEEE J. Solid-State Circuits*, vol. 33, no. 3, pp. 454–462, March 1998.
- [14] F. Ichiba, K. Suzuki, S. Mita, T. Kuroda, and T. Furuyama, "Variable Supply-voltage Scheme with 95%-efficiency DC-DC Converter for MPEG-4 codec," *Proc. ISLPED'99*, Aug. 1999.
- [15] H. Mizuno, K. Ishibashi, T. Shimura, T. Hattori, S. Narita, K. Shiozawa, S. Ikeda, and K. Uchiyama, "A 18 μA-standby-current 1.8 V 200 MHz Microprocessor with Self Substrate-biased Data-retention Mode," *ISSCC Dig. Tech. Papers*, pp. 280–281, Feb. 1999.

Chapter  
**5**

# SOI TECHNOLOGY AND CIRCUITS

Ghavam G. Shahidi, Fari Assaderaghi

*IBM*

Dimitri Antoniadis

*Massachusetts Institute of Technology*

## 5.1 INTRODUCTION

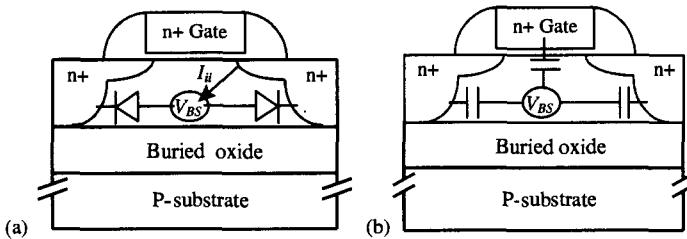
Deep submicron room-temperature bulk CMOS is the main technology for ULSI systems. CMOS scaling, which has been the main method for improving system performance, is rapidly approaching its limits. Use of SOI for improving CMOS performance has been under consideration for nearly three decades [1]. The main challenges to the acceptance of CMOS on SOI have been: (1) Material quality; (2) SOI device design; (3) bulk CMOS scaling and the resulting performance gain per generation; and (4) the lack of demonstration of a mainstream complex application implemented in SOI. The restriction of SOI nFET burn-in voltage to low values also posed a serious, and less talked about, barrier to the use of SOI: Before the  $0.25\text{ }\mu\text{m}$  CMOS generation the supply voltage was 3.3 V and above, while in SOI nFETs with  $L < 0.5\text{ }\mu\text{m}$  the breakdown voltage is about 4 V.

Over the last few years, significant progress has been made in all of the above areas. SOI material quality has improved significantly, and now there is a much better understanding of SOI defects, their characterization and their impact on the devices and circuits. The adoption of the partially depleted (PD) SOI device design was also a major departure from early prevailing thinking about SOI device design [2] and it has significantly simplified device design, manufacturing, and circuit design with SOI (as compared to fully depleted (FD) SOI device design). In fact, as discussed later, the “partially depleted” condition is an important source of SOI performance gain over bulk CMOS. The adoption of PD SOI devices introduces a few new circuit behaviors that do not exist in bulk CMOS technology. These are the so-called floating-body effects, namely, “kink effect,” “history dependence” of propagation delay, excess transient “pass-gate” leakage current. Also, SOI FETs exhibit enhanced “self-heating” under some operating conditions. A key breakthrough in SOI technology development was the ability to design circuits in the presence of the floating-body effects. These effects had to be accurately measured and modeled; their consequence in circuits had to be well understood; and, when needed, design techniques were deployed to minimize them.

SOI technology is also being considered for low-power applications. In the last section, aspects of the SOI technology which make it attractive for low-power applications are discussed.

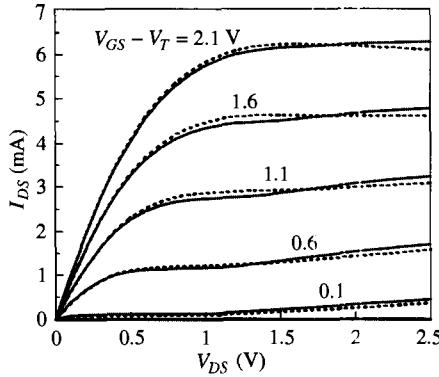
## 5.2 DEVICE DESIGN CONSIDERATIONS FOR PD SOI VERSUS FD SOI

Figure 5.1(a) shows the cross section of an NMOS transistor on SOI. The main feature of MOS in SOI is that the device's local substrate ("body") is electrically floating, and therefore its voltage,  $V_{BS}$ , is not fixed. Under static-steady-state conditions  $V_{BS}$  is determined by the balance of DC currents through the two back-to-back diodes and impact ionization near the drain. Impact ionization is caused by channel carriers energized as they travel through the high field region near the drain in saturation, and results in the creation of hole-electron pairs. Under normal device operation the minority carriers of those pairs (e.g., electrons for nFETs) are swept to the drain along with the channel carriers, causing no noticeable effect. On the other hand, the majority carriers (e.g., holes in nFETs) are swept to the substrate giving rise to the well-known substrate current which is shown in Fig. 5.1(a) by the arrow,  $I_{hi}$ . Under dynamic switching conditions  $V_{BS}$  depends on the previous electrical switching "history" of the device, as well as on the instantaneous node voltages through the capacitive network shown in Fig. 5.1(b). This dynamic modulation of  $V_{BS}$  (and  $V_T$ ) in SOI FETs is discussed in more detail in Section 5.4.2.

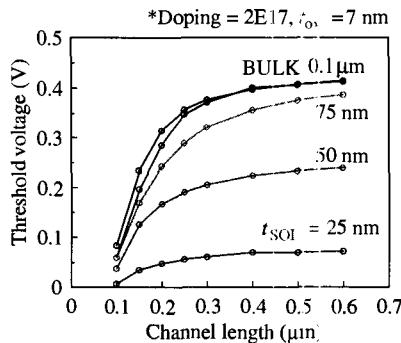


**Figure 5.1** Schematic cross section of a PD SOI nFET with floating body. (a) DC circuit elements: The "undepleted" body is DC-connected to the source and drain terminals only through the diodes and the impact ionization (I/I) "source" near the drain. The latter results in injection of majority carriers to the substrate (holes for nFETs) and minority carriers to the drain (electrons for nFETs). (b) AC circuit elements: The body is connected to the device terminal via nonlinear capacitances. The much smaller capacitance to the p-substrate has been omitted for clarity.

The changing  $V_{BS}$  can modulate  $V_T$  if it is large enough. An example of modulation of  $V_T$ , under static conditions, is the "kink effect" in the output I-V characteristics shown in Fig. 5.2 (solid lines) as the increase of the output conductance of the device near  $V_{DS} = 1$  V. This is caused by the impact-ionization-induced increase in  $V_{BS}$  with increasing  $V_{DS}$ , and the resulting reduction of  $V_T$ ; when  $V_{DS}$  becomes large enough impact ionization current (holes) flow to the undepleted body increasing the body charge and therefore  $V_{BS}$ . One method widely held to minimize the floating body effects is to use fully depleted SOI devices. In FD devices, the SOI film thickness is smaller than the minimum channel depletion width, and therefore the body charge is fixed. Any impact ionization charges (majority carriers) flowing into the depleted body are readily swept to the source because of the much reduced potential barrier. The "kink effect" disappears in FD devices, as shown in Fig. 5.2 (dashed lines). For this reason in early



**Figure 5.2** Example output I-V characteristics for SOI nFET PD (solid lines) and FD (dashed lines).



**Figure 5.3** Threshold voltage versus channel length for nFETs in bulk CMOS and SOI of various Si film thickness.

stages of the SOI technology development the focus was on FD SOI devices, since the kink effect (and other floating-body effects such as dynamic  $V_T$  variation) were considered seriously problematic. On the other hand, dynamic  $V_T$  variation is a significant cause of the performance boost of digital circuits in SOI because of switching current increase, and is only present in PD SOI. With the adoption of PD SOI other methods have been developed to eliminate the floating-body effects in the special cases (e.g., analog circuits) where they should not be present.

A perceived benefit of SOI technology (and in particular FD SOI) has been the improvement in short-channel effects (SCE) [1]. At first sight this would appear to be so: Fig. 5.3 shows simulated  $V_T$  roll-off of a bulk-Si (top curve) and four SOI nFETs with different film thickness. As the SOI film thickness is reduced, the  $V_T$  roll-off does improve but the improvement is only caused by the reduction in junction depth. In fact, for a given channel length FD SOI devices exhibit increased SCE relative to PD SOI unless the silicon film thickness becomes much smaller than the depletion depth [3]. For example, for 130 nm, SOI silicon thickness below 20 nm is required to match the CMOS *drain-induced barrier lowering* (DIBL) of 100-nm film-thickness PD devices [4]. Indeed, adopting PD device design, one can use many of the techniques developed for controlling the SCE in bulk CMOS, i.e., retrograde well, halo, and S/D shallow extension [2]. Also, in PD SOI, because in general  $V_{BS} > 0$  the SCE is decreased even compared to a bulk technology with identical doping configuration.

There are many other convincing benefits associated with PD SOI: It is easier to manufacture because the SOI film is thicker (150 nm compared to required FD SOI

thickness much less than 50 nm). A high  $V_T$  can be achieved easily in PD SOI, whereas it is difficult to obtain a high  $V_T$  in FD SOI. If the film doping is increased to increase the  $V_T$ , the device becomes PD; if the film thickness is reduced to make it FD, then  $V_T$  is reduced. Deep-submicron CMOS technologies use multiple  $V_T$  values to balance performance against noise and power. It is relatively simple to have multiple- $V_T$  offering with PD SOI technology, but not so with FD SOI. All floating-body effects are actually present to some degree in FD devices unless the film thickness is much smaller than the natural channel depletion depth. For example, the excess transient “pass-gate” leakage is actually higher in FD devices until this condition is met. The “history-dependence” of propagation delay is larger in PD SOI, but as will be shown later, this is a manageable effect in most circuits. And, for those cases where floating-body effects must be completely eliminated, it is possible to form an electrical body contact in PD SOI but not so in FD. In summary, there are compelling reasons why the PD device design is preferable.

## 5.3 DEVICE RESULTS

As the mainstream bulk CMOS supply voltage has dropped below 1.8 V, uncompromised SOI CMOS technology has become feasible (including high-voltage defect-screening methodology as part of test) because floating-body effects, which are proportional to supply voltage, have been reduced. Initially, an SOI CMOS with 0.18  $\mu\text{m}$  gate lithography and aluminum interconnect was developed at IBM [5; “0.25  $\mu\text{m}$ ” in the title refers to baseline lithography], followed by a 0.22  $\mu\text{m}$  CMOS SOI with copper interconnect [6].

Proper characterization of SOI CMOS devices requires good understanding of SOI-specific effects that make their behavior different from bulk CMOS. These are: (1) Self-heating, and (2)  $V_T$  variation due to the floating body. Indeed extraction of proper device model parameters for circuit simulation is one of the challenges in SOI CMOS adoption. We discuss these effects in the following two subsections.

### 5.3.1 SOI Device Self-Heating

The presence of the buried oxide in SOI causes reduction of the thermal conductance to the substrate. The decreased thermal conductance can result in increase of the device temperature, with corresponding reduction in current and possible impact on reliability. One of the most obvious manifestations of this current reduction is the evidence of negative differential output resistance which appears when it is not masked by the kink effect, or by strong DIBL. This negative resistance can be seen clearly in the “kink-less” FD-SOI device characteristics in Fig. 5.2.

To put self-heating of SOI devices in perspective it is important to consider some numbers: The 1-D heat resistivity, due to the silicon substrate alone, of a conventional chip (thinned to fit the package to approx. 300  $\mu\text{m}$ ) is  $20 \text{ mK}/(\text{W/cm}^2)$ . So even for a hypothetical MPU with very high power dissipation of  $100 \text{ W/cm}^2$  the temperature rise from the front to the back of the silicon chip is only 2 K. Such a chip would have an expected device operational temperature of about 100°C, i.e. a temperature rise of approximately  $\Delta T = 75 \text{ K}$  above ambient. Evidently, 97% of this comes from the heat resistance of the substrate bond and the package with only 3% contributed by the silicon substrate itself. Consider now a conventional SOI with buried oxide thickness of 200 nm. If we assume that the power dissipation is uniform across the entire chip surface, we still are solving a 1-D heat flow problem. The insulator (oxide) has a specific heat resistance

100 times that of undoped silicon, so the 200 nm of oxide thermally corresponds to 20  $\mu\text{m}$  of Si. So for 1-D heat flow our 300  $\mu\text{m}$  chip now looks like a 320  $\mu\text{m}$  chip and its 1-D heat resistivity is 21.3 mK/(W/cm<sup>2</sup>). This 7% increase in thermal resistance is insignificant.

For individual SOI devices, as is the case for devices under test, the heat flow to the substrate is three dimensional. Approximately, the thermal resistance to the substrate, normalized to device width, is 60–100  $\mu\text{mK}/\text{mW}$  [7]. This 3-D heat flow resistivity is about 4–5 times that of isolated bulk-Si devices. In 0.22  $\mu\text{m}$  CMOS at 1.8 V, a continuously “on” nFET has a current near 600  $\mu\text{A}/\mu\text{m}$ , dissipating about 1 mW/ $\mu\text{m}$  and resulting in a 60–100 K temperature rise. Its bulk-Si counterpart would have a 12–20 K temperature rise. This increased SOI temperature rise needs to be compensated more accurately during parameter extraction.

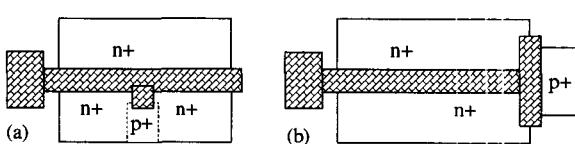
At first sight this 4–5  $\times$  3-D heat resistance increase looks bad. One would be tempted to say that individual SOI devices would be about 4–5  $\times$  hotter in an SOI microprocessor than in its bulk counterpart. The reason this is not so is that small devices are not isolated in an IC, so it is 1-D heat flow that dominates rather than 3-D heat flow. This happens because the 3-D heat resistance of isolated devices is 5–10  $\times$  smaller than the 1-D heat resistance corresponding to their area. So 1-D heat flow, which as we saw has very similar heat resistance in SOI and bulk-Si, is the limiting resistance when devices are densely laid out.

Additionally, in a switching circuit, the device conducts current only during falling (or rising) drain voltage transients. Furthermore, the device does not see the full  $V_{DD}$  throughout the transient. The power per unit width of the device, even in the clock drivers or highly loaded stages, is much less than the static 1 mW/ $\mu\text{m}$ . The temperature rise in most SOI devices is proportionally small, about 2–3 K. In terminated output drivers, where the device actually drives a resistive load and conducts current for a considerable fraction of the cycle, the temperature increase is about 10–15 K.

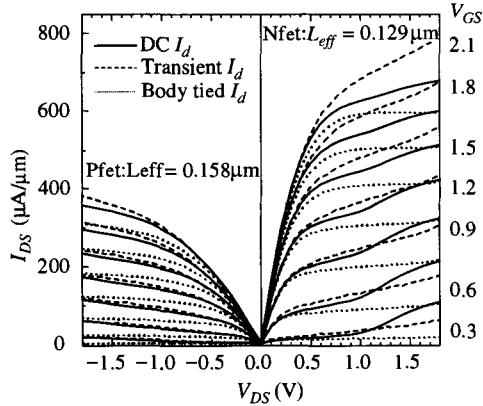
In conclusion the main effect of self-heating in SOI devices is that it complicates device characterization and needs to be compensated when extracting compact-model device parameters from typical isolated test devices under DC conditions.

### 5.3.2 $V_T$ Variation Due to Floating Body

One of the challenges of PD SOI CMOS technology is understanding, characterization, modeling, and circuit design in the presence of floating body effects. Figure 5.4 shows the circuit elements determining the  $V_{BS}$  of a SOI FET. The body voltage is determined by the leakage current of the p-n diodes and the impact ionization current [Fig. 5.4(a)], and by the capacitive coupling to device terminals during switching [Fig. 5.4(b)]. The response time of the leakage and impact ionization currents is slow (order of ms) while the response time of capacitive coupling is fast (order of ps). As in bulk-Si MOSFETs, the change in  $V_{BS}$  modulates  $V_T$ . In PD SOI,  $V_{BS} > 0$  (in nFETs, opposite in pFETs) in most cases. This is in contrast to bulk-Si where  $V_{BS} \leq 0$  all the time. If it is desired to avoid the floating-body effects at all cost, direct contact can be



**Figure 5.4** Simplified schematics of source-tied (a), and body-tied (b) nFETs. The p+ body in (b) can be connected to any voltage, while the source tied in (a) ensures  $V_{BS} = 0$ . In both cases the effectiveness of the tie is limited by the resistance of the undepleted body, which limits the maximum width per tie.



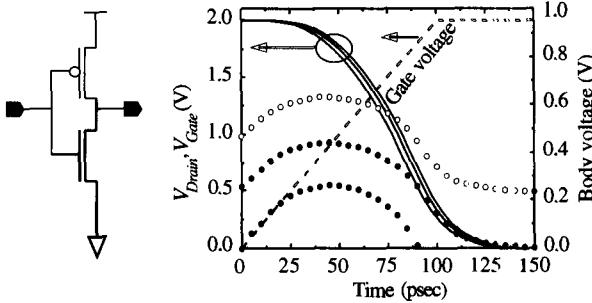
**Figure 5.5** Measured output I-V characteristics showing DC (solid), body-tied- $V_{BS} = 0$  (short-dash) and transient (long-dash) modes.

made to the MOSFET body (local substrate), either grounding it or shorting it to the source ( $V_{BS} = 0$ ) [8], using the “body-contact” device structures shown in Fig. 5.4.

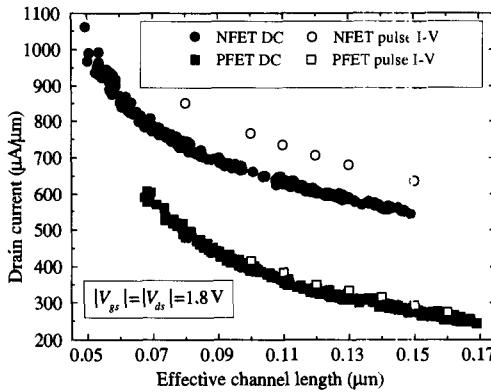
The floating-body as well as the self-heating effect on nFET/pFET device characteristics are illustrated in Fig. 5.5. The solid lines represent DC floating-body measurements. Since DC measurements are relatively slow,  $V_{BS}$  is constantly at the steady-state value (positive/negative for nFET/pFET, respectively) dictated by the DC circuit in Fig. 5.1(a). The kink is clearly visible particularly at the lower  $V_{GS}$  values. The short-dash curves represent DC  $V_{BS} = 0$  measurements. The kink has now disappeared, and at the high  $V_{GS}$  values in the nFET a slight negative output conductance, which is typically masked by the kink, has appeared. The negative output conductance is the signature of the self-heating effect described earlier and is due to decrease of carrier velocity with increasing temperature. Under DC conditions the device temperature is in steady state with respect to the input power because the thermal time constant is of order  $\mu\text{s}$  [9].

Since in a digital circuit the devices are switching on order of tens of ps,  $V_{BS}$  will not be in instantaneous steady state with either self-heating or diode leakage and impact ionization currents. The I-V characteristics that are relevant for digital applications are shown by the long-dash lines in Fig. 5.5. These measurements were obtained under transient conditions with ns-rise-time  $V_{GS}$  pulses from 0 V to the desired measurement value [10]. Now the instantaneous value of  $V_{BS}$  is determined by the capacitive coupling in the AC circuit of Fig. 5.1(b), and by the initial condition of  $V_{BS}$  that in turn is determined by the steady state of the DC circuit of Fig. 5.1(a) for  $V_{GS} = 0$  and the  $V_{DS}$  value of each measurement point.

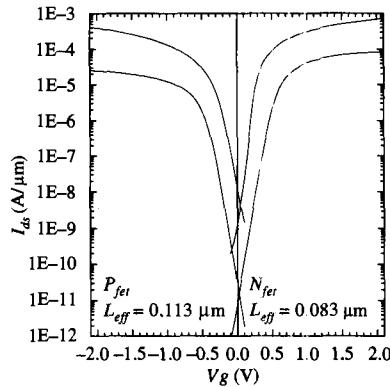
The capacitive coupling of the gate to the body in PD-SOI devices is actually a beneficial effect. This is because during a switching transient, e.g., in nFET, the rising  $V_{GS}$  induces a transient rise in the  $V_{BS}$ —effectively a transient reduction in  $V_T$  which boosts the switching current and reduces propagation delay. This is illustrated in Fig. 5.6 which shows simulated  $V_{BS}$  (for three different initial conditions) and  $V_{DS}$  voltages for a rising  $V_{GS}$  transient in a CMOS inverter [11]. All three cases benefit from the transient rise in  $V_{BS}$ , and as expected, the propagation delay is decreasing as the initial  $V_{BS}$  value is increased. A further illustration of this beneficial effect is shown in Fig. 5.7 where the DC and transient saturation drain currents are shown for nFETs and pFETs. The increase in current is due to both the capacitive coupling and the elimination of the self-heating effect. Equivalent technology bulk CMOS DC currents are close to the SOI currents, but transient SOI device “on” currents are 10–15% higher.



**Figure 5.6** Simulated output voltage and nFET  $V_{BS}$  of a PD-SOI inverter undergoing pull-down. Three different initial  $V_{BS}$  values were used. The higher the value the shorter the pull-down delay [after 11].



**Figure 5.7** Measured DC and transient drain current of PD-SOI devices.



**Figure 5.8** Measured transfer I-V characteristics of PD-SOI devices.

Figure 5.8 shows device DC subthreshold currents [6]. Because floating-body  $V_{BS}$  increases in the forward direction with increasing  $|V_{DS}|$ , PD-SOI MOSFETs have larger drain-induced barrier lowering (DIBL) and lower saturation  $|V_T|$  (at nominal channel lengths) than their bulk CMOS counterparts. However, with proper device design, and at the elevated temperature of operation in high-performance MPUs this difference is negligible [12].

In summary, floating-body and transient  $V_{BS}$  effects are unique features of SOI that require good understanding for SOI device design and modeling, or for comparison with equivalent bulk CMOS devices.

## 5.4 PD-SOI CMOS DIGITAL CIRCUITS

Circuits in SOI operate faster than identical circuits in a bulk CMOS technology of the same lithography and device characteristics such as  $I_{D OFF}$  at operating conditions. There are three primary causes of performance gain of SOI over bulk CMOS: (1) area and exterior periphery junction capacitance is nearly absent in SOI; (2) the classic “MOS reverse-body-voltage effect” is absent in SOI, because the body-source diode is never reverse biased (as is the case of stacked transistors and pass-gates in bulk CMOS)—indeed in SOI, this bias is typically in the forward direction as already discussed above; and (3) because of forward  $V_{BS}$  in most switching cases, even when no stacked transistors are involved, SOI “on” current during the switching transient is higher than in bulk CMOS. Since one does not have to worry about “reverse-body effect” in SOI, this opens up a new device optimization space, in which one can decrease some of the inherent internal device capacitances, such as the gate-drain overlap capacitance. Decreasing the gate-drain overlap capacitance can have a significant benefit, because it reduces the Miller effect during switching.

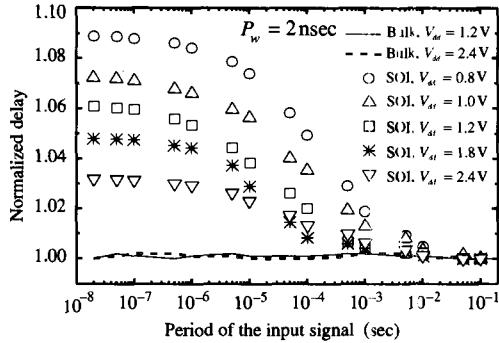
Unloaded inverters in SOI are 20–25% faster than their bulk counterparts (higher “on” current and smaller junction capacitance). As the number of stacked nFETs increases from two-, to four-input NAND gates the performance gain in SOI increases from 27% to 50% because the floating body in SOI prevents  $V_T$  increase as the number of series nFET transistors increases. Table 5.1 lists results of simulation of critical paths of a number of microprocessors, using the device models for 0.22  $\mu\text{m}$  CMOS-SOI. References [5] and [6] describe microprocessors built using this SOI technology. It is clear that the floating-body voltage is beneficial in SOI CMOS circuit operation. However, it does give rise to two SOI-unique effects that must be well understood and modeled when designing in SOI or migrating a design from bulk CMOS. These are the “history dependence” of propagation delay through a gate, and the transient excess leakage through pass-gate configured devices. These two effects are discussed next in some detail.

**TABLE 5.1** Simulation of Critical Paths of CPUs

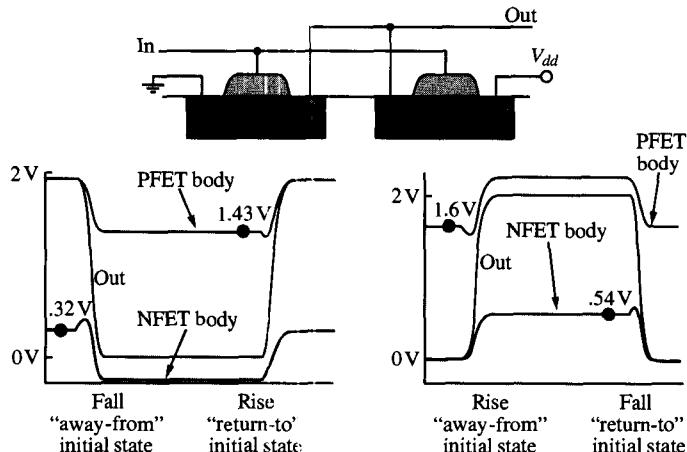
Circuit	$(t_{BULK} - t_{SOI})/t_{BULK}$
CPU1 L2 Directory Access time	29.8%
CPU2 (Gate dominated)	27.1%
CPU2 (Wire dominated)	20.9%
CPU3 (Gate dominated)	30.4%
CPU3 (Wire dominated)	19.6%

### 5.4.1 History Dependence of Propagation Delay

History dependence in SOI refers to uncertainty or variation in delay through a gate. This is also referred to as “hysteretic” gate-delay. The basic cause of the hysteretic delay is the variability of the  $V_{BS}$ , and the subsequent variation in  $V_T$ , as illustrated in Fig. 5.6 and discussed further in Section 5.4.2. As the initial  $V_{BS}$  is increased, the nFET pre-switch  $V_T$  decreases and the switching time is shorter. In actual circuits, the initial condition of  $V_{BS}$  is set by the switching history of the device, so the history dependence is the change in delay through a gate as a function of the switching history of the device. Figure 5.9 shows experimentally measured fall propagation delay through an even



**Figure 5.9** Measured normalized delay through a chain of gates versus input pulse repetition rate for various power supply voltages [13].



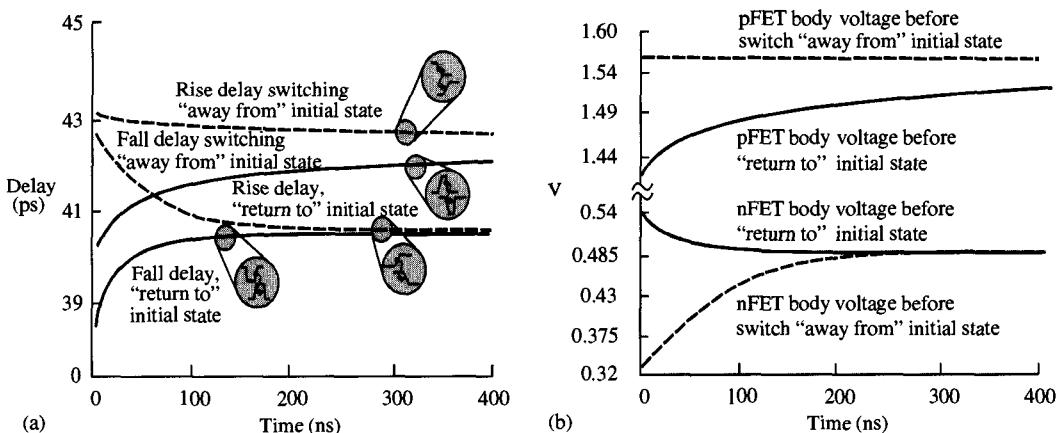
**Figure 5.10** Simulated output voltage and nFET/pFET  $V_{BS}$  for the first two transitions of a PD SOI inverter starting from output-high or output-low static steady state. The dots mark the pre-switch  $V_{BS}$  of the device responsible for the corresponding transition [after 14].

number of NAND gates at room temperature. When the driving pulse period is slow (10 ms) the delay is shorter than when the period is fast (1  $\mu$ s). It can also be seen that the effect increases as the  $V_{DD}$  voltage decreases because the proportion of floating-body-dependent  $V_T$  variation increases. It is important that at each technology node, as  $V_{DD}$  is scaled down, the device design should be optimized to minimize this effect.

During switching the  $V_{BS}$  of SOI devices undergoes considerable variation. Figure 5.10 shows simulated  $V_{BS}$  in the nFET and pFET of an inverter during switching starting from two initial states: output-high and output-low [14]. In both cases, it is assumed that the inverter was idling at the initial state for a very long time. The values of  $V_{BS}$  that determine the  $V_T$  of the device doing the relevant pull-down or pull-up switching are shown with a dot. As can be seen in all cases the  $V_{BS}$  is in the forward direction leading to reduced  $|V_T|$  and therefore increased switching current. These pre-switching  $V_{BS}$  values for the first transition (away-from initial state) are determined by the static steady-state balance of DC currents of Fig. 5.1(a) alone, and by that plus the capacitive network of Fig. 5.1(b) for the second transition ("return-to" initial state). As the devices undergo fast switching starting from idle, the pre-switching  $V_{BS}$  values

change slowly from their first post-static values toward their switching-steady-state values as shown in Fig. 5.11(a). This happens in order to establish body-voltage conditions such that over one cycle the net current into the body, due to the combined elements of Fig. 5.1, is zero in switching steady state. The reason why it takes a long time, and a large number of cycles to reach steady state, is because the body net charge must be changed from initial static- to switching-steady-state value but the diode leakage and impact ionization currents, which provide the only means of changing the body charge, are very small. The result of this slow change in pre-switch  $V_{BS}$  is that the inverter rise times change slowly over time and are different in steady state from what they are during the first switch from static state, as shown in Fig. 5.11(b). Note that over the number of cycles in the 400 ns simulation interval the two nFET pre-switch  $V_{BS}$  values have converged to their common switching-steady-state value (and so do the pull-down delays), but the pFETs are not yet in switching steady state—their transient being significantly slower.

Table 5.2 lists the maximum variation in delay observed experimentally for a number of gates between static- and switching steady state.



**Figure 5.11** (a) Simulated dependence of the relevant sampled pre-switch  $V_{BS}$  (Fig. 5.10) versus time in the course of many transitions starting from static steady state and tending toward dynamic switching steady state. (b) Simulated dependence of rise and fall delays, corresponding to Fig. 5.10, versus time in the course of many transitions starting from static steady state and tending toward dynamic switching steady state. Note how the time dependence here reflects the  $V_{BS}$  time dependence of part (a) [after 14].

**TABLE 5.2** Variations in Delay

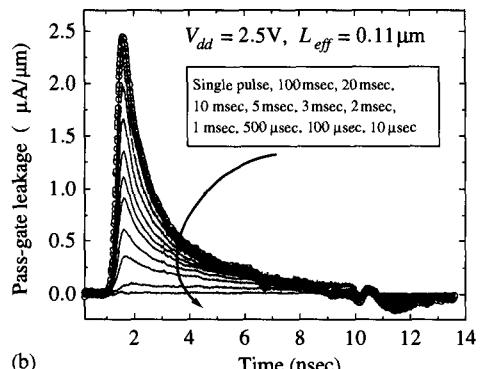
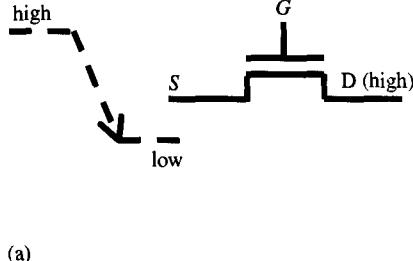
Circuit	Change in delay (%)
Inverter	5%
NAND2, bot. device	3%
NAND2, top device	5%
NAND4, bot. device	4%
NAND4, top device	7%
Inv. + 2 mm wire	3%

The uncertainty in the delays in Table 5.2 may seem excessive. However, there is considerable uncertainty in the delay through a gate in a bulk CMOS design also. Among the phenomena contributing to uncertainty in a bulk-Si design are: intra-chip process variation (about 15–20% of a gate delay); delay variation caused by top- versus bottom-switching device in NAND gates, or the number of simultaneous “1”s arriving at a NOR gate (about 20–35%); on-chip  $V_{DD}$  variations (about 10%); and temperature variations (on-chip and ambient, which can cause 10–20% change in delay). In addition, interconnect line lateral capacitive coupling can contribute as much as  $3 \times$  uncertainty in long line propagation delay [15]. The SOI-induced uncertainty in delay, while not negligible, is generally smaller than the other sources of uncertainty in a chip, and can be handled in a similar fashion [16].

### 5.4.2 “Pass-Gate” Transient Leakage

The other SOI-unique effect is pass-gate leakage [17]–[19] which is most significant in nFETs. In SOI if the source and drain are both high ( $V_{DD}$ ) for a relatively long interval (i.e., tens of ms) the body charges up, through diode leakage, all the way to  $V_{DD}$ . Moreover, if the gate is “off,” i.e., at 0 V, then the channel accumulates holes. Now if the source is pulled low [Fig. 5.12(a)], a transient forward  $V_{BS}$  voltage appears across the source-body diode, with initial magnitude that depends on  $V_{DD}$ , the capacitance values of the capacitive network of Fig. 5.1(b), and the accumulated hole charge, i.e., idle time prior to the source pull-down. This forward voltage can turn on the parasitic source-body-drain npn bipolar transistor of the nFET and will give rise to a pulse of current in the device, even though the gate is “off” [Fig. 5.12(b)]. The magnitude of this pulse depends on the initial value of  $V_{BS}$  and the parasitic BJT current gain, and can lead to the discharge of dynamic circuit nodes with no weak or no keeper devices as discussed in the next section. Note that this is also a “history-dependent” effect in that the magnitude of the current depends strongly on the device idle time. In constantly switched devices there is no transient leakage [Fig. 5.12(b)].

There are two ways to minimize this transient leakage: First, minimize the BJT gain, and second, minimize the initial transient  $V_{BS}$  value. The first is achieved through proper source-body diode doping design that makes carrier (hole) injection into the



**Figure 5.12** Transient pass-gate leakage set-up (a); and measured transient current versus source pull-down pulse repetition rate for a PD SOI nFET (b) [20]. Note that the effect disappears for devices switching with shorter than 10  $\mu$ s rates.

source very efficient. The second is naturally accomplished as  $V_{DD}$  scales down with advancing CMOS generations.  $V_{BS}$  can also be reduced further by proper balancing of the source-body junction versus gate-body capacitance for minimum-size devices. Indeed it is observed that the transient leakage current does peak at minimum channel length longer than devices because with increasing length the gate capacitance tends to hold the body voltage high during the initial transient  $V_{BS}$  while the BJT gain decreases.

It should also be pointed out that besides the transient BJT leakage there also exists transient MOSFET leakage, caused by the spike in  $V_{BS}$  and the attendant transient  $V_T$  reduction. While in typical- $V_T$  devices the MOSFET current is very small compared to the BJT, in low- $V_T$  devices it can be the dominant effect.

It is clear from this discussion that modeling the transient leakage current requires SOI-specific device characterization and compact models. Indeed this effect is the one that requires the most design attention in migrating circuits from bulk CMOS to SOI CMOS.

### 5.4.3 Circuit Issues in PD-SOI CMOS

A major part PD-SOI technology development has been understanding the impact of floating-body effects on circuit operation, and devising techniques to minimize their effect when needed. The floating-body effects that can impact circuit operation are the hysteretic gate-delay, pass-gate transient leakage,  $V_T$  mismatch (when close tracking of  $V_T$ 's in a circuit is needed), reduced noise margin in SOI, and circuit operation at screening (burn-in) voltages.

#### 5.4.3.1 Hysteretic Gate Delay

At first glance, the history effect on gate delay seems very intimidating to most designers. (How can one design a circuit when one is not sure of the delay through the gate.) As mentioned before there are numerous sources of uncertainty even in bulk-Si design (poly-width variation across the chip,  $V_{DD}$  and temperature variations, noise, switching patterns, device degradation over the life of a part). Indeed a good circuit design must work across a range of channel length, voltage, temperature, and device mismatch that could be due to channel length (gate poly-width) variations. The basic technique to manage the uncertainties in exact operating environment and device condition is to introduce margins in the critical timings (specifically through set-up and hold times of latches). SOI history dependence (which is in the same order of magnitude as other uncertainties in a typical bulk CMOS design) increases slightly the margins that a designer must add to the timing rules. In most circuits, one can bound the delay variations through SOI circuits by initializing the body to high (for the fastest case) or low (for the slowest case). When looking at the fast path, it is important to use the fastest timing rules (i.e., for the latch set-up times). In performance sizing, one should use the slowest body setting.

In a complex circuit, such as a microprocessor, it is possible to measure the magnitude of the history effect on the chip performance. In such circuits, the cycle time is determined by a few cycle-limiting paths (which set the chip  $f_{MAX}$ ). By initializing the chip at different initial chip logic conditions (i.e., running different patterns) prior to running the cycle-limiting path and measuring the resulting variation in the cycle time, one can estimate the impact of the history effect on the chip performance. The reported data so far indicate that the history effect causes 2% variation in the cycle time of the cycle-limiting paths at the processor level [20]. This number is smaller than the numbers

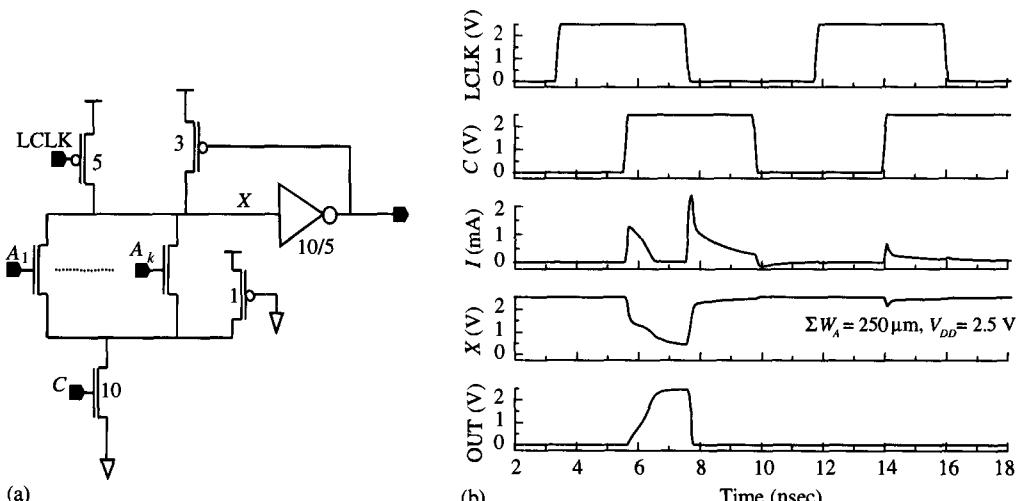
reported in Table 5.2. The delay variation for the logic stages in Table 5.2 were obtained by initializing the bodies at the worst case to measure the maximum hysteresis. The critical path of a microprocessor is made of many gates, all of which cannot be initialized for the worst hysteretic behavior; thus the amount of delay variation is expected to be less than the one in Table 5.2.

#### 5.4.3.2 Transient Pass-Gate Leakage Effect on Circuits

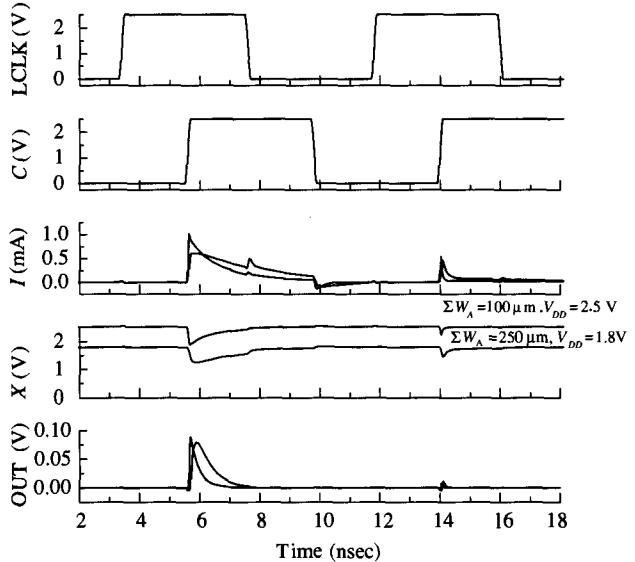
As discussed in Section 5.4.2 transient pass-gate leakage leads to a parasitic current even though the gate is off. This current is much smaller than the ON current of the device (by about a factor of 100). Nevertheless pass-gate leakage can cause circuit failure in a number of circuits. There are two classes of circuits that are susceptible to pass-gate leakage: Nodes with no keeper device (such as DRAM cell, or dynamic nodes with no keeper device), and wide NORs (such as in dynamic circuits or in SRAMs).

The case of wide NOR in a dynamic circuit is illustrated in Fig. 5.13(a). If  $V_D$  and  $V_S$  across all of the devices  $A_i$  are high for a long time, then they will accumulate. Now if device C is turned on (i.e., the common source of  $A_i$  devices is pulled low), then a pulse of current will be passed through the devices  $A_i$ . If the cumulative channel width of the  $A_i$  devices is large enough, they can discharge the dynamic node [as shown in Fig. 5.13(b)]. Possible solutions to this problem are: increasing the size of the keeper device (which results in loss of performance); limiting the cumulative width of the NOR function (Fig. 5.14); eliminating the “charge-sharing” pFET; or discharging the common source of  $A_i$  devices periodically (order of ms) such that accumulation does not have time to occur [19], [20].

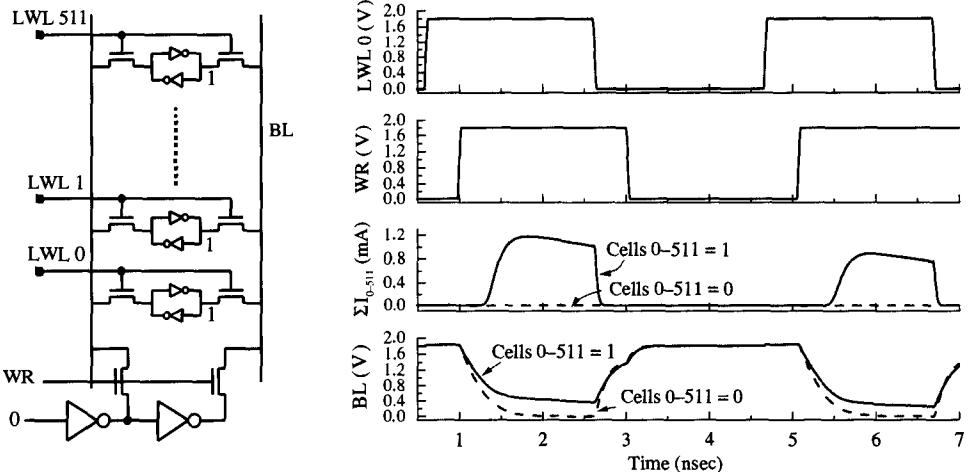
In SRAMs, pass-gate leakage through a large number of pass-gate-configured access transistors on the bit line can impact the SRAM timing, depending on the pattern held in the SRAM cell. Figure 5.15 shows the impact of the pass-gate leakage on the write time. If all the cells on a bit line (BL) hold a “1” [Fig. 5.15(a)], and have been keeping that data for enough time for the access transistors to accumulate, then writing



**Figure 5.13** Simulated transient-pass-gate-leakage-induced output voltage disturbance of a dynamic wide-NOR (a) with weak keeper. Relevant waveforms are shown in (b) indicating that the output signal is failing.

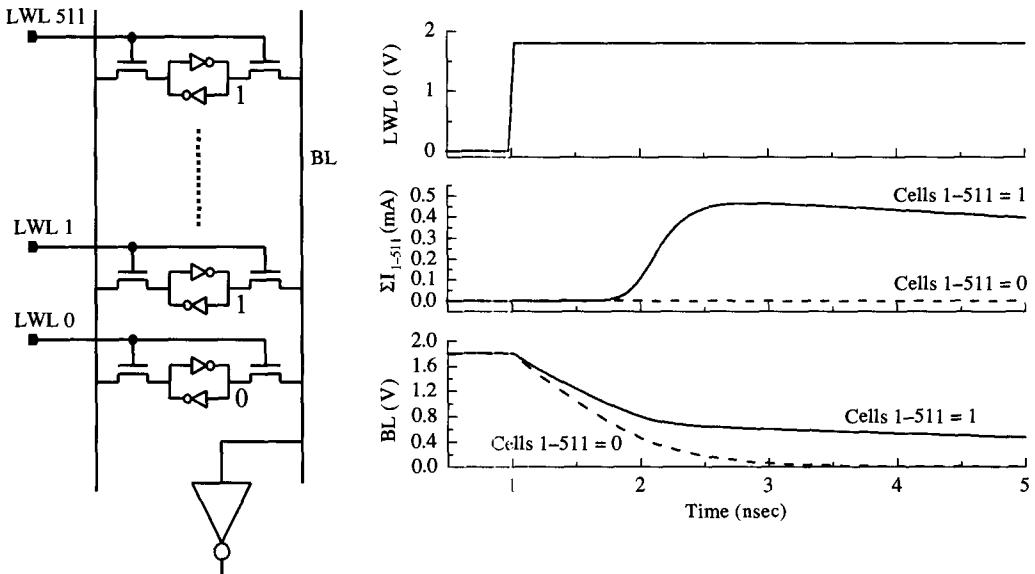


**Figure 5.14** Simulated effect of cumulative NOR nFET width on relevant waveforms. Note the voltage scale at the output node; the signal is very small, indicating that there is no failure in this case.



**Figure 5.15** Simulated data pattern dependence on “write-0” on the bit-line (BL) in a PD-SOI SRAM. This effect is due to the transient pass-gate leakage and so is also history dependent. The offending cells should have been storing a “1” for a long time without any discharge.

a “0” to a cell on the bit line would cause all the other cells to inject their transient pass-gate leakage current on the BL. The excess current would act as extra load on the write amplifier. In this case, the write time would be longer than the case where all the cells were keeping a “0” [Fig. 5.15(b)]. A similar problem can also occur during a read. If all the cells hold a “1” except for one cell that holds a “0,” then when reading the “0” all the other cells would discharge, and increase the time that the differential voltage between BL and BL-bar needs to develop. This case is illustrated in Fig. 5.16, where the BL voltage is shown during read for the cases that all the cells hold “0,” versus the case that



**Figure 5.16** Simulated data pattern dependence of “read-0” on the bit-line (BL) in a PD-SOI SRAM. Similar to Fig. 5.15, this effect is due to the transient pass-gate leakage and so is also history dependent. The offending cells should have been storing a “1” for a long time without any discharge.

only one cell holds a “0” (the sense-amp has been removed to better illustrate the impact of pattern dependence). The SRAM timing dependence on the stored pattern is a unique problem in SOI and should be taken into account by analyzing the circuit at the worst stored pattern. There are circuit methods that can be used to minimize these effects, such as limiting the number of cells in a bit line, precharging the BL to less than full  $V_{DD}$ , using super-bit line architectures, increasing the size of write drivers, or simply ignoring the problem, if it is limited to high-voltage stress corners.

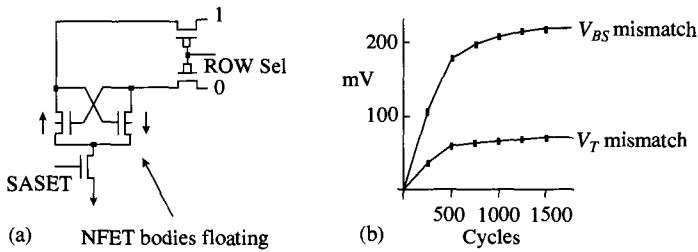
#### 5.4.3.3 Reduced Noise Margin in SOI

Because the FET SOI body is floating it can charge up to relatively large forward voltage, which, as we have discussed, leads to reduction of  $|V_T|$ . The most important effect of reduced  $|V_T|$  is that it makes circuits susceptible to noise. Noise analysis is very much circuit and methodology dependent. When migrating CMOS circuits to SOI, this is one area that needs reanalysis and a significant amount of work, even though at first glance a problem may not be visible.

Elimination of the well-substrate capacitance on SOI, and the reduced junction capacitance, reduces fixed cap on nodes and makes them more susceptible to interconnect coupling compared to bulk CMOS. Noise analysis on SOI must take this degraded behavior into account, and more decoupling capacitance should be added if needed.

#### 5.4.3.4 $V_T$ Mismatch

One key feature of SOI devices is that the  $V_{BS}$  is dependent on the switching history. In cases where close device matching is important, a differential  $V_{BS}$  is very

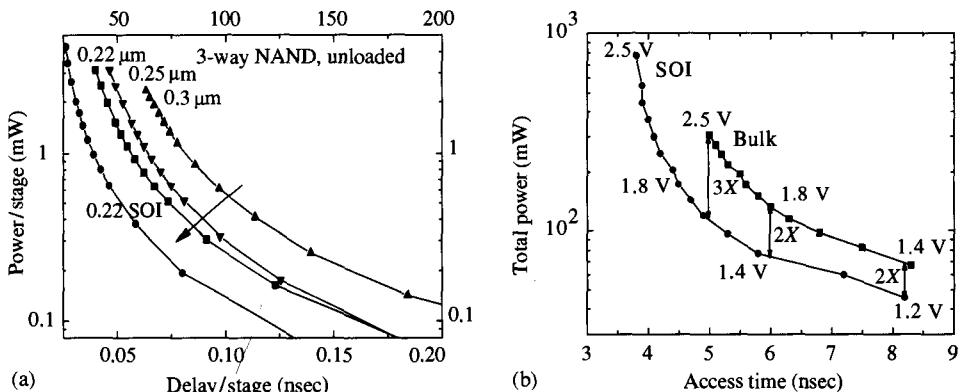


**Figure 5.17** Simulated  $V_{BS}$  and corresponding  $V_T$  mismatch developing over the course of many cycles in a floating-body PD-SOI SRAM sense-amp reading repeatedly the same data pattern.

likely. This is demonstrated in Fig. 5.17(a), where a sense amplifier in a SRAM is repeatedly reading the same pattern. As can be seen in Fig. 5.17(b), a differential in  $V_{BS}$  and  $V_T$  develops. The solution in such a case is to introduce a body contact on the input devices of the sense amplifier. Similarly when mapping analog circuits from bulk to SOI, adding the substrate contact would almost eliminate all the floating-body effects. As an example, analog PLL was mapped from bulk to SOI, with minimal changes, but adding substrate contact to devices which required tracking [8].

## 5.5 SOI FOR LOW POWER

One of the attractions of the SOI technology is its low-power behavior. This has two origins: for a given CMOS generation, SOI has higher performance than a comparable bulk technology. This performance “headroom” allows for operation at lower voltage, and much lower power. This effect is illustrated in Fig. 5.18(a), where power is plotted against delay for an unloaded three-input NAND delay-chain for a number of technologies (0.30  $\mu\text{m}$  to 0.22  $\mu\text{m}$  bulk CMOS, and 0.22  $\mu\text{m}$  SOI CMOS). As the performance of the CMOS technology improves, its power is reduced.



**Figure 5.18** (a) Power/stage versus delay/stage of unloaded three-way NAND gates in 0.22–0.30  $\mu\text{m}$  bulk-CMOS, and 0.22  $\mu\text{m}$  PD-SOI CMOS. (b) Power versus access delay for a 4 Mb SRAM for different power supply voltages in both bulk CMOS and PD-SOI.

The other attraction of SOI is that  $V_{BS}$  is in the forward direction in most switching cases. Forward  $V_{BS}$  dynamically lowers  $|V_T|$ , and a lower  $|V_T|$  is valuable at low-voltage operation where the performance and functionality is proportional to  $V_{DD} - V_T$ . This feature of SOI may facilitate introduction of circuit families which have low-voltage/low-power functionality such as complementary pass gate (CPL) logic. Figure 5.18(b) shows power versus delay for a 4 Mb SRAM at different  $V_{DD}$  values for both bulk CMOS and SOI. At a given delay, the SOI implementation has two to three times less power than the bulk CMOS implementation.

## 5.6 CONCLUSION

We have described a PD-SOI CMOS technology, with considerable advantage over FD-SOI technology in terms of design point and manufacturability. We described the sources of SOI performance advantage, enabling SOI CMOS to achieve a 20–35% performance advantage over bulk CMOS. The floating-body topology introduces a number of unique effects that can impact circuit functionality. They are the history dependence of gate delay and the transient pass-gate leakage. All these effects are manageable, and circuits can be designed around them. We also discussed the attributes of SOI that are valuable in low-power applications.

## REFERENCES

- [1] J. P. Colinge, *SOI Technology*. Kluwer Press, Boston, 1991.
- [2] G. Shahidi, et al., "A Room Temperature 0.1  $\mu\text{m}$  CMOS on SOI," *IEEE Trans Electron Dev.* ED-41, p. 2405, 1994.
- [3] L.T. Su, J. B. Jacobs, J. E. Chung, and D. A. Antoniadis, "Deep-Submicrometer Channel Design in Silicon-on-Insulator (SOI) MOSFETs," *IEEE Electron Dev. Lett.*, EDL-15, p. 366, 1994.
- [4] D. A. Antoniadis, A. Wei, and A. Loctefeld, "SOI Devices and Technology," 29th European Solid-State Device Research Conference, Leuven, Belgium, 1999.
- [5] D. Schepis, et al., "A 0.25  $\mu\text{m}$  CMOS on SOI and its Application to 4 Mb SRAM," *IEDM Tech. Dig.*, p. 587, Dec. 1997.
- [6] A. Ajmera, et al., "A 0.22  $\mu\text{m}$  CMOS-SOI Technology with Cu BEOL," *VLSI Tech. Dig.*, p.15, 1999.
- [7] L. T. Su, J. E. Chung, D. A. Antoniadis, K. E. Goodson, and M. I. Flik, "Self-heating in SOI NMOSFETs," *IEEE Trans Electron Dev.*, ED-41, p. 69, 1994.
- [8] J. Eckhardt, et al., "0.25 micron 1.8 V 1 Ghz PLL for SOI Microprocessors," *ISSCC Dig.*, p. 438, Feb. 1999.
- [9] L. T. Su, D. A. Antoniadis, N. D. Arora, B. S. Doyle, and D. B. Krakauer, "SPICE Model and Parameters for Full-depleted SOI MOSFET's," *IEEE Electron. Dev. Lett.*, EDL-15, p. 374, 1994.
- [10] K. Jenkins, et al., "Measurements of SOI MOSFET I-V Characteristics Without Self-Heating," Proc. of IEEE Int. SOI Conf., 1994, p. 121.
- [11] J. Gautier, K. A. Jenkins, and J. Y. C. Sun, "Body Charge Related Transient Effects in Floating Body SOI nMOSFETs," *IEDM Tech. Dig.*, p. 623, Dec. 1995.
- [12] E. Leobandung, et al., "Scalability of SOI Technology into 0.13  $\mu\text{m}$  1.2 V CMOS Generation," *IEDM Tech. Dig.*, p. 403, Dec. 1998.
- [13] F. Assaderaghi, et al., "History-Dependence of Non-fully Depleted (NFD) Digital SOI Circuits," *VLSI Tech. Dig.*, 1996, p. 122.

- [14] M. Canada, et al., "A 580 MHz 32 bit PowerPC Microprocessor in 0.12 micron  $L_{eff}$  CMOS SOI with Cu Interconnects," *ISSCC Dig.*, Feb. 1999.
- [15] C. A. Maier, et al., "A 533-MHz BiCMOS Superscalar RISC Microprocessor," *IEEE J. Solid-State Circuits*, vol. 32, p. 1625, 1997.
- [16] G. Shahidi, et al., "Non-Fully-Depleted SOI Technology for Digital Logic," *ISSCC Dig.*, Feb. 1999.
- [17] A. Wei and D. A. Antoniadis, "Measurement of Transient Effects in SOI SRAM/SRAM Access Transistors," *IEEE Electron Dev. Lett.*, EDL-17, p. 193, 1996.
- [18] M. M. Pellela, J. G. Fossum, D. Suh, S. Krishnan, K. A. Jenkins, and M. J. Hargrove, "Low-voltage Bipolar Effect Induced by Dynamic Floating-body Charging in Scaled PD/SOI MOSFET's," *IEEE Electron Dev. Lett.*, EDL-17, p. 196, 1996.
- [19] F. Assaderaghi, et al., "Transient Pass-transistor Leakage Current in SOI MOSFET's," *IEEE Electron Dev. Lett.*, EDL-18, p. 241, 1997.
- [20] D. Allen, et al., "A 0.2 micron 1.8 V SOI 550 MHz 64b PowerPC Microprocessor," *ISSCC Dig.*, Feb. 1999.

Duane Boning

*Massachusetts Institute of Technology*

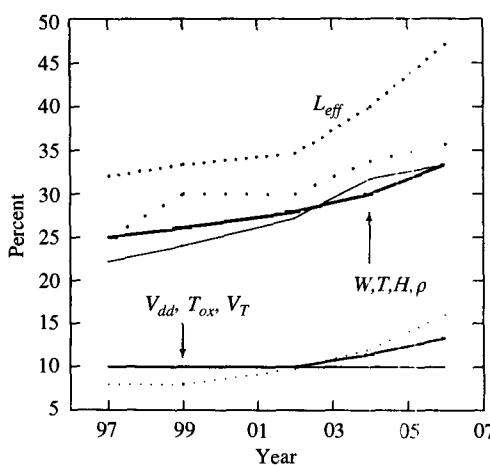
Sani Nassif

*IBM Austin Research Laboratory*

## 6.1 INTRODUCTION: SOURCES OF VARIATION

Variation is the deviation from intended or designed values for a structure or circuit parameter of concern. The electrical performance of microprocessors or other integrated circuits are impacted by two sources of variation. *Environmental factors* arise during the operation of a circuit, and include variations in power supply, switching activity, and temperature of the chip or across the chip. *Physical factors* during manufacture result in structural device and interconnect variations that are essentially permanent. These variations arise due to processing and masking limitations, and result in random or spatially varying deviations from designed parameter values. In this chapter we focus on *parametric variation* due to continuously varying structural or electrical parameters, as these can significantly impact not only yield but also performance in high-speed microprocessor and other digital circuits.

Such parametric variation is becoming a larger concern, as variation and margins for device and interconnect do not appear to be scaling at the same rate. Figure 6.1



**Figure 6.1** Device and interconnect variation trends for different technology generations. The percentage of the total variation accounted for by within-die variation for selected parameters is plotted.

shows the general trend in the ratio between within-die and total variation for some key technology device and wire parameters. The data are extracted from the 1997 National Technology Roadmap for Semiconductors report, augmented with data from IBM processes. Over the time period of interest, we see that the within-die proportion of  $L_{eff}$  variation increases from 40% to 65%. The wire geometry parameters, width  $W$ , height  $H$ , thickness  $T$ , and resistivity  $\rho$  also undergo a fairly major increase. Other parameters such as the oxide thickness  $T_{ox}$  and threshold voltage  $V_{th}$  increase at a lower rate. Models and methods for dealing with such variation trends will become an increasingly important part of high-performance circuit design.

Section 6.2 presents an overview of parametric variation statistical descriptions ranging from simple lumped distributions, to separation of inter-die and intra-die variation, and finally to detailed models of spatial variation. Typical process variations of concern in device and interconnect are discussed in Section 6.3. Section 6.4 overviews the circuit analysis approaches that are used to evaluate and guard against variation, including random, spatial, and worst-case analyses. Examples of the impact of process variation on circuit performance are presented in Section 6.5, illustrating statistical analysis and impact assessment methods in the context of typical circuit concerns including clock distribution and global path evaluation.

## 6.2 OVERVIEW: STATISTICAL DESCRIPTIONS

In order to design integrated circuits, some statistical description for parametric variation is used to (a) estimate the characteristics or statistical distribution for a parameter of interest, and (b) understand or minimize the impact on circuit performance of those variations. In this section we overview the successive levels of detail used, ranging from simplified or “lumped” statistical descriptions to more specific decompositions of the sources of variation.

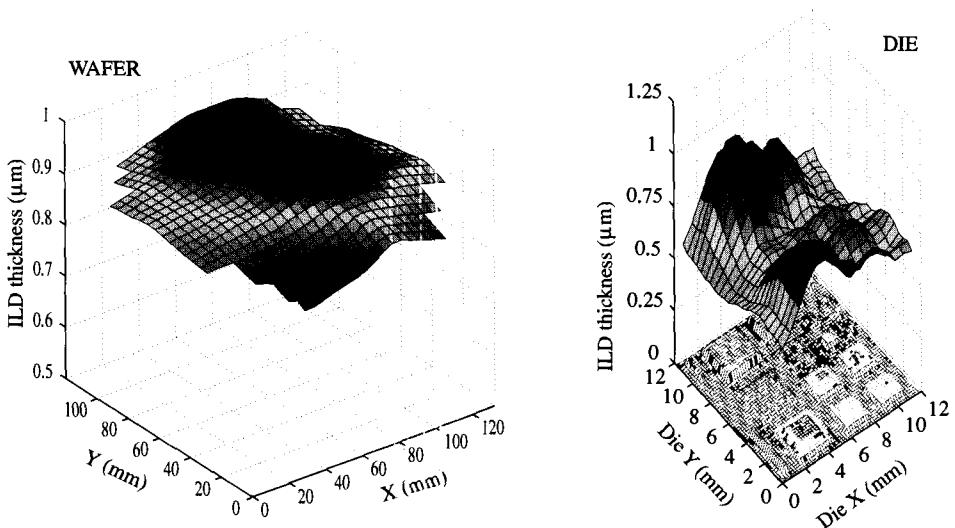
### 6.2.1 Lumped Statistics

The most basic approach is to characterize the distribution of some parameter of interest  $P$  over some sample of devices or structures, and to estimate some small number of statistical moments to characterize that distribution. For example, one might measure the channel length  $L$  over some large number of randomly selected nominally 0.25  $\mu\text{m}$  devices across a wafer (or across many lots, wafers, and chips), find that these are approximately normally distributed, and estimate the mean and variance (or standard deviation) based on those measurements. In this approach the details of the physical sources of this variation are not considered; rather, the combined set of underlying deterministic (but not understood) as well as random contributions are simply lumped into a combined “random” statistical description.

The simplest approach (although not very accurate when there are correlations between parameters) for statistical circuit design in the face of random variation is to treat each variation as independent, and build in additional margin (e.g., 3–4 $\sigma$  or more, depending on the number of devices that must stay within the required margin). Methods that provide better worst-case estimates that account for cross-correlation between parameters are presented in Section 6.4.

### 6.2.2 Separation of Inter-Die and Intra-Die Variation

For the purposes of circuit design, it is often important to separate sources of parametric variation into two categories [7], [15]: *inter-die* and *intra-die* variation. These two sources of variation are illustrated in Fig. 6.2 for the case of interlevel dielectric thickness ( $T_{ILD}$ ) variation after oxide deposition and chemical mechanical planarization (CMP). As can be seen in Fig. 6.2, in some cases the variation within the die (e.g., due to layout pattern dependencies) may be substantially larger than variation across the wafer; thus, the matching of devices within the die may be very different than the matching at the same location from one chip to the next.



**Figure 6.2** Variation in ILD thickness across the wafer (left), and across the die (right).

We can thus decompose the simple lumped variation described in the previous section into two or more separate contributions. One may be able to model some of these more accurately, enabling one to ignore or block against that particular variation component. For example, inter-die variation may have relatively minor impact, while intra-die variation may be a large concern for circuit timing.

### 6.2.3 Inter-Die Variation

Inter-die variation is the difference in the value of a parameter across nominally identical die (whether those die are fabricated on the same wafer, on different wafers, or in different lots) and is typically accounted for in circuit design as a shift in the mean of some parameter value (e.g.,  $V_{th}$  or wire width) equally across all devices or structures on any one chip. For purposes of circuit design it is usually assumed that each contribution or component in the inter-die variation is due to different physical and independent sources, and it is usually sufficient to lump these contributions together into a single effective die-to-die variation component with a single mean and variance.

While a simplified distribution is often assumed to capture the variance die-to-die, it is often possible and helpful to characterize and model systematic trends across the wafer. For example, chip speed may depend on some parameter that varies in a systematic “bowl” fashion across the wafer; accurate speed binning or yield analysis and improvement may depend on understanding such patterns from one die to the next. In typical circuit design, die-to-die variations are the simplest to analyze; a small number of situations may be analyzed in which all device or interconnect parameters on the chip are mean shifted together up or down by some amount (e.g.,  $2-3\sigma$ ).

### 6.2.4 Intra-Die Variation

Intra-die variation is the deviation occurring spatially within any one die. Such intra-die variation may have a variety of sources depending on the physics of the manufacturing steps that determine the parameter of interest. In contrast to inter-die variation (affecting all structures on any one chip equally), intra-die variation (or variance across the chip) contributes to the loss of *matched* behavior between structures on the same chip. In the case of inter-die variation, the concern is how variation that “rises or falls” in unison across a chip may impact performance or parametric yield. In the case of intra-die variation, we are concerned with an entire set of elements (such elements may be individual MOS transistors, signal lines or segments of signal lines, or any other geometric or electrical parameter of the circuit) and how such devices vary differently from designed or nominal values, or how such devices thus differ unintentionally from each other.

Such intra-die or “across-chip variation” can arise from a number of manufacturing sources. Two sources are of particular importance: wafer level trends, and die-pattern dependencies. Wafer scale variation can result in small trends that are reflected across the spatial range of the chip. For example, many deposition processes suffer from a gentle “bowl” or concentric ring pattern in thickness from the center of the wafer outward. While such trends may be gentle (e.g., 5–10% across the wafer), they can still result in systematic trends across the die (e.g., a “slanted plane”) which may be important. Methods for dealing with such trends are discussed in Section 6.4.

Die-pattern (or layout) dependencies, on the other hand, can create additional variations that have become increasingly problematic in IC fabrication. The layout pattern is found to interact badly with some fabrication processes creating additional unwanted dependencies. For example, two interconnect lines that are designed identically in different regions of the die may result in lines of different width, due to photolithographic interactions, plasma etch microloading, or other causes. Distortions in lens and other elements of a lithographic system are also known to create systematic variations across the die. The length scales of such perturbations can vary: (1) line distortions in exposure may occur on the scale of a micron or less; and (2) film thickness variations arising in chemical mechanical polishing may occur on the scale of millimeters.

While such variation may be *systematic* in any given die, the set of these variations across many die may have a random distribution. For example, a “slanted plane” pattern may be oriented randomly from one die to the next. For this and other reasons (e.g., lack of layout information), systematic variations are often bounded by or treated as some large estimated random variation.

## 6.3 SURVEY OF PROCESS VARIATIONS

Given the perspective of different categories of variation, we can now briefly survey some of the specific process variations typically of concern in evaluating devices and interconnect in integrated circuit design [2].

### 6.3.1 Device Geometry Variations

The first set of process variations of concern relate to the physical geometric structure of MOSFET and other devices (resistors, capacitors) in the circuit. These typically include:

- *Film thickness variations:* The gate oxide thickness is a critical but usually relatively well-controlled parameter. Variation tends to occur primarily from one wafer to another with good across-wafer and across-die control. Other intermediate process thickness variations (e.g., poly or spacer thickness) can impact channel length, but are rarely directly modeled.
- *Lateral dimension variations:* Lateral dimensions (length, width) typically arise due to photolithography proximity effects (a systematic pattern dependency); mask, lens, or photo system deviations (a repeated die-dependent variation, though not directly a function of the layout density or other layout parameters); or plasma etch dependencies (which can have both wafer scale etch rate dependencies, as well as layout density, aspect ratio, or other dependencies).

MOSFETS are well known to be particularly sensitive to effective channel length (and thus to poly gate length), as well as gate oxide thickness and to some degree the channel width. Of these, channel length variation often is singled out for particular attention, due to the direct impact such variation can have on device output current characteristics.

### 6.3.2 Device Material Parameter Variations

Another class of process variations in MOSFETS relate to internal material parameters.

- *Doping variations:* Deviations arising due to implant dose, energy, or angle variation can affect junction depth and dopant profiles (and thus may also impact effective channel length), as well as other electrical parameters such as threshold voltage. In deep submicron devices, drain engineering (e.g., halos to reduce short-channel effects) further increase the possible impact of implant and diffusion variation. Variation in thermal anneal and gate doping can also change the degree of gate depletion in an active device, and cause variation in the effective gate oxide thickness. Depending on the gate technology used, these deviations can lead to some loss in the matching of NMOS versus PMOS devices even in the case where within-wafer and within-die variation is very small.

- *Deposition and anneal:* Additional material parameter deviations are observed in silicide formation, and in the grain structure of poly or metal lines. These variations may depend on the deposition and anneal processes, and thus suffer from substantial wafer-to-wafer and within-wafer deviations, and may also have large random device-to-device components (due to crystallographic grain or phase effects). These material parameter deviations can contribute to appreciable contact and line resistance variation.

### 6.3.3 Device Electrical Parameter Variations

While device geometry and electrical parameter variations are often important to understand based on their “root causes,” one often focuses on the actual electrical parameter variations. In many cases, the underlying geometry distribution is not characterized, but instead the key electrical parameters are directly extracted and modeled.

- *$V_T$  variation:* A key concern is threshold voltage ( $V_T$ ) variation. In addition to geometric sources, mobile charge in the gate oxide can introduce a bias dependent variation; this is sometimes approximated as being about 10% of the  $V_T$  of the smallest device in a given technology [2].
- *Discrete dopant variation:* Another source of  $V_T$  variation that is just beginning to become important in SRAM and other circuits is related to random dopant fluctuations. Concern has been raised about the random placement and concentration fluctuations due to discrete location of dopant atoms in the channel and source/drain regions. One approach to examining these effects is to use Monte Carlo simulation over an ensemble of simulated doping events. For small device size (0.1  $\mu\text{m}$ ), it becomes feasible to track the locations of each silicon and dopant atom in a small volume, and to consider the distribution of device behavior that results. Frank et al. [5] quantified the magnitude of threshold voltage variations in a 25 nm MOSFET, and found that devices with well-designed doping profiles have an intrinsic  $V_T$  voltage uncertainty of about  $10/\sqrt{W} \text{ mV } \mu\text{m}^{\frac{1}{2}}$  with  $W$  being the width of the device. The implication is that such random discrete dopant fluctuations will likely be tolerable for logic, but may prove a problem for narrow and dense devices such as SRAM blocks containing large numbers of devices that must be well matched.
- *Leakage currents:* Other electrical parameter variations can also be of concern in circuit design. Subthreshold leakage currents may vary substantially, and can be impacted by shallow trench isolation structure and stress imperfections due to oxidation and CMP.

### 6.3.4 Interconnect Geometry Variations

Just as in devices, vertical and lateral dimensions as well as material property deviations can be important sources of variation in interconnect structures. Key

geometry concerns include:

- *Line width and line space:* Deviations in the width of patterned lines again arise primarily due to photolithography and etch dependencies. At the smallest dimensions (lower metal levels) proximity and lithographic effects may be important, while at other levels etch effects (so-called RIE lag or aspect ratio dependent etching), which depend on line width and local layout, can be significant. Deviations in line width can directly impact line resistance, as well as the capacitance from one layer to layers above or below. Deviations in line width can also result in line space differences affecting the magnitude of line-to-line coupling within the layer (and can impact not only capacitance but also cross talk and signal integrity).
- *Metal thickness:* In conventional metal interconnect, the thickness of sputtered or otherwise deposited metal films and liners or barriers is usually well controlled, but can vary from wafer-to-wafer and across wafer. In damascene (e.g., copper polishing) processes, on the other hand, dishing and erosion can significantly impact the final thickness of patterned lines [11], with line thickness losses of 10 to 20% depending on the particular patterns.
- *Dielectric thickness:* The thickness of deposited and polished oxide films can also suffer substantial deviations. While wafer level deposition thickness can vary (typically on the order of 5%), more troublesome are pattern-dependent aspects of such deposition. For example, the deposition profile using high-density plasmas (HDP) can depend strongly on the width or size of a feature being deposited over. Furthermore (as illustrated in Fig. 6.2) the CMP process can introduce strong variations across the die, resulting from the effective density of raised topography in different regions across the chip [16].
- *Contact and via size:* Contact and via size can be affected by etch process variation, as well as systematic layer thickness dependencies. Depending on the via or contact location, for example, the etch depth can be substantially different, resulting in different degrees of lateral opening. Such size differences can directly affect the resulting contact or via resistance.

### 6.3.5 Interconnect Material Parameter Variations

Finally, some degree of material property variation may also be important in interconnect structures.

- *Contact and via resistance:* Contact and via resistance related to good ohmic contact can be sensitive to etch and clean processes, with substantial wafer-to-wafer and random components.
- *Metal resistivity and dielectric constant:* While metal resistivity variation can occur (and include a small random element), resistivity usually varies appreciably on a wafer-to-wafer basis and is usually well controlled. Similarly, dielectric constant may vary depending on the deposition process, but is usually well controlled. It is possible that pattern-dependent and directional effects may become important for some low K dielectrics being considered for future interconnect technologies.

## 6.4 METHODS TO CHARACTERIZE AND ADDRESS VARIATION

In this section we briefly examine the methodologies used to understand and address process variations such as those presented in the previous section. Many of these approaches are built on statistical modeling and mathematical optimization methods; however, in this section we seek only to overview these approaches, and other references should be consulted for mathematical details. Key elements of statistical circuit analysis overviewed here are (a) extraction of statistical device models; (b) sensitivity analysis to estimate the effect of variation sources; (c) worst-case methods to more carefully limit and study variation concerns in circuit designs; and (d) spatial modeling and mismatch analyses.

### 6.4.1 Statistical Device Models

From the perspective of circuit design, the parameters  $P$ , which characterize a process, are often the model parameters required to perform circuit simulation. Internal to a circuit simulator, the model parameters are used to express the dependence between quantities such as current, charge, and voltage. A simple example of a device model is the Spice Level-1 MOSFET model:

$$\begin{aligned} I_{ds} &= 0 \quad \text{for } V_{gs} - V_{th} < 0 \\ I_{ds} &= \mu C_{ox} \frac{W}{L - \Delta L} \left[ (V_{gs} - V_{th}) V_{ds} - \frac{V_{ds}^2}{2} \right] \quad \text{for } 0 < V_{ds} < V_{gs} - V_{th} \\ I_{ds} &= \mu C_{ox} \frac{W}{L - \Delta L} (V_{gs} - V_{th})^2 \quad \text{for } 0 < V_{gs} - V_{th} < V_{ds} \end{aligned} \quad (61)$$

for which  $P = \{W, L, V_{th}, \mu, \Delta L, C_{ox}\}$ . Many of these quantities are not directly measurable and must therefore be inferred from measurements of  $I_{ds}$  versus  $V_{gs}$  and  $V_{ds}$ . This inference process, called *model parameter extraction* is usually performed using a nonlinear least squares analysis (e.g., see [1] Chap. 6). Due to the fact that the model is only an approximation to reality, and because the minimization is performed with finite tolerances, the parameter estimate derived is subject to error.

In addition to a nominal model fit, we need to characterize variations in  $P$ . This may be done by measuring a number of devices, performing parameter extraction on each set of measurements to get a population of parameters  $P$ , and using the population to estimate the statistics of  $P$ .<sup>1</sup> Based on these statistics, large numbers of hypothetical cases can be simulated to study the resulting variation in performance. Numerous difficulties in this approach exist, including computational costs, data collection requirements, and potentially large errors in performance estimates due to propagation of systematic device model fitting errors.

The difficulty and high cost of getting reliable and accurate statistics for the model parameters result in a situation where (1) the parameter statistics are not updated often to reflect changes and maturation in the fabrication process; and (2) there is often a large incentive to use analysis and design methods that are less sensitive to the detailed

<sup>1</sup> There are a number of other approaches to solving this problem; see, for example [3], [8].

statistics of  $P$ , hence the extensive use of worst-case analysis techniques (which we will discuss below). Nevertheless, a large number of other analysis techniques have been tried with various levels of success. More information on the various alternative statistical analysis techniques can be found in [20], [14], and [4].

### 6.4.2 Sensitivity Analysis

In some cases, it is easier to characterize one set of parameters (perhaps related to geometric variation) than the resulting electrical or simulation model parameters. In such cases, one is concerned with the *transmitted variability* or propagation of variance from one parameter  $P$  through to another parameter  $Q$ , by way of a known analytic or numeric function  $f$ , where  $Q = f(P)$ .

In the case where  $f$  can only be evaluated numerically (e.g., in understanding the impact of some process variation such as an anneal temperature on resulting geometric structure), monte carlo or other sampling methods are often utilized [13]. In many cases, however, a simple *sensitivity analysis* approach is used through a first-order expansion of some analytical function  $f$  relating  $P$  and  $Q$ :

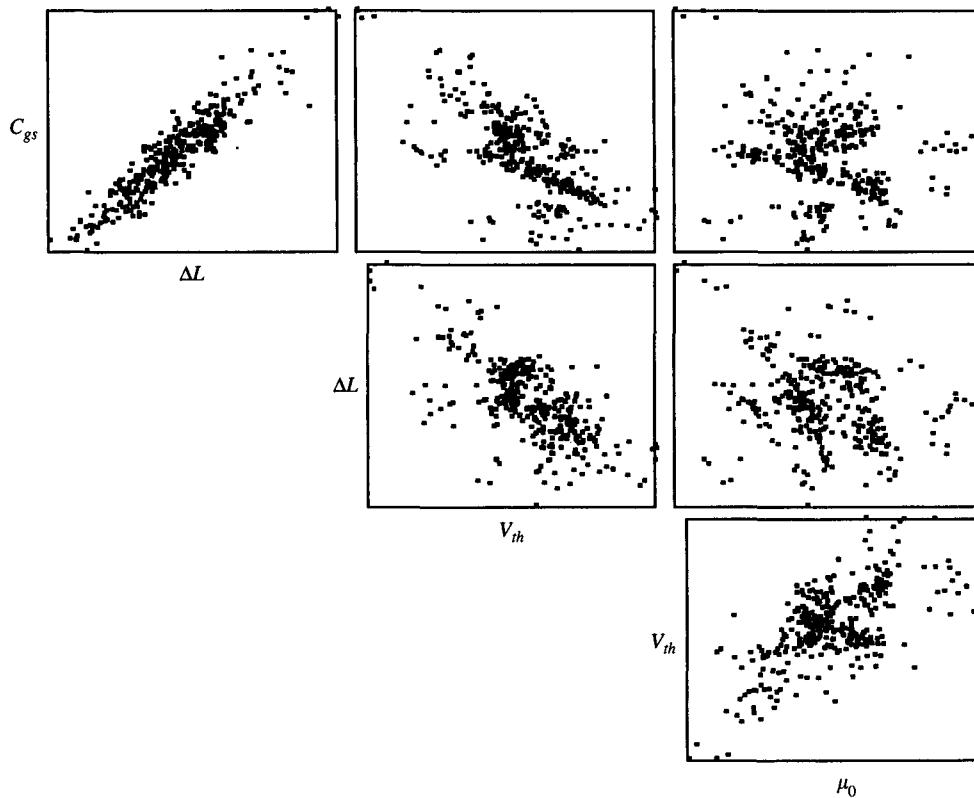
$$\begin{aligned} Q + \Delta Q &= f(P + \Delta P) \\ \Delta Q &\approx \left| \frac{\partial f}{\partial P} \right| \Delta P \end{aligned} \tag{6.2}$$

where  $\Delta P$  and  $\Delta Q$  are typically considered to be the standard deviations of parameters  $P$  and  $Q$ . While many functions do not preserve normality, it is often assumed that the small deviations of  $\Delta Q$  can also be approximated by a normal distribution, so that the variance propagation is approximated as  $\sigma_Q^2 \approx (\partial f / \partial P)^2 \sigma_P^2$ . Given approximate variance values for some set of process variations, the resulting first-order electrical impact of the variations on device or canonical circuits is often derived and compared for different circuit, layout, or other design rule options (as, for example, in Section 6.5.2).

### 6.4.3 Worst-Case Analysis

The various components of  $P$  are usually correlated. For example, Fig. 6.3 shows a representative distribution of four MOS transistor model parameters from a modern 0.25  $\mu\text{m}$  process. The correlation structure of  $P$  is thus required for accurate statistical analysis. Ignoring the correlation, i.e., assuming it is zero, leads to statistical performance estimates which are in reality extremely improbable and are overly pessimistic. Principal component analysis [12] is often used to transform highly correlated process parameters to a smaller set of uncorrelated parameters to simplify statistical design analysis.

The most common method for analyzing the implications of random and correlated variations is worst-case analysis [4]. Consider a circuit performance  $z$  (e.g., clock speed) that is a function of model parameters  $P$ , expressed as  $z = f(P)$ . Due to variations in  $P$ , the performance  $z$  is a random variable. The goal of worst-case analysis, like all other forms of statistical design analysis [14], is to determine a measure of goodness or quality of the design. The ideal such measure is the *yield* of the design, which is defined as the proportion of circuits that meet the specifications. Since computing the function  $f$  typically involves performing a computationally expensive circuit simulation, computing the yield directly is very expensive. Worst-case analysis is the



**Figure 6.3** Distribution of transistor model parameters.

standard method for indirect measurement of the yield by effectively finding the worst-case bounds for the yield. Various methods also exist whose goal is to work backward to find the worst-case process parameters  $P_{wc}$  (or process “corners” for  $P$ ) that assure that the required yield is achieved.

The effort necessary for finding  $P_{wc}$  is nontrivial, and in fact is greater than that required to compute the yield since it requires the complete statistics of the output parameter  $z$ . Fortunately, however, the same performance of two structurally similar circuits often exhibits similar sensitivity to the parameters  $P$  and therefore results in nearly identical worst-case parameters [4]. As a practical example, a library of ASIC cells can use one unique setting for worst-case parameters for each type of performance (e.g., delay, power dissipation, or noise immunity). This practice is so prevalent, in fact, that it is standard practice for manufacturing organizations to specify several sets of model parameters which are the worst cases or corners for typical digital circuit performances.

#### 6.4.4 Spatial Variation Modeling and Mismatch

In analog circuit design, substantial work has been done to understand and model issues in device matching [7]. Increasingly, device matching is also of concern in high-performance digital subcircuits. Here we overview some of these variation issues and the approaches used to analyze or guard against them.

Suppose we are concerned with the difference  $\Delta P$  in some parameter between two devices. Ideally, we want perfect matching, or  $\Delta P = 0$ ; but since we can't always achieve this, the spread or variance in  $\Delta P$  gives us a sense of possible mismatch. The most simple case is when the two devices are truly independent; then the variance in the mismatch is twice the variance of an individual device. A more interesting case occurs when the variation in any given device depends on the device area (or length). In particular, when the underlying source of variation can be "averaged" out for larger devices, the device mismatch depends on both the underlying statistics and the size of the respective devices. In such cases, it is possible to achieve better matching (i.e., reduce the variance in  $\Delta P$ ) at the cost of increasing the size of the structures.

A third important situation is when the mismatch depends on the separation distance between two devices. For example, the effect of a wafer-level trend as seen by two devices on the same die will often be dependent on the distance between those devices; a useful approximation is to consider the variance of the mismatch to be proportional to the square of the distance between the devices [7]. An obvious strategy is to keep devices that require good matching as close together on the die as possible.

A final situation is when some deterministic (e.g., layout dependent) spatial process variation exists. Early in the design process such systematic variations are functions of as-yet unavailable layout information and therefore must be treated as random. Consider a parameter  $P$  whose intra-die variations are systematic and whose total variations can then be expressed in Eq. (6.3):

$$P = P_0 - \mathcal{F}(x, y, \theta) + \tilde{P}_\epsilon \quad (6.3)$$

where  $P_0$  is the mean for the given die, the function  $\mathcal{F}$  describes the intra-die spatial distribution of  $P$ ,  $\theta$  denotes a vector of random parameters of the spatial function, and  $\tilde{P}_\epsilon$  is remaining unexplained random variation. Before spatial  $(x, y)$  information is available, the intra-die component of  $P$  can only be modeled as a random distribution having variance consistent with the maximum deviation possible due to  $\mathcal{F}$ . One approach is to simply use a broad random variance in a simplified approximation for the intra-die variation. An improvement is to treat  $x$  and  $y$  as randomly distributed across the chip, and then evaluate the function  $\mathcal{F}$  across this distribution. In general, this will result in a larger total variance for  $P$  than using Eq. (6.3) but smaller (more accurate) variance than using a large lumped "random" approximation. This approach is illustrated for a clock tree example in Section 6.5.1.

Matching of devices in the face of such layout variation can often be improved by making the layout around the devices as matched as possible. For example, one often adds dummy lines to match proximity effects, and dummy fill patterns are widely used to make the layout pattern density as uniform across the die as possible [17].

## 6.5 APPLICATION OF METHODS TO INTERCONNECT IMPACT ANALYSIS

In the previous sections we have introduced the notion of statistical description of parameter variation, described process variation sources, and discussed methods for characterization and analysis of random and spatial variation. In this section we discuss applications of these different methods for the analysis of the impact of variation on interconnect performance.

Clock distribution is a common and important problem in large synchronous digital circuits. A measure of the quality of the clock distribution network is the *skew* or difference in clock arrival times at any two destinations. A common technique to minimize skew is to use so-called *H-Trees* where the wire length between the center of the tree and all destinations is identical. Figure 6.4 depicts such a tree with a single central buffer and 64 loads.

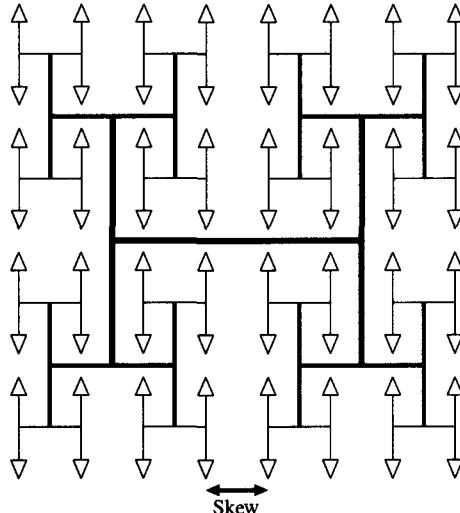


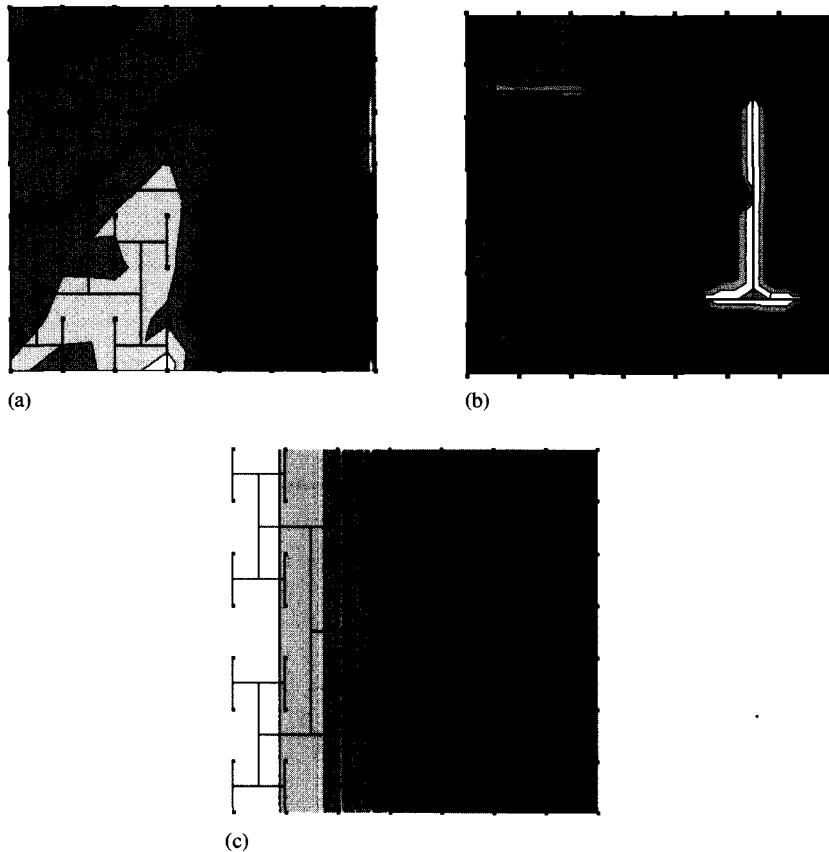
Figure 6.4 H-Tree for clock distribution.

### 6.5.1 Example: Random and Wafer Level Variation Impact

In this example [9] we assume that all deterministic factors<sup>2</sup> have been nulled out, and we analyze the impact of three sources of within-chip variations: (1) random variations in FET channel length,  $\Delta L$ ; (2) random variations in wire width,  $\Delta W$ ; and (3) wafer-level variations in  $\Delta L$ . We select the skew between the two leaves illustrated in Fig. 6.4 as our performance metric  $z$ . We seek to find the maximum (worst-case) skew due to the wafer-level or random portion of within-chip variations subject to constraints on the statistics of  $\Delta L$  and  $\Delta W$ . In each case we represent the sample of values of  $\Delta L$  or  $\Delta W$  by  $\psi$ .

- Random  $\Delta L$  effects: In this case  $\psi$  consisted of 65 samples of  $\Delta L$ , for the central buffer and each of the 64 loads. The performance  $z$  was found by simulating using the nominal parameters of a  $0.25\text{ }\mu\text{m}$  CMOS process. The statistics for  $\Delta L$  were taken to be  $N(0.0, 0.035\text{ }\mu\text{m})$  to reflect the technology's within chip tolerance on  $\Delta L$ . The resulting spatial distribution of  $\Delta L$  is illustrated in Fig. 6.5(a).
- Random  $\Delta W$  effects: In this case  $\psi$  consisted of 63 samples of wire width  $W$ , one for each of the wire segments in the H-Tree. The impact of  $W$  on wire resistance and capacitance was taken into account. The statistics of  $\Delta W$  were taken to be

<sup>2</sup> An example may be a difference in loading from one node in the tree to another, which can also result in skew.



**Figure 6.5** Results of intra-die worst-case analysis for (a) random channel length variation, (b) random line width variation, and (c) spatial channel length trend analysis.

$N(0.0, 0.25 \mu\text{m})$  reflecting the technology's within-chip tolerance on  $W$ . The resulting spatial distribution of  $\Delta W$  is illustrated in Fig. 6.5(b).

- Spatial  $\Delta L$  effects: In this experiment  $\psi$  consisted of the three variables  $w_0$ ,  $w_x$ , and  $w_y$  in a spatial function  $\Delta L = w_0 + w_x x + w_y y$ . A uniform distribution was assumed for all three variables. The resulting worst-case distribution of  $\Delta L$  is illustrated in Fig. 6.5(c).

The importance of accounting for constraints on the wafer level and random statistics is illustrated by the worst-case skew values in the three scenarios above. In a simple worst-case analysis where many samples of  $\psi$  are evaluated without consideration of spatial constraints, the worst-case skew is overestimated. In the case of  $\Delta L$ , the skew using a constrained worst-case analysis (described more fully in [9]) is 139 ps. This compares to the more pessimistic worst-case skew based on  $\pm 3\sigma$  of 171.5 ps. Similarly, in the case of  $\Delta W$  effects, the constrained variation analysis gives a worst-case skew of 41 ps, compared to a pessimistic worst-case analysis of 48.9 ps. In the case of wafer-level spatial  $\Delta L$  trends, the skew is 9.98 ps under both methods (since systematic only and no random components are included in both analyses).

### 6.5.2 Example: Interconnect Sensitivity Analysis

Zarkesh-Ha et al. considered both interconnect geometry and device variation, as well as “system” or environmental (e.g., temperature) disturbances on chip clock distribution networks [18]. The methodology used is essentially a sensitivity analysis coupled to analytic modeling, as introduced in Section 6.4.2. Zarkesh-Ha derives a compact model using a simplified equivalent circuit composed of the H clock tree, the driver device, and other sub-blocks, so that the total delay  $T_{Delay}$  of the clock network can be written as

$$T_{Delay} = T_{H-Tree} + T_{Driver} + T_{SubBlk} \quad (6.4)$$

where analytic expressions for each component are derived.

Given the relation for  $T_{Delay}$  and assuming small variations around the nominal, a first-order sensitivity analysis can be performed to evaluate the clock skew  $T_{CSK}$  due to each variation component  $x$  as

$$T_{CSK}(x) = \Delta T_{Delay} \approx \left| \frac{\partial T_{Delay}}{\partial x} \right| \Delta x \quad (6.5)$$

In this fashion, analytic approximations for variations due to each contributing factor  $x$  are derived and considered. For example, an important “system” (design) variation is that due to uneven distribution of registers or load capacitance across the chip. If an approximate value for  $\Delta C_L$  is available, the clock skew resulting from this variation can be approximated as

$$T_{CKS}(C_L) = 0.7 R_{tr} C_L \cdot \left( \frac{\Delta C_L}{C_L} \right) \quad (6.6)$$

Several system parameter variations are considered in [18] and [19] including IR drop and temperature gradients, as well as manufacturing variations in key device and interconnect geometry parameters. Device variations include threshold voltage fluctuation, transistor channel length tolerance, and gate oxide thickness tolerance. Interconnect geometry parameters considered include ILD thickness variation and wire thickness variation. Using assumed nominal and variation values for process and system design parameters, Zarkesh-Ha estimates sensitivities and derives a feasible design plane for the clock distribution network [19].

### 6.5.3 Example: Statistical Interconnect Impact

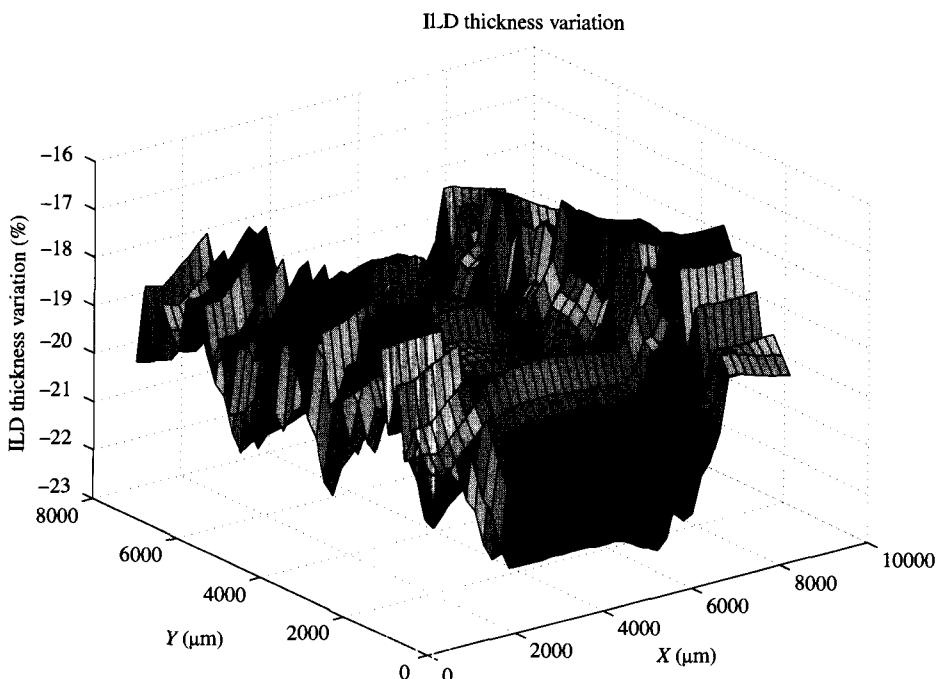
While Zarkesh-Ha et al. lump variation, such as ILD thickness, into a variation component (e.g.,  $\Delta T_{ILD}$  equal to 3% in [19]), in many cases such parameter variation is largely systematic. In these cases it can be dangerous to assume that the variation along different paths or in different locations is independent of each other, and a variance estimate can be misleading. The correlation structure can either be such that one seriously overestimates the variance, or seriously underestimates the actual range of values experienced.

For example, Orshansky, Spanos, and Hu [10] considered the relative importance of device and interconnect variability and delay, but make a correction due to parameters that affect local (devices, local interconnect) versus global structures (global interconnect). In particular, Orshansky noted that the variance in some parameters (e.g., capacitance in long lines) may be averaged over the length of the line thus

decreasing variance (as discussed in Section 6.4.4). Specifically, if the variance  $\sigma_L^2$  in a parameter  $P$  of interconnect with length  $L_i$  is characterized, then the total variance  $\sigma_P^2$  for a longer interconnect with length  $L$  is reduced to  $\sigma_P^2 = \sigma_L^2/N$  where  $N = L/L_i$ . This assumes that each segment of the interconnect (of length  $L_i$ ) can be treated as independent with respect to parameter  $P$  and is drawn from the same Gaussian distribution. In some cases this might be a good assumption, but in cases where systematic spatial trends exist, or where layout-dependent parameter variation exists, this can also be a risky assumption. Based on their models and assumptions, Orshansky et al. compare the relative significance of variation sources on critical path delay in a 0.18  $\mu\text{m}$  technology. They find that when hierarchically scaled interconnect is used (e.g., “fat” wires at upper levels), delay variation is about 48% of the delay, and only 12% of that variability is due to interconnect structure. In a realistic tight pitch structure, they estimate the overall delay variation to be 18%, of which 48% would be due to device variation and 52% due to interconnect variation.

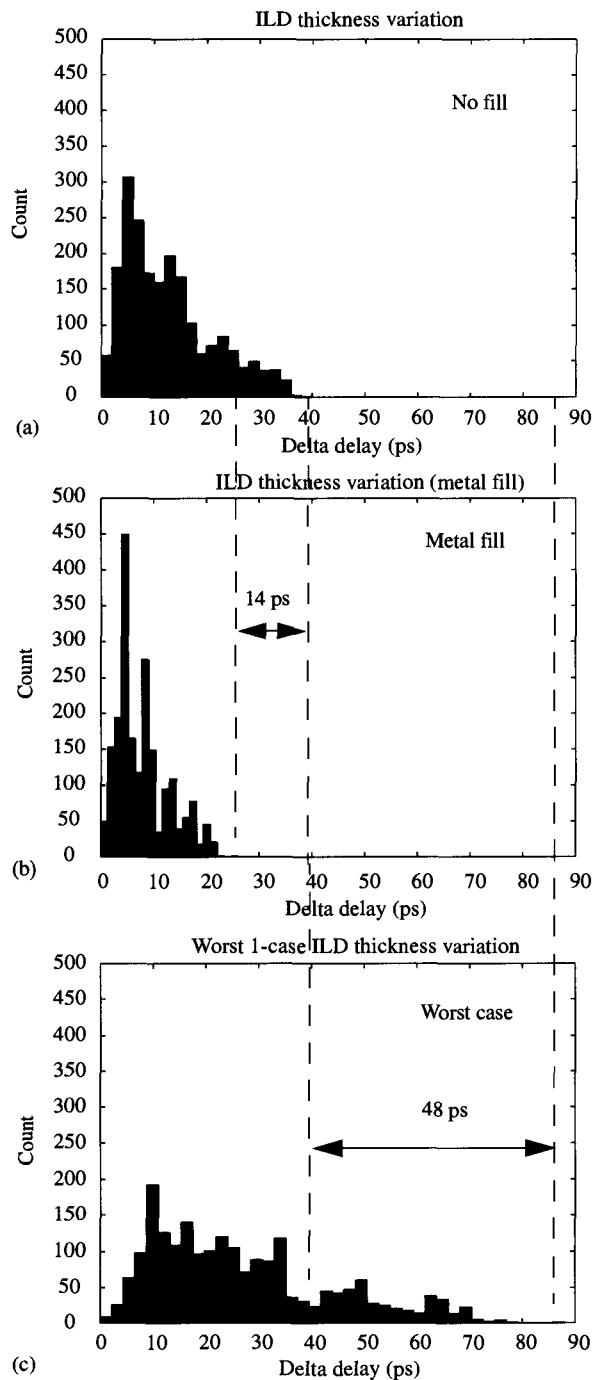
#### 6.5.4 Example: Deterministic Interconnect Variation Impact

In some cases a dominant source of intra-die variation can be modeled sufficiently to examine in detail the impact of that variation on circuit performance. In this example, global signal paths in a 1 GHz microprocessor design are examined based on a model of the interlevel dielectric (ILD) thickness variation created across the chip due to CMP dependencies [6].



**Figure 6.6** Modeled pattern-dependent ILD thickness variation profile for the top metal layer in a 1 GHz microprocessor design [6].

The predicted spatial function for ILD thickness,  $T_{ILD}(x, y)$ , is generated based on the circuit layout. In this case,  $T_{ILD}(x, y) = f(\rho(x, y, PL))$ , where  $\rho$  is the effective density of raised oxide topography “seen” by the polishing process. The spatial map



**Figure 6.7** The delay variation distribution with (a) the ILD thickness variation model, (b) ILD thickness variation after metal fill, and (c) the worst-case ILD thickness variation [6].

of the effective density is calculated as a moving average of underlying layout density given knowledge of the “planarization length”  $PL$  [16]. From this map,  $T_{ILD}$  is estimated as  $T_{ILD} = T_{ILD_0} + (\rho - \rho_0)h_0$ , where  $T_{ILD_0}$  is the nominal ILD thickness,  $\rho_0$  is the nominal or average effective pattern density, and  $h_0$  is the as-deposited step height of the oxide over patterned metal features. For a particular ILD layer, the resulting estimated ILD thickness variation for the 1 GHz microprocessor design is illustrated in Fig. 6.6.

Given  $T_{ILD}(x, y)$ , a distributed RC network is extracted for each global path to be considered, with a modification  $\Delta C$  made to the extracted capacitance for each segment based on the spatially dependent, rather than nominal, value for  $T_{ILD}$ . A timing analysis is then performed, and the difference in delay between the nominal design and that resulting from systematic spatial ILD thickness variation is computed. Figure 6.7 shows the delay variation distribution for 2100 global paths under three assumptions: (1) due to ILD thickness variation; (2) with ILD thickness variation reduced by the addition of metal fill [17]; and (3) under a simple worst-case ILD thickness variation analysis. As seen in the figure, worst-case analysis can be substantially pessimistic, and detailed knowledge and analysis of a variation source can offer the opportunity to further improve performance (e.g., clock speed) appreciably.

## 6.6 CONCLUSION

In this chapter we have examined descriptions, models, and methods for assessment of random, correlated, and spatial variation in integrated circuit fabrication. Analysis of process variation and its impact on both performance and yield are increasingly important for advanced high-speed circuits, and further work to develop and apply these methods will be needed.

## REFERENCES

- [1] P. Antognetti and G. Massobrio, *Semiconductor Device Modeling with SPICE*. McGraw-Hill, New York, 1988.
- [2] K. Bernstein, K. Carrig, C. Durham, P. Hansen, D. Hogenmiller, E. Nowak, and N. Rohrer, *High Speed CMOS Design Styles*. Kluwer, Boston, 1998.
- [3] P. Cox, P. Yang, S. Mahant-Shetti, and P. Chatterjee, “Statistical Modeling for Efficient Parametric Yield Estimation of MOS VLSI Circuits,” *IEEE Trans. Electron Devices*, vol. ED-32, 1985.
- [4] S. Director and W. Maly, *Statistical Approach to VLSI*. North-Holland, New York 1994.
- [5] D. J. Frank, Y. Taur, M. Ieong, and H.-S. P. Wong, “Monte Carlo Modeling of Threshold Variation due to Dopant Fluctuations,” *VLSI Technology Symposium*, pp. 169–170, June 1999.
- [6] V. Mehrotra, S. Nassif, D. Boning, and J. Chung, “Modeling the Effects of Manufacturing Variation on High-speed Microprocessor Interconnect Performance,” *IEDM*, pp. 767–770, 1998.
- [7] C. Michael and M. Ismail, *Statistical Modeling for Computer-Aided Design of MOS VLSI Circuits*. Kluwer, Boston, 1992.
- [8] S. Nassif, A. Stroywas, and S. Director, “FABRICS-II: A Statistical Simulator of the IC Fabrication Process,” *IEEE Trans. CAD*, vol. 3, 1984.
- [9] S. Nassif, “Within-chip Variability Analysis,” *IEDM*, pp. 283–286, 1998.

- [10] M. Orshansky, C. Spanos, and C. Hu, "Circuit Performance Variability Decomposition," *Fourth Int. Workshop on Statistical Metrology*, pp. 10–13, June 1999.
- [11] T. Park, T. Tugbawa, D. Boning, J. Chung, S. Hymes, R. Muralidhar, B. Wilks, K. Smekalin, and G. Bersuker, "Electrical Characterization of Copper Chemical Mechanical Polishing," *CMP-MIC*, pp. 184–191, Santa Clara, CA, Feb. 1999.
- [12] J. Power, A. Mathewson, and W. Lane, "MOSFET Statistical Parameter Extraction using Multivariate Statistics," *Proc. ICMTS*, pp. 209–214, 1991.
- [13] S. Sharifzadeh, J. R. Koehler, A. B. Owen, and J. D. Shott, "Using Simulators to Model Transmitted Variability in IC Manufacturing," *IEEE Trans. Semi. Manuf.*, vol. 2, no. 3, pp. 82–93, Aug. 1989.
- [14] R. Spence and R. Soin, *Tolerance Design of Electronic Circuits*. Addison-Wesley, Reading, MA, 1988.
- [15] B. Stine, D. Boning, and J. Chung, "Analysis and Decomposition of Spatial Variation in Integrated Circuit Processes and Devices," *IEEE Trans. Semi. Manuf.*, vol. 10, no. 1, pp. 24–41, Feb. 1997.
- [16] B. Stine, D. Ouma, R. Divecha, D. Boning, J. Chung, D. L. Hetherington, I. Ali, G. Shinn, J. Clark, O. S. Nakagawa, and S.-Y. Oh, "A Closed-Form Analytic Model for ILD Thickness Variation in CMP Processes," *CMP-MIC*, pp. 266–273, Santa Clara, CA, Feb. 1997.
- [17] B. Stine, D. Boning, J. Chung, L. Camilletti, F. Kruppa, E. Equi, W. Loh, S. Prasad, M. Muthukrishnan, D. Towery, M. Berman, and A. Kapoor, "The Physical and Electrical Effects of Metal Fill Patterning Practices for Oxide Chemical Mechanical Polishing Processes," *IEEE Trans. Electron Devices*, vol. 45, no. 3, pp. 665–679, March 1998.
- [18] P. Zarkesh-Ha, T. Mule, and J. D. Meindl, "Characterization and Modeling of Clock Skew with Process Variations," *CICC*, pp. 441–444, 1999.
- [19] P. Zarkesh-Ha and J. D. Meindl, "Optimum Chip Clock Distribution Networks," *ITC*, pp. 18–20, 1999.
- [20] J. Zhang and M. Styblinski, *Yield and Variability Optimization of Integrated Circuits*. Kluwer, Boston, 1996.

PART  
III

## CIRCUIT STYLES FOR LOGIC

Kerry Bernstein  
IBM

## 7.1 INTRODUCTION

The selection of circuit topologies implementing the logic of a high-speed microprocessor is central to the success of the product. Circuit style choices collectively define the characteristics that realize the intended design point. High performance is the desired attribute; but testability, power consumption, and ease of design are associated characteristics one implicitly selects.

It is rare when a single circuit topology accommodates well all of a design's logic functions; as we will discover, particular logical functions are best realized in silicon using specific device configurations. For that reason, the design methodology must be trustworthy with a variety of structures that exercise different device operation regions. For that reason, it is incumbent upon the design team to

- *Be aware of circuit alternatives that yield optimal logic realizations*
- *Choose compatible MOSFET and interconnect technologies*
- *Develop design/synthesis tools that encourage use of the best circuit*
- *Develop checking techniques to verify that circuit styles used will speak to each other in a "friendly" manner*
- *Verify that the resulting chip will meet the product's power, die size, performance, reliability, and design time expectations*

So, let's begin by examining the predominant circuits alternatives available to the engineer. Later we will survey the issues associated with choosing a structure.

## 7.2 NONCLOCKED LOGIC

Nonclocked CMOS, which includes *static combinatorial CMOS*, *differential cascode voltage-switch logic*, and *pass-transistor logic*, remains the predominant circuit approach to realizing logical function, for a number of reasons:

- Low power consumption
- Ease of automated synthesis
- Straightforward delay rule timing

- Inherent reliability and logic noise immunity
- Process variation and defect tolerance
- Migratability into successive technology generations
- Deterministic diagnostic capability

The demand for performance is leading to the use of more aggressive circuit families, but it is reasonable to expect that unclocked, static circuitry will be onboard CMOS designs in some quantity on future microprocessors.

Nonclocked circuits provide high design reliability because nodes that determine the value of successive stages are constantly maintained, never left to float. Direct control provides implicit logic noise immunity. Careful PFET/NFET device size ratio tuning achieves desired *switch points* and *unity gain points*,<sup>1</sup> resolving most **delay** noise concerns. Because nodes are strongly held at all times in a complementary fashion, nonclocked CMOS is also especially forgiving to defects and process variation. Since control devices receive full gate voltage even after its capacitive load has been charged, “on” MOSFETs in static circuitry remain inverted, and charge is readily available as needed to overcome coupling events, or short-circuit paths/leakage mechanisms caused by minor defects. This high degree of tolerance enables scaling of nonclocked designs readily into successive generations of the technology.

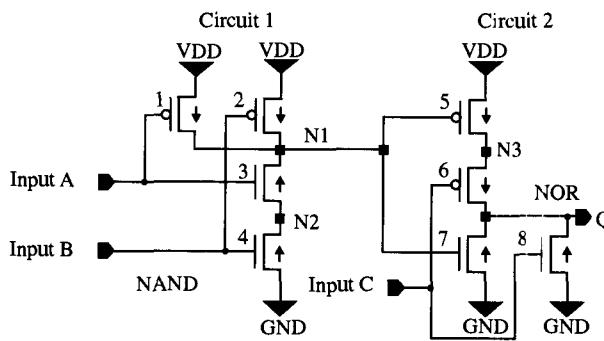
Three common nonclocked structures traditionally of interest to high-speed circuit designers are reviewed in the following section.

### 7.2.1 Static Combinatorial CMOS Logic

A *static combinatorial CMOS logic* path executing the NAND function followed by the NOR function is shown in Fig. 7.1.

Static combinatorial logic circuit operation makes use of *push-pull action* [5], [4]. Referring to Fig. 7.1, with input B high, transitioning high input A pulls circuit 1 (NAND) output node N1 and node N2 to ground. In the subsequent circuit 2 (NOR), node N3 rises to voltage  $V_{DD}$ . With input C low, NOR output Q switches high.

Some notable traits characterize static combinatorial CMOS behavior, and are described below.



**Figure 7.1** Classic static combinatorial CMOS logic [6].

<sup>1</sup> See p. 121.

### 7.2.1.1 Transfer Function

*Transfer function* refers to the relationship between input and output voltage for a given stage. The transfer function for a simple inverter is shown in Fig. 7.2. Two important parameters can be measured:

1. *Unity gain point* (UGP) refers to the point the transfer function where the first derivatives of the function equal  $-1$ , i.e., where a tangent to the curve has a slope of negative one. The given circuit will attempt to attenuate inputs less than the lower UGP, and amplify inputs higher than the lower UGP.
2. *Switch point* (SWP) is that point where  $V_{in} = V_{out}$ . The switch point potential is often strongly skewed high or low by device size selection to hasten stage transitions in a given direction.
3. *Noise margin* (“NM-H”, “NM-L”) is the difference between the *least positive up level* (LPUL) of the preceding stage and the upper UGP of the given stage, or the *most positive down level* (MPDL) of the previous stage and the lower UGP of the given stage.

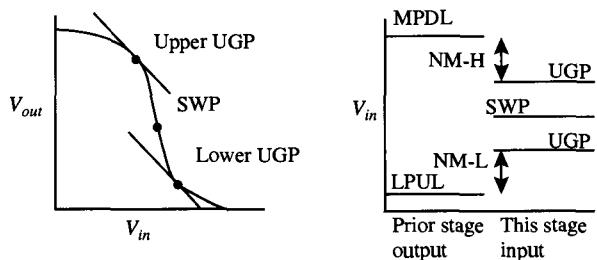


Figure 7.2 Inverter transfer function and noise margin.

### 7.2.1.2 Delay Variability

Differing input patterns (instructions and data) in a microprocessor implicitly invoke separate circuit responses in moving through a logic partition, determined entirely by the operations the design needed to accomplish in that path. Each of these logic vectors invokes a unique composition of device and interconnect RC delays. It follows then that the data/instruction stream is responsible for a predominant source of design delay variation.

A subtle effect causing delay variability in static logic is use history. Referring again to Fig. 7.1, if a previous cycle left the capacitance associated with node N2 charged high, then this circuit's evaluation delay in a subsequent cycle gated by input B will take a little longer to evaluate than if node N2 had previously been discharged low.

A second, often overlooked source of delay variation is active fan-out load. Two types of active load should be considered, and are not unique to static logic:

1. The portion of the output load associated with the gates of transistors in subsequent stages will vary, depending on whether the transistors are on or off. As the inversion channel forms in the device, the gate-substrate capacitance drops, as the effective dielectric thickness increases. Eventually it is replaced by gate-channel

capacitance. In Fig. 7.1, for example, the load on circuit 1 presented by the fan-out to devices 5 and 7 will depend on the original voltage on node N1.

2. That portion of the fan-out associated with the interconnect will change based on the coupling of that wire to neighboring wires which may also be transitioning. Again referring to Fig. 7.1, if N1 is going high and its neighbors are going low, the effective lateral wire capacitance is  $2 \times$  what it would be if the neighboring wires were not moving. Timing loops often may be disrupted by this source of variation, and challenges accurate delay rule generation and synthesis.

### 7.2.1.3 Device Size and Switching Threshold

Two important design metrics describe circuits.

1. *Alpha ratio* is a way of describing the gain in capacitance per stage. Effective logic must support fan-out load. If the load is too high with respect to the stage driving it, performance suffers. If the load is too low relative to the given stage, power and area is wasted. Alpha ratio is the total output capacitance on a given stage divided by its total input capacitance.

**Rule of thumb:** Alpha ratio of 2.7 (Euler's constant) produces minimum power-delay product.

2. *Beta ratio* is a measure of the relative strength of pull-up to pull-down for a given inverter stage. Specifically, it is the ratio of a given stage's PFET's width/length quotient to its NFET width/length quotient, at the wafer level. As the value of beta gets larger, pull-up strength increases and the switchpoint potential of the stage rises.

**Rule of thumb:** Since PFETs have approximately half the transconductance of NFETs per unit width/length, a stage with a beta ratio of 2 has roughly balanced pull-up with pull-down strength, and has a switch point at  $V_{DD}/2$ .

The noise immunity of a given circuit is related directly to its beta ratio. If the stage's beta ratio is excessively high or low, then noise on a signal has an easier time of achieving the stage's switch point and causing the propagation of a false signal. The motivation for strongly biasing the switchpoint, of course, is reducing the voltage swing and delay necessary to propagate a signal through that stage.

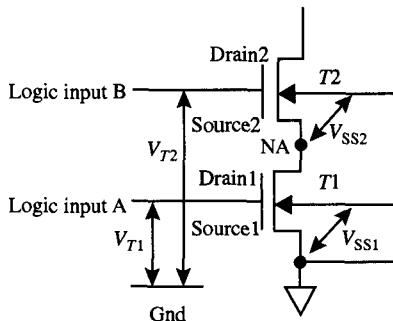
### 7.2.1.4 NAND Design

NAND stacks are one of the most common logic elements, owing to its leveraging of NFET superior transconductance. It's worthwhile, then, to spend a little time looking at issues specific to the NAND.

**Rule of thumb:** Approximately 75% of all random static combinatorial logic structures are NAND stacks.

*Body effect* is very important to the NAND. Referring to Fig. 7.3, the drop across T1 elevates the potential of node NA, source of device T2. This increase in T2's source-substrate bias increases its threshold voltage and depresses performance. Because of this, NAND stack heights are typically limited to three or four devices.

The discussion above on alpha device sizing is equally applicable to NAND structures, if an equivalent width-to-length ratio for the stack is calculated, and then treated as a single device.



**Figure 7.3** Body effect in sequential devices [6].

Finally, the *ordering* of inputs in a NAND stack affects stage delay. Referring to Fig. 7.1, if input B is known via timing to arrive before input A, then it is preferable to couple input B to the lower device 4. The capacitance of node N2 is then discharged early, and the amount of capacitance which needs to be discharged by the latest arriving signal A is then minimized. Similarly, because of the increase in total capacitance seen as one moves down the tree, it was common practice to *taper* device widths out as one moved down the tree [1], [2]. Because of the resulting layout area, capacitance and increased body effect penalties with scaling, tapering is no longer considered an essential design practice in deep submicron VLSI.

#### 7.2.1.5 False Switching

Nonclocked circuits respond instantaneously to changes in their inputs. If inputs to a nonclocked circuit do not change, then its outputs do not toggle, assuring that low-activity-factor paths return lower nonswitching power consumption. Alternately, because static circuit response is instantaneous and circuit inputs do not all have identical arrival times, static circuits exhibit *false switching* behavior. Intermediate values are propagated along the given net as the logic block derives its solution. This unnecessary activity causes extra DC (crossover) and AC (capacitance) power consumption, commonly referred to as *glitching power*.

**Rule of thumb:** Caused by paths of varying delay providing inputs to a given stage, false switching has been estimated to consume 15% of overall chip power.

It follows, then, that in timing critical path delay, margin must be provided for the resolution of every stage assuming all possible false switching along the path. Power distribution must anticipate the noise and current consumed by glitching.

#### 7.2.1.6 Crossover Current

As inputs to a static gate transition from high to low or low to high, a short-circuit current, or *crowbar current* flows through both devices during the interval of the transition when both devices are on.

**Rule of thumb:** Power consumed by crowbar current is a function of the input transition's slew rate, but can consume approximately 10% of a static chip's total power.

Crowbar currents can be minimized by sizing devices appropriately. The current flows during the duration of the interval during the output voltage swing when both devices are on; the crowbar current can be reduced by ensuring that the rise and fall times are approximately equal [3].

**Lower thresholds** are useful in reducing cycle limiting path delay. Unfortunately, as NFET and PFET device thresholds are reduced, the interval when the pair are both conducting increases, causing *additional* power to be lost to crowbar current. Nonetheless, noise margin can be preserved by ensuring via device size ratio that the circuit's switch point remains unchanged at the lower threshold. Many studies of the relationship between performance, noise margin, and crossover current appear in the literature [4].

### 7.2.1.7 Overall Static Combinatorial CMOS Power Consumption

Although a nonclocked function typically consumes less total power than synchronous approaches, the propagation of logic along its path tends to charge more capacitance than its clocked counterpart. The extra capacitance is associated with the need to replicate “NFET-tree” logical functions in the “PFET trees,” in many cases tripling the fan-in load. Static circuits save power, on the other hand, by avoiding the need for each **logic stage** to be clocked, and by avoiding DC current paths. Power is minimized when circuitry utilizes the smallest possible devices, reducing the capacitive load of preceding stages. Static CMOS performance is enhanced with the use of lowered threshold voltages, “library” versions of the same logical circuit at various output power levels, and by optimized device sizing via “tapering” and “betaratio.”

## 7.2.2 DCVS

*Differential cascode voltage switched logic* (DCVSL) is the foundation of many contemporary higher-speed circuit structures [5]. Inspired by Domino CMOS, DCVS influenced subsequent differential circuit design. There are a number of DCVS structures described in the literature, using pairs of *differential* logic inputs to flip a static cross-coupled device pair and store an output state. Differential circuits require logic structures whose evaluated result on a given side of the logic design is exclusively the complement of the evaluated result of the other side. For example, in Fig. 7.4(a), the left side of the DCVS structure shown produces the **NAND** and the right side the **AND** of inputs A and B.

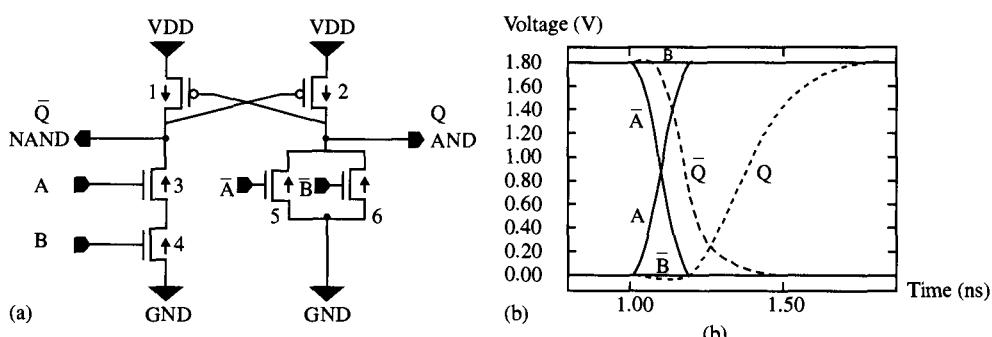


Figure 7.4 Basic DCVSL AND gate with PMOS cross-coupled loads [6].

Logical arrangements, or *trees*, of stacked evaluation devices potentially couple the circuit's output node to ground (or to a developed signal  $V_{DD}$  in some structures), conditional on the result of the evaluation.

Referring to Fig. 7.4(b), with input B high and input  $\bar{B}$  low, transitioning-high input A and transitioning-low input  $\bar{A}$  are fed to the differential evaluate tree, latching node  $\bar{Q}$  low, and node Q high.

Differential cascode voltage switching achieves improved logic density by evaluating complex logic trees in one delay stage. Further efficiency is achieved in the elimination of large PFETs from each logic function executed in the tree. Boolean functions are implemented in NFETs only; the PFETs serve solely as pull-up devices. Portions of independent DCVSL evaluation trees may also be combined to further reduce device count and fan-out load; a number of effective algorithms have been developed to do this. Static differential cascode-voltage switched logic offers implicit noise immunity at each stage, due to its cross-coupled nature.

Other DCVS styles include *differential split level logic*, and *cascode non-threshold logic* [5], [6].

### 7.2.3 Pass-Gate Logic

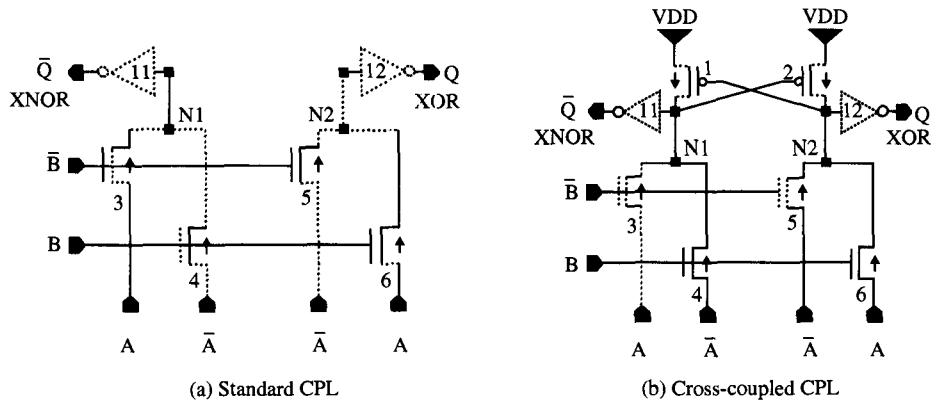
Since *pass gates* are critical components in both the one-device DRAM cell and the six-device SRAM memory cell, it follows that they could provide important functionality to VLSI logic. Pass structures indeed reduce logic delay by coupling an input signal to a selected net, rather than evaluating and redriving that signal. With lower load requirements, pass gates are indeed very fast, exceeding static performance by 20–50%. They do, however, carry additional liabilities, which must be evaluated, including:

1. *Limited fan-in capability.* Current to be discharged to ground through a pass-gate structure must be limited in order to achieve acceptable down levels with reasonable delay.
2. *Excessive fan-out.* The output inverter providing logic signals to subsequent pass gates must be sized to provide sufficient high and low levels to all the paths it serves. If the pass-gate fan-out is excessive, resulting performance will be disappointing.
3. *Additional noise vulnerability.* Noise on signal lines caused by interconnect coupling can be propagated through a pass-gate circuit via device gates *or* sources, doubling the opportunity for noise-driven disturbances of the output data.
4. *Rail voltage offsets/biases.* Poor supply levels can create voltage differentials on inputs. This bias reduces overdrive, decreases noise immunity, and aggravates the known body effect in the pass-gate structure.
5. *Decode exclusivity.* Pass gates, when used as multiplexers, must be provided gate inputs which preserve the exclusivity of the various paths coupled to the mux.
6. *High-level restoration.* Exclusive use of NFET devices in the pass-gate logic path passes logic “high” levels of  $V_{DD} - V_T$ . This voltage, presented to the output buffer, causes its PFET device to not turn fully off, allowing leakage current and loss of noise immunity.
7. *Body effect.* Positive source-to-substrate voltage created in the pass gate reduces the overdrive of the device, and can slow down the circuit.

Because of the above considerations, many design teams use only simple NFET-PFET pass gate pairs, or *Transmission Gates*, to transfer levels without loss of signal voltage. Many incarnations of NFET-only pass-gate-based logic appear both in literature and product, however. Let's examine two popular pass-gate structures:

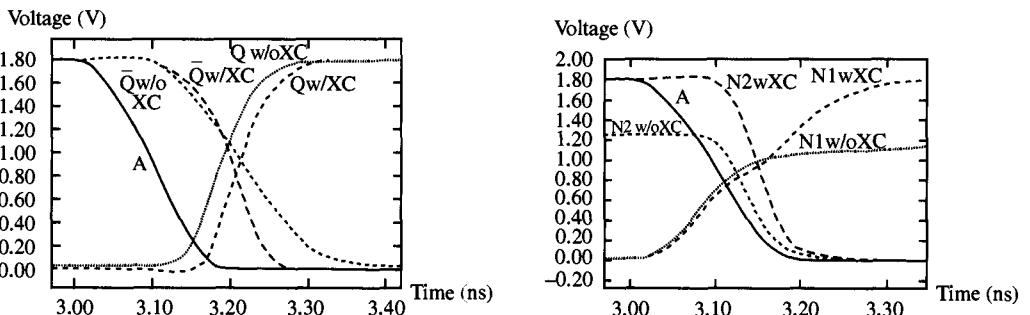
### 7.2.3.1 Complementary Pass-Gate Logic (CPL)

Despite continued innovations using other structures, *complementary pass-gate logic* remains among the simplest, fastest, and most frugal of the circuit families using transistors in pass-gate configurations [5], [7]. Two CPL implementations of an XOR are shown in Figs. 7.5(a) and (b).



**Figure 7.5** Complementary pass-gate logic implementations of XOR: standard CPL (a); cross-coupled CPL (b) [6].

For one of the two inverting output buffers in Fig. 7.5(a) receiving gate drive of  $V_{DD} - V_T$ , its PFET is never entirely shut off, resulting in DC current and power consumption in the inverter producing the low output side. Referring to Fig. 7.5(b), logical control inputs B and  $\bar{B}$  gate A and  $\bar{A}$  into differential nodes N1 and N2. Optional cross-coupled PFET devices 1 and 2 shown in Fig. 7.5(b) complete the swing of node N1 or N2, whichever is high, from  $V_{DD} - V_T$  the rest of the way up to  $V_{DD}$ . Inverting buffers I1 and I2, tied to N1 and N2, respectively, develop outputs  $\bar{Q}$  and Q. In Fig. 7.6, the



**Figure 7.6** CPL operation of circuits shown in Fig. 7.5; external (a) and internal (b) node behavior [6].

difference between standard CPL without cross-coupling (“w/o XC” in plot) and cross-coupled CPL (“w/XC” in plot) is apparent.

The latch structure formed by PFETs 1 and 2 in Fig. 7.5(b), however, reduces performance by adding extra evaluate capacitance. Another alternative, reducing the threshold of NFET devices 3–6 in Fig. 7.5(a), to near-zero voltage brings the high evaluate node closer to  $V_{DD}$ , but increases process complexity and noise sensitivity. Replacing inverters I1 and I2 with “keeper” feedback buffers commonly found in dynamic circuits is a third option. Keepers are described in Section 7.3.1.

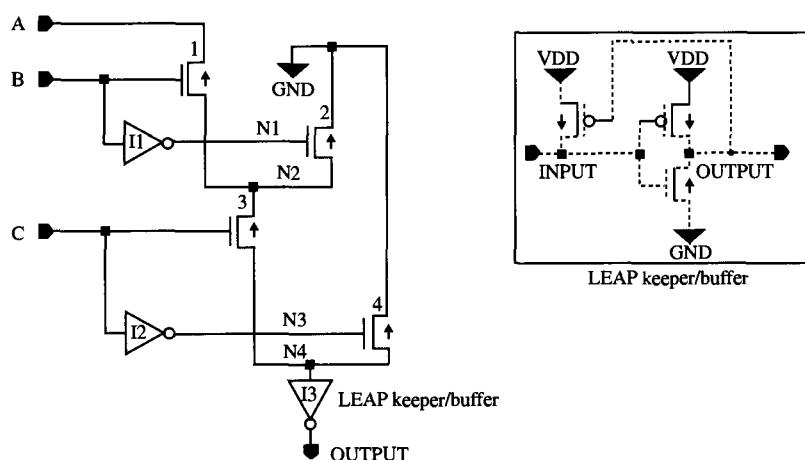
### **7.2.3.2 LEAN/LEAP Logic**

*“Top-down pass-transistor logic,”* disclosed as a key component in commercial LSIs announced by Hitachi [8], proposes reduction in size of a random logic library through the use of a “lean” set of simple, flexible pass-gate cells which can be used bidirectionally, and whose inputs could be connected to signals or either supply rail if needed for a particular logic voltage level. *Lean integration using pass gates* (LEAP) concentrates on synthesizing the function of full logic blocks rather than individual logic functions. The LEAP realization of a logical three-way NAND is shown in Fig. 7.7.

LEAP realization of a logical three-way AND is shown in Fig. 7.77.

LEAP logic addresses the issues needed to implement pass-gate logic in an automated design methodology. Traditional design methodologies attempting to execute discrete functions in pass gates realize limited advantages over static logic. Generally, in LEAP, discrete logical functions *cannot readily be recognized!* Flexibility comes from the simplicity of its library, comprising potentially only three separate cells, and the ability to use  $V_{DD}$  and GND as additional logical inputs.

Functions realized in LEAP form trees, comprising sequential two-way multiplexers. Paths appear similar to simple NFET pass-gate logic, but integrate *whole functions* via automated logic synthesis software, rather than independent logic cells. Logic is not differential, in contrast to CPL. Because the logic is single-ended, inverted signals are available only at the end of each tree, at the output of the inverter, or must be made available by inverting selected intermediate nodes. The keeper-like output buffer is essential to recapture full rail swing, and is not unique to LEAP.



**Figure 7.7** LEAP pass-gate logic of NAND function [6].

A number of other pass-gate-based structures have been used successfully [7]. They include *Swing-Restored Pass-Gate Logic*, *DCVS with the Pass-Gate*, *Double Pass-Transistor Logic*, *Energy Economized Pass-Gate Logic*, and *Push-Pull Pass-Gate Logic*.

## 7.3 CLOCKED LOGIC

*Clocked logic* families are those in which the circuit executes the intended logical function during a specific evaluate period, followed by a precharge or reconditioning interval. False switching is eliminated in clocked paths, as reconditioning following a transition will not occur until the next phase. Clocked circuits get “one chance” only!

Unlike the static approaches described previously, clocked logic data states are typically developed in only one direction. A single “device polarity” (usually NFET) is predominant in the evaluate path. Since the device size is optimized for that transition, capacitance can be dramatically reduced. There are less devices to drive, and those devices tend to be smaller. Also, clocked dynamic circuits employ lower switch points than static designs.

The cost of clocked logic performance is a restraint for some product applications. Signal evaluation is in an implicit race with the clock at each stage. Logic clocking also increases power consumption, accounting for 20% or more of total chip power.

Additional power is spent in duty factor, as each stage must evaluate and recharge even if the data does not change. Finally, dynamic logic is intrinsically less stable. The logic summand node is vulnerable during evaluate to a number of process-induced and design-induced failure mechanisms. Clocked structure performance is very impressive, however, and can provide the delay improvement of a full CMOS technology migration. We briefly look at two clocked circuit styles, covered in more detail later in the text.

### 7.3.1 Domino Logic—Single and Differential

Dynamic dominos are composed of precharge, evaluate, and buffer functional blocks [9]. Domino logic has been very effective in high-speed chips such as the **Alpha 21164 Microprocessor** [10]. Referring to Fig. 7.8(a), during precharge clock PC

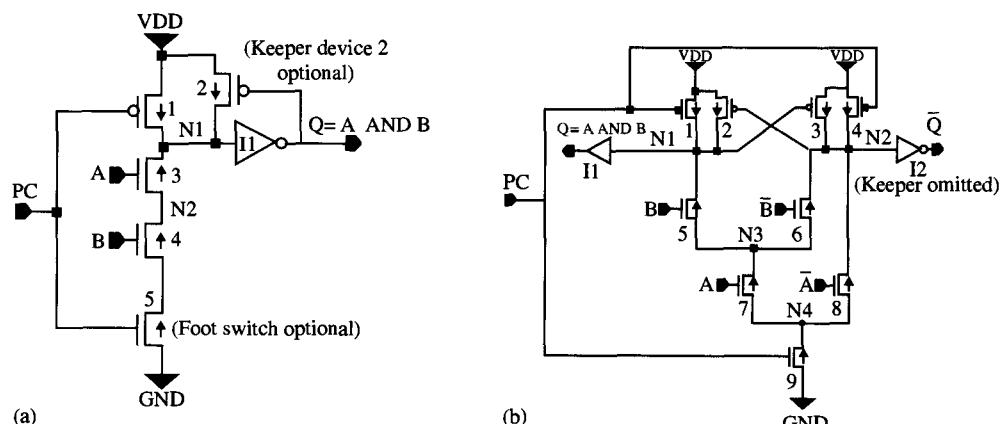


Figure 7.8 (a) Dynamic domino structure, single-ended; (b) dual rail [6].

is low. PFET device 1 charges node N1 to  $V_{DD}$ , driving output  $Q$  to ground and turning on “keeper” PFET device 2. Keeper devices are intended to replace current lost to leakage, but are generally too slow to provide active noise immunity. “Foot switch” NFET device 5 is off, interrupting the path to ground during precharge of the evaluate block. The evaluate block in Fig. 7.8(a) is represented by devices 3 and 4. When PC transitions high, the circuit switches from precharge to evaluate mode. With logic inputs A and B high, node N1 is discharged to ground, switching the output of inverter buffer I1 high and turning off keeper PFET device 2.

**Rule of thumb:** Typical domino keepers have an effective width/length ratio of 5–20% of the effective width of the evaluate tree. Typical output buffers have a beta ratio of approximately 6/1 (PFET/NFET), pushing its switchpoint high in voltage for higher performance, but reducing its noise margin.

Dynamic dominos can contain a substantial amount of logical “width” in the NOR direction, and “depth” in the NAND direction. This allows significant “logic gain” along a path of dominos. Further advantages are found in sequences of dominos: if logic inputs A and B are outputs of other domino circuits using the same clock, then during precharge, they are also low and the footswitch, NFET device 5, which increases delay by adding resistance to ground, may be omitted.

*Crowbar currents*, a known static combinatorial CMOS characteristic (see p. 123), are not usually an issue in dominos, but may be occasionally observed. Referring to Fig. 7.8(a), if inputs A and B become valid and high before the transition of clock PC into evaluate mode, classic crowbar current will pass through the tree when the clock does transition. This crossover current can be substantial if the stage does not use a foot switch. Mechanisms giving rise to domino crossover current include:

1. Clock skew into a given stage, with respect to the stages providing it logical inputs.
2. Supply or process tolerance creating across-chip delay variation between stages.
3. Non-monotonic inputs.<sup>2</sup>
4. Premature input setup to a given stage.

In normal operation, *crossover power* is not expected, as domino inputs will be provided by other domino stages, all of which are gated by the same precharge/evaluate clock phase. Dominos behave as “one-shots,” and as such do not consume *false switching power*.

*Capacitive charge sharing* is also an issue in dynamic domino. If input A goes high but input B remains low in Fig. 7.8(a) after clock PC goes high, and if node N2 was discharged low in a previous cycle, then even though there is no path to ground, the precharge established on node N1 will now capacitively divide between nodes N1 and N2. At best, this voltage drop erodes noise immunity of the structure; at worst it can cause a false switch of the output signal.

Single-ended domino is an incomplete logic family; the structure is incapable of inverting logic. Inverted logic must be provided by propagating a separate path originating from the complement output of the last latch. Dual rail, or *differential* domino propagates true and complement logic together to execute complete logical functions. In Fig. 7.8(b), a dual rail differential domino achieves the same two-way AND logical function, but provides current replacement and noise margin by driving feedback from

<sup>2</sup> *Non-monotonic inputs* refers to signals that may be valid high or low, originating in other topologies.

the complement side's summand node, rather than off the same side's inverted summand node. Thus, the delay required to overcome the switchpoint of the keeper half latch is avoided, without sacrificing charge replacement and noise immunity during evaluate mode. The performance improvement associated with eliminating the half latch has been shown to exceed the performance penalty associated with the additional cross-coupled capacitance of devices 2 and 3. *Compound domino*, *multiple output domino*, and *noise-tolerant precharge domino* are a few other single-ended domino alternatives used in the industry [6]. Differential domino logic which appear in the literature include *Dynamic Differential Split-Level Logic*, and *Pseudo-Clocked Domino* [6]. **Domino-like Clocked Pass Gate structures** have also been advocated, namely *Clocked Pass-Transistor Logic* and *Sense Amplifying Pass Transistor Logic* [6].

### 7.3.2 Latched Evaluate Logic

Increasing a latch's function by asserting logic along the latch's path to ground introduces important features not normally associated with clocked differential logic. These properties can be very useful:

1. Results developed in each stage are fully latched, allowing the circuit to also act as a master or slave latch in a transparently latched logic scheme.
2. Stage delays are all roughly equivalent, independent of logic content.
3. A latch can assist in evaluate pull-down, allowing higher stage logic content per stage.
4. Sense amplification and additional noise immunity add design reliability.
5. The need for an output buffer is eliminated.

The static-like latch adds additional devices and associated interconnects. While these resulting structures tend to be larger and slower than their dynamic equivalent, these latches would have been logically transparent, and incorporate additional functionality for very little extra delay.

*Sample-Set Differential Logic* (SSDL), shown in Fig. 7.9, is an example of such a clocked latch structure [11]. Sample-set differential logic uses cross-coupling to reduce

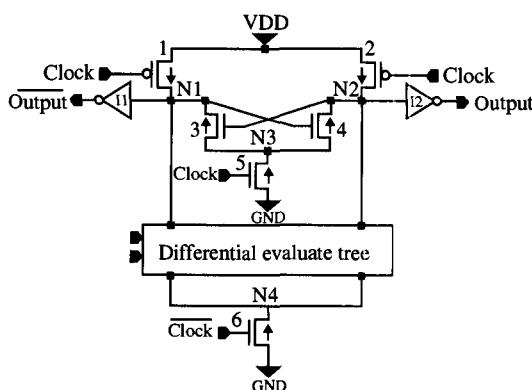


Figure 7.9 Sample-Set Differential Logic [6].

the voltage swing of the evaluation logic, and to hasten the “decision” delay of the differential evaluate tree. Here, cross-coupling acts as a simple regenerative differential amplifier, and completes the generation of output begun by the complex tree. During the “sample” interval, the clock is low, and voltages on nodes N1 and N2 are set differentially through the current path established by PFET device 1 or 2, the differential evaluate tree, and NFET foot device 6. When the clock transitions high, devices 1, 2, and 6 turn off, and differential discharge is arrested; NFET device 5 is on, and the small differential voltage across nodes N1 and N2 is sensed and fully developed by cross-coupled devices 3 and 4. The outputs are inverted and buffered by inverters I1 and I2.

*Swing-Restored Pass-Gate Logic, Enable/Disable Differential Logic, and Switched Output Differential Structures* are a few of many latch-based domino-like topologies which have appeared in the literature [6].

## 7.4 SELF-TIMED AND ASYNCHRONOUS LOGIC

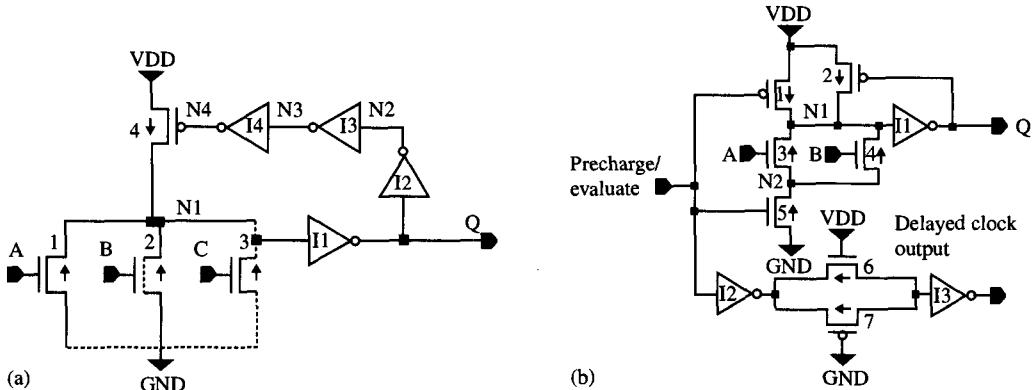
The microprocessor engine used in present high-performance consumer computing contains predominantly clocked, synchronous logic. The system, in each clock cycle, moves from one *machine state* to the next by steering the contents of the chip’s many registers through a sequence of on-chip circuit structures which achieve the desired logic. The result is again latched and carried across the cycle time boundary to define machine state of the subsequent cycle. While this technique has served our industry well since the advent of electronics, it has its shortcomings.

- Maximum operating frequency is limited by the longest path.
- Maximum operating frequency must be reduced to provide margin against process variation, voltage supply tolerance, and operating temperature range.
- Power is wasted because clocks continue to run during inactive periods, and internal performance must always exceed clock frequency during active periods.

*Asynchronous* logic is a powerful emerging alternative. It dispenses with structured timing boundaries, but **must** use either unclocked or self-coded/self-timed circuit topologies. Unclocked functions may be realized in static combinatorial structures as described in Section 7.2.1; self-coded/self-timed families are important not only in asynchronous systems, but are useful in conventionally timed systems as well. Self-timed logic is extremely fast, but can be difficult to time. Inputs to a given stage must arrive within a specific interval, made even more difficult by high process sensitivity. Timing “*Interlocks*,” and *Muller C* elements [12] can relieve some of this difficulty. Diagnosing failure is complicated by signal transience. Alternative structures resolve this by gating the reset inverter chain during test to support static evaluation and diagnostics. Finally, **logic** noise may be propagated along with data, and so must be considered during layout. Two different self-timed domino structures are explored below.

### 7.4.1 Self-resetting CMOS

Self-resetting CMOS (SRCMOS), shown in Fig. 7.10(a) below closely resembles conventional domino structures, with the addition of devices which automatically



**Figure 7.10** Self-coded logic: self-resetting CMOS (a); clock-delayed domino logic (b) [6].

return the circuit to the precharge mode after enough time has been allowed for the signal to propagate through the circuit [13]. Logic values exist and move through SRCMOS structures as *waves* or *bubbles*, rather than as voltage *levels*. Unlike other styles, the output values are available for only a limited duration; the restore waveform chases the logic evaluation wavefront through the logic path, through a less tortuous path. The input signal to a SRCMOS stage is a pulse rather than a level, whose duration must be less than the reset delay, in this example the propagation delay through inverters I1–I4 in Fig. 7.10(a).

#### **7.4.2 Clock-delayed Domino**

Clock-delayed (CD) domino eliminates the fundamental monotonic signal requirement imposed on standard dynamic domino by propagating a clock network in parallel to the logic, providing a dedicated clock to each logic stage [14]. Clock-delayed domino logic was mainly used in **IBM's 1 GHz microprocessor** test site, the highest performance microprocessor reported to date [15]. A simple CD domino stage is shown in Fig. 7.10(b). A standard two-way CD'ed OR is executed in single-ended domino by transistors 1–5 and inverter I1. In addition, a clock-delay element formed by inverters I2 and I3, and transmission gate transistors 6 and 7 are required. When the precharge/evaluate transitions high, device 5 turns on, and the domino circuit evaluates the OR of logical inputs A and B. After inverter I1 has fully developed output  $Q$ , the delay circuit produces a delayed clock output provided to subsequent stages. Each stage's clock is tuned to its delay by appropriately sizing the width/length ratios of devices 6 and 7.

CD domino can provide both inverting and noninverting functions, without the need for differential pairs or cross-coupled latches. Inverter II in Fig. 7.10(b) is not essential; the clock always arrives *after* the interval when evaluate should have been completed. Two timing schemes have been proposed.

1. A single delayed clock is coupled to every domino gate requiring a given delay, across multiple logic paths. The clock delay is large enough to allow evaluation of the slowest of the gates.

2. Alternately, *each* domino gate can have generated in parallel a clock, with delay slightly longer than the slowest logic path through that stage.<sup>3</sup> Extra devices are required, but performance is improved. In addition, lower peak currents and less simultaneous switching is observed, as clock action becomes less synchronous.

Clock delay specific to the path ensures that all inputs have stabilized and that the domino has had time to complete its evaluation. Design margin must be added to the clock delay to guarantee function across process, circuit, and application variations.

## 7.5 IMPLEMENTATION ISSUES

As we saw in the last three sections, there are not only a number of different circuit families to choose from, but multiple flavors within each style. From this array of possibilities, how does the circuit designer ever decide what structure to implement? The answer is that it is not as daunting a task as one might expect. By keeping in mind the product's performance, power, reliability, area, and cost/price boundary conditions, the number of eligible alternatives quickly narrow. Let's examine some considerations involved in selecting the best circuit style, and then benchmarks used to compare them to one another.

### 7.5.1 Circuit Style Selection Criteria

A few fundamental principles govern chip critical path performance and power consumption. Since performance in CMOS reduces to RC delay, low "on" resistance means wide gates and more capacitance, which means higher power. So, then, what makes for a fast, effective circuit implementation of a logic function?

- *When the function can use small NFETs and a minimal number of PFETs.* Because PFETs have half the transconductance of NFETs with the same gate capacitance, it hurts to include them in critical paths. Small devices also reduce loading on the driver.
- *When the resulting layout is tight, and does not require long wires.* Tight layouts are self-shielding for noise and reduce the interconnect fan-out capacitance presented to the driver.
- *When the fan-out capacitance is minimized.* Large loads require preceding logic stages to progressively step up in capacitive gain, as described by Carver Mead [4]. Logic implementations that call for large fan-out tend to escalate design area and power consumption, as each switch must drive larger devices.

There is little algorithmic "order to the universe" steering the engineer into a preferred style for a given function. Instead, selecting a schematic means looking in the toolbox to see what works best. In a power-conscious design, some performance may be sacrificed to limit power consumption, while in, say, a high-end server application, the highest performance is pursued with little regard for resulting power. Here are a few of the considerations.

<sup>3</sup> Clock delay adjustment is accomplished by varying the size of transmission gate devices 6 and 7, or by adding additional transmission gate devices in series with devices 6 and 7.

### 7.5.1.1 Static Circuit Selection

Static CMOS has been the default electronic logic design structure. It is a complete logic family; both true and complementary outputs of any required combinatorial logic function can be generated anywhere along the logic path. It is low power, in that it passes very little DC current and requires no clocking when inactive. It is quite resilient to logic noise, and it is flexible. Logic can be synthesized in static circuits via automated tools quite readily.

Combinatorial CMOS circuitry is the preferred **unclocked** circuit style when heavy loads must be driven via successive stages of capacitive gain. Individual logic stages also do not require clocking. For these reasons, static applications include:

- Random combinatorial logic with low logical width
- Automated synthesis methodologies, such as in *application-specific integrated circuits (ASICs)*
- Unclocked heavily loaded signal propagation
- Asynchronous operations, as static circuitry requires no preconditioning

By selecting this style, however, the designer sacrifices area, compromises performance somewhat, and creates a noisy environment for other styles on chip. While the style is robust to logic noise, it remains vulnerable to delay noise arising predominantly from coupling wires.

### 7.5.1.2 Dynamic Circuit Examples

Optimum match to the task is often achieved when the logic function “resembles” its specific circuit implementation; that is, its logic representation is suggestive of a specific circuit structure. Consider the following simple example:

The conventional four-way OR built in static CMOS and shown in Fig. 7.11 is about as logically wide a circuit as one would want to build in combinational logic, due to the height of the PFET stack. In dynamic logic however, wide ORs are naturals; four-way ORs barely tax delay compared to its static embodiment. Single-ended dynamic domino representation of the same function is shown Fig. 7.12.

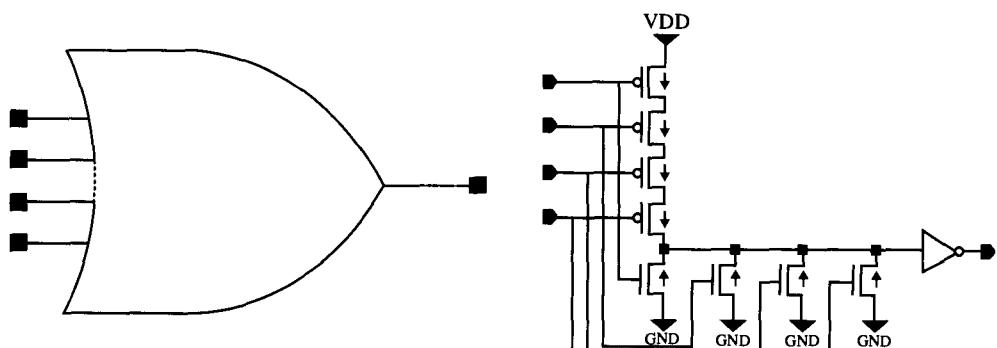
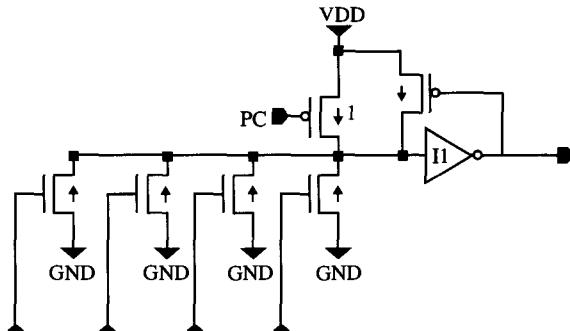


Figure 7.11 Conventional static four-way OR.



**Figure 7.12** Dynamic domino implementation of the four-way OR.

From Fig. 7.12, it is apparent that dynamic dominos are very good at ORs: there is no series PFET usage, the logical width is easily accommodated by the circuit, and the wiring requirements to the logical inputs are minimal. Don't we wish all the selections were this straightforward! The designer, however, does choose to buy into the logic evaluate/precharge clock overhead, added noise vulnerability, higher switch factor power, and monotonic evaluation.

Dynamic logic is an effective means of evaluating heavy loads, as the loads may be precharged high previous to the critical path operation, then evaluated in their faster direction during the cycle-limiting evaluation delay.

In summary, dynamic logic quite effectively implements the following logic functions needed in high-speed microprocessors:

- Zero detect circuits
- Adders
- Heavy load propagation
- Wide-ORs, as we saw in our example above

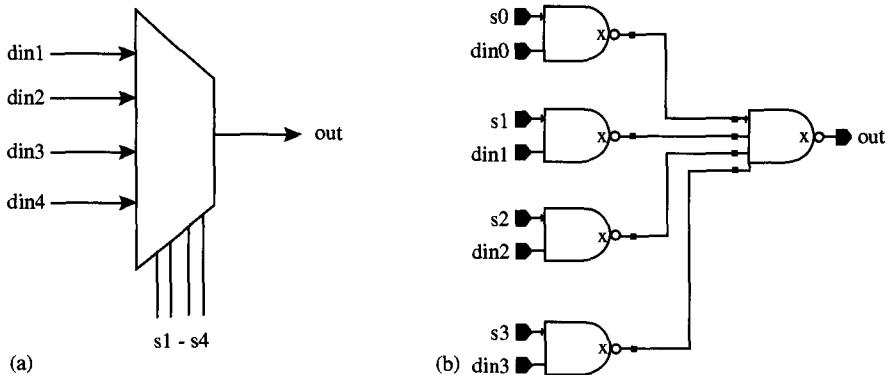
**Rule of thumb:** Dynamic logic is especially good at wide ORs/NORs in cycle-limiting logic paths, typically reducing delay by 50% over the static CMOS implementation.

**Rule of thumb:** Dynamic logic consumes 2 $\times$  the power of static logic, due to its phase activity, not including the power associated with clock generations and distribution.

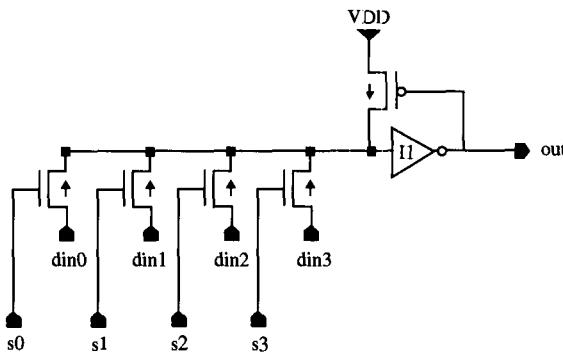
### 7.5.1.3 Pass-gate Examples

Pass-gate circuitry is made-to-order for the correct application. Because pass-gate performance suffers from *body effect* and *source follower action*, its use in the past had been limited. With the advent of deep submicron VLSI, however, scaling has motivated the use of substantially lower thresholds, resolving some of the impediments to pass-gate usage. To demonstrate the value of pass gates, let's consider another simple example.

In Fig. 7.13, the four-way multiplexer function is achieved in static two-way and four-way NANDs, occupying two intrinsic delay stages and 24 MOSFETs. In addition, there is a limit to how wide the mux can become before going to three tiers of logic. Each tier of logic must accommodate the load of the next stage. Consider now the pass-gate mux shown in Fig. 7.14, a circuit perfectly suited to the task at hand. The circuit requires seven transistors to accommodate the static equivalent, in approximately half the delay. Selected logical functions on high-speed processors were "made" for



**Figure 7.13** Combinatorial multiplexor in static CMOS: symbolic (a) and schematic (b) representations.



**Figure 7.14** Pass-gate multiplexor.

pass gates. Those functions include:

- **Multiplexing;** as shown in this example
- **Alignments;** byte aligners, rotators, barrel shifter operations
- **XOR;** random logic and parity generation
- **Arithmetic operations;** ALUs, multipliers, divide-bys

Pass-gate logic is a prudent choice in specifically crafted and controlled settings, and is powerful in performance-critical paths.

**Rule of thumb:** Pass-gate logic may consume half the power of static logic in certain scenarios. Attention must be paid to completing transitions that otherwise end at one threshold voltage drop below  $V_{DD}$ .

**Rule of thumb:** Pass-gate-based logic is not appropriate when long interconnects separate logic stages, or when logic circuits have high fan-out load.

#### 7.5.1.4 DCVS

Although useful in a number of applications, DCVS has been shown to be especially effective in executing *error correction code* (ECC). The reader is referred to the cited reference for an example of Hamming Error Coding accomplished with DCVS structures in an actual product application [16].

### 7.5.2 Design Metrics

A number of different figures of merit can be used to characterize the qualities of a given circuit design style. Four measurements are described below.

- Delay.** In the design of high-performance microprocessors, performance, of course, is the first parameter of interest when comparing styles. One must not overlook the trade-offs each of the styles make to achieve low circuit delays.
- Power.** Power consumption is a concern in all market segments. In portable applications, battery life is the boundary condition. In the high end, excessive power consumption leads to expensive heat dissipation techniques and thermal “run-away.”
- Energy-delay product.** For power concerns, energy consumed in a logic circuit is a superior measure of the “cost” of a function, because it is independent of frequency. The power dependence on frequency, which is true of most any circuit topology, is thus avoided. Energy alone does not reflect the performance impact of reducing operating voltage, however. *Energy-delay product* (EDP) is a much more meaningful number [17]. Voltage and device size have limited effects on EDP; technology scaling, however, has a profound influence. Figure 7.15 is a plot of energy, delay, and their product against supply voltage, in executing a given logical operation.
- Logical effort/branching effort/electrical effort.** The *Logical Effort Metrics*, advanced by Sutherland and Sproull [18], offers a straightforward means of determining the fastest CMOS implementation of an arbitrary logic function. The concept may be summarized by reviewing its components.
  - *Logical effort (“G”)* is a measure of how much harder than an inverter a circuit must work to produce output current, given equivalent input loading.
  - *Electrical effort (“H”)* is the output capacitance divided by the input capacitance for a given input to a CMOS stage. It is a measure of the step-up in capacitance achieved at the stage.
  - *Branching effort (“B”)* is the total output load of a given stage divided by the load associated only with the active path under evaluation. It is a measure of the load overhead necessary to drive a given net.
  - *Path effort (“F”)* is defined as the product of the three:

$$F = GBH$$

- *Parasitic delay (“P”)* is that component of a circuit’s delay independent of load.

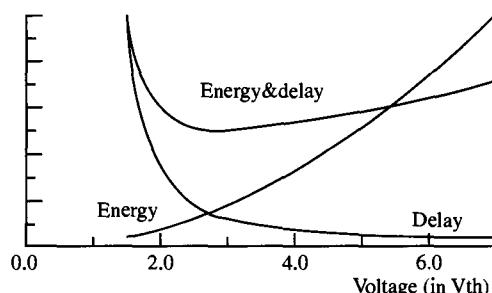


Figure 7.15 Energy, delay, and their product versus operating voltage.

The delay incurred by invoking path effort  $F$  is called  $D_F$ .  $D_F$  added to a path's total parasitic delay  $P$  forms the total delay of the path.

$$D = D_F + P$$

Path delay is minimized when each stage supports the same value of  $F$ , that is, when each stage effort  $f$  in  $N$  stages of a path is the same:

$$f = F^{1/N}$$

Over the whole path of  $N$  stages, then, the path may be optimized by computing transistor sizes that minimize total delay, described by

$$D = N(GBH)^{1/N} + P$$

This simple but elegant approach to optimizing logical paths is an important tool, and the reader is directed to the cited literature for a complete treatment of this method. The alpha ratio rule-of-thumb cited on p. 122 falls directly out of this optimization.

5. **Style comparisons.** Superb assessments of the relative merits of the different topologies exist in the literature and demonstrate the trade-offs in design space [19].

## 7.6 CONCLUSION

A survey of various means of executing logic offers a glimpse of the multiple solutions available to the designer. Nonclocked implementations, such as static combinatorial CMOS, pass gate, and differential cascode voltage switch each produce a design point that weighs differently the trade-offs between performance, area, noise immunity, wear-out, and reliability. Clocked structures such as single-ended and differential dominos adapt more aggressive conventions to improve performance. Self-timed or self-resetting designs, while much more difficult to configure, enable the introduction of asynchronous high-speed logic to more and more of the total state machine.

An optimized chip design point includes a mix of circuit design topologies, each specifically applied to functions they work best in. It is incumbent upon the designer to recognize the settings appropriate to each family. Application of technology scaling implicitly requires anticipating the response of each of these styles in the migrated design. The pressures on scaling seem to favor structures that transfer charge rather than voltage levels.

## ACKNOWLEDGMENTS

The author acknowledges valuable inputs by I. Aller, I. Bergida, W. Bowhill, A. Chandrakasan, E. Nowak, N. Rohrer; and the support of his management team at IBM Microelectronics, K. S. Ginn, S. Lewin, and H. Levine.

## REFERENCES

- [1] M. Shoji, *High Speed Digital Circuits*. Addison-Wesley, Reading, MA, 1996.
- [2] M. Shoji, *CMOS Digital Circuit Technology*. Prentice Hall, Englewood Cliffs, NJ, 1988.

- [3] H.J.M. Veendrick, *MOS ICs: From Basics of ASICs*. Wiley VCH, Weinheim, Germany, 1992.
- [4] C. Mead and L. Conway, *Introduction to VLSI Systems*. Addison-Wesley, Reading, MA, 1979.
- [5] L. G. Heller, et al., “Cascode Voltage Switch Logic: A Differential CMOS Logic Family,” Proceedings of 1984 IEEE International Solid-State Circuits Conference, San Francisco, pp. 16–17.
- [6] K. Bernstein, et al., *High Speed CMOS Design Styles*. Kluwer, Boston, 1998.
- [7] K. Yano, et al., “A 3.8 ns CMOS  $16 \times 16$  Multiplier Using Complementary Pass Transistor Logic,” *IEEE Journal Solid-State Circuits*, vol. 25, no. 2, April 1990, pp. 388–395.
- [8] K. Yano, et al., “Lean Integration, Achieving a Quantum Leap in Performance and Cost of Logic LSIs,” Proceedings of IEEE 1994 Custom Integrated Circuits Conference, San Diego, CA, pp. 603–606.
- [9] R. H. Krambeck, et al., “High Speed Compact Circuits with CMOS,” *IEEE Journal Solid-State Circuits*, vol. SC-17, no. 3, June 1982, pp. 614–619.
- [10] P. Gronowski, “A 433 MHz 64 bit Quad-Issue RISC Microprocessor,” Massachusetts Institute of Technology MTL VSLI Seminar, Cambridge, MA, March 12, 1996.
- [11] T. A. Grotjohn, et al., “Sample-Set Differential Logic (SSDL) for Complex High-Speed VLSI,” *IEEE Journal Solid-State Circuits*, vol. SC-21, no. 2, April 1986, pp. 367–369.
- [12] T. Wuu, et al., “A Design of a Fast and Area Efficient Multi-Input Muller C Element,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 2, June 1993, pp. 215–219.
- [13] A. K. Woo, et al., “Static PLA or ROM Circuit with Self-generated Precharge,” *U.S. Patent #4728827*, December 3, 1986.
- [14] G. Yee, et al., “Clock-Delayed Domino for Adder and Combinational Logic Design,” Proceedings of the 1996 International Conference on Computer Design, Austin, TX, pp. 332–337.
- [15] J. Silberman, et al., “A 1.0 GHz Single-Issue 64 b PowerPC Integer Processor,” Proceedings of the 1998 IEEE International Solid-State Circuits Conference, San Francisco, pp. 230–231.
- [16] J. Fifield, et al., “High Speed On-chip ECC for Synergistic Fault Tolerant Memory Chips,” *IEEE Journal Solid-State Circuits*, vol. 26, no. 10, October 1991, pp. 1449–1452.
- [17] M. Horowitz, et al., “Low-Power Digital Design,” Proceedings of the 1994 IEEE Symposium on Low Power Electronics, San Diego, CA, pp. 8–11.
- [18] I. E. Sutherland and R. F. Sproull, “Logical Effort: Designing for Speed on the Back of an Envelope,” *Advanced Research in VLSI 1991*: UC Santa Cruz, pp. 1–16.
- [19] U. Ko, et al., “Low Power Design Techniques for High Performance CMOS Adders,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 2, June 1995, pp. 327–333.

Paul Gronowski  
*Compaq Computer Corporation*

## 8.1 INTRODUCTION TO DYNAMIC LOGIC

Dynamic logic is a circuit style that is well suited for high-performance microprocessor designs. It offers a significant performance advantage over static circuits with reduced area. Unfortunately, dynamic logic is more susceptible to noise. While noise in static circuits can lead to reduced performance, noise in dynamic circuits can lead to functional failures. Strict design methodology and circuit guidelines can reduce some of the risk associated with using dynamic logic, but significant analysis and verification are still required to ensure proper functionality. Analysis and verification take time and resources, which could have a negative impact on time to market. Therefore, dynamic logic should only be used when the performance benefit is necessary or when it simplifies the implementation.

The basic concepts of dynamic logic and various dynamic circuit styles are discussed in more detail below.

### 8.1.1 Basics of Dynamic Logic

In static logic, the output node is always actively driven typically either to  $V_{DD}$  or to  $V_{SS}$ . In dynamic logic the output node can either be driven or left floating. When the output node is left floating, the value is stored on the parasitic capacitance of the output node. Figure 8.1 illustrates some simple dynamic structures.

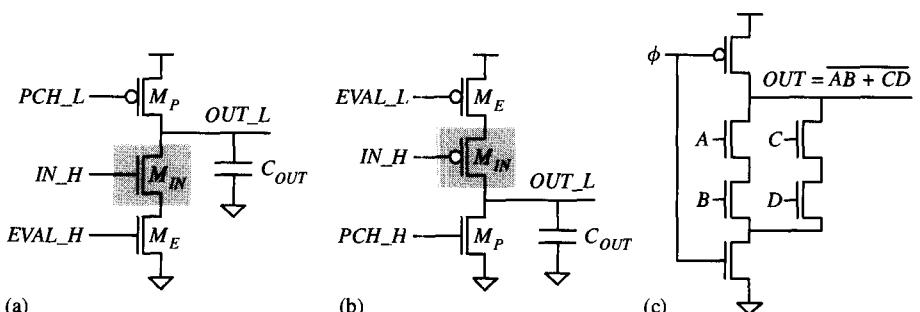


Figure 8.1 Simple dynamic gates: (a) NMOS inverter; (b) PMOS inverter; (c) multiple-input NMOS gate.

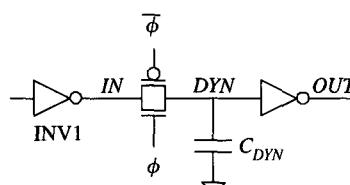
The circuit shown in Fig. 8.1(a) is a NMOS dynamic inverter. It utilizes a pre-charge-evaluate configuration. To function properly, the precharge device  $M_P$  and the evaluate device  $M_E$  cannot be conducting at the same time. Therefore, the signals  $PCH\_L$  and  $EVAL\_H$  cannot be asserted at the same time. Typically, this is achieved by connecting  $M_P$  and  $M_E$  to the same clock signal. If both devices are on simultaneously and the input  $IN\_H$  is high, short-circuit current flows between  $V_{DD}$  and  $V_{SS}$  resulting in unnecessary power dissipation, reduced performance, and possibly improper evaluation of the circuit.

Dynamic logic has two distinct phases of operation, the precharge phase and the evaluation phase. During the precharge phase when  $PCH\_L = 0$  and  $EVAL\_H = 0$ ,  $M_P$  is on and the output node  $OUT\_L$  is charged to  $V_{DD}$ . Since  $M_E$  is off, the state of  $IN\_H$  cannot effect the state of  $OUT\_L$ . During the evaluation phase when  $EVAL\_H = 1$  and  $PCH\_L = 1$ ,  $M_E$  is on and  $M_P$  is off. The output node will remain at  $V_{DD}$  if  $IN\_H = 0$  and will be discharged to  $V_{SS}$  if  $IN\_H = 1$ .  $IN\_H$  must be stable throughout the entire evaluation phase, or monotonically transition from  $0 \rightarrow 1$  during the evaluation phase.

The dynamic inverter can also be built using PMOS logic as shown in Fig. 8.1(b). The operation is similar to that of the NMOS dynamic inverter. The output node  $OUT\_L$  is predischarged to  $V_{SS}$  by the NMOS device  $M_P$  when  $PCH\_H = 1$ . During the evaluation phase when  $EVAL\_L = 0$ ,  $OUT\_L$  is conditionally pulled high based on the value of the input signal. In a two-phase clock implementation, the dynamic PMOS inverter evaluates in the opposite clock phase as the dynamic NMOS inverter. The input  $IN\_H$  must be valid throughout the entire evaluation phase, or monotonically transition from  $1 \rightarrow 0$ . Since NMOS devices conduct at least twice the current of PMOS devices for a given transistor size, the NMOS structure is preferred to the PMOS structure.

The logical complexity of a dynamic gate can be increased by replacing the highlighted input transistor  $M_{IN}$  in Fig. 8.1(a) or (b) with an NMOS or PMOS network. A four-input NMOS dynamic gate example is shown in Fig. 8.1(c). In addition, the precharge and evaluate devices are both connected to the clock signal  $\phi$ . As with all complementary CMOS gates, dynamic precharge-discharge gates always create inverted output signals.

A second type of dynamic structure, a dynamic transmission gate latch, is shown in Fig. 8.2. When  $\phi = 1$ , the latch is open and the value on  $IN$  is actively driven onto  $DYN$  by the inverter (INV1) through the transmission gate. When  $\phi = 0$ , the latch is closed and  $DYN$  is left floating, relying on charge stored on the node capacitance  $C_{DYN}$  to retain its value. Changes in the state of  $IN$  do not effect the value on  $DYN$  while the transmission gate is closed.



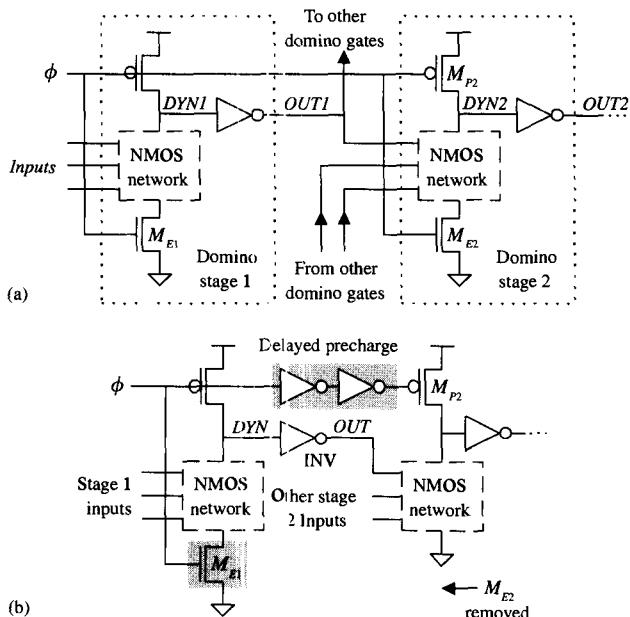
**Figure 8.2** Simple dynamic transmission gate latch.

## 8.1.2 Examples of Dynamic Logic

### 8.1.2.1 Domino Logic

One of the most commonly used dynamic structures is the domino logic gate [1]. Each domino gate is composed of a dynamic gate with an NMOS pull-down or PMOS pull-up network followed by an inverter. Domino gates are cascaded such that the inputs to each stage are driven directly by the outputs of other domino stages. The addition of the inverter guarantees that the inputs to each stage are all deasserted by the start of the evaluation phase. One drawback of domino logic is that it can only perform noninverting functions. Figure 8.3(a) illustrates a two-stage NMOS domino structure. During the precharge phase ( $\phi = 0$ ),  $DYN1$  and  $DYN2$  are precharged to  $V_{DD}$  and the output nodes  $OUT1$  and  $OUT2$  are driven to  $V_{SS}$ . All of the other inputs to stage 2 are driven to  $V_{SS}$  as well, since they are outputs from other domino stages. When  $\phi$  transitions from  $0 \rightarrow 1$ ,  $M_{E1}$  turns on and the dynamic gate in stage 1 starts to evaluate.  $M_{E2}$  turns on as well, but stage 2 cannot immediately evaluate because its inputs are still at 0. Evaluation of stage 2 is delayed until the domino gates driving its inputs have evaluated.  $DYN1$  conditionally discharges depending on the value of the inputs to stage 1 and the configuration of the NMOS pull-down network. If  $DYN1$  discharges,  $OUT1$  will be driven to  $V_{DD}$  enabling evaluation of stage 2.

Since the evaluation of stage 2 cannot proceed until evaluation of its input is complete, the evaluation device is redundant and can be removed [Fig. 8.3(b)]. Removing  $M_{E2}$  reduces the stack height and improves the speed of the circuit, but it also creates a power issue during the precharge phase that must be addressed. With the evaluation device removed, the pull-down path is not disabled until the inputs to stage 2 are all



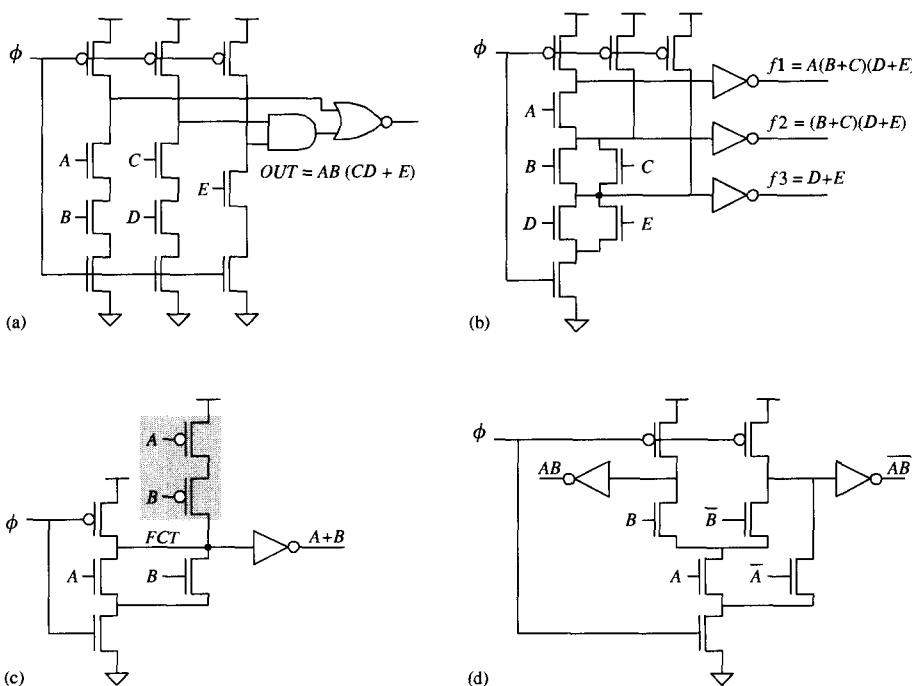
**Figure 8.3** Domino logic: (a) NMOS structure; (b) clocked evaluation device removed and delayed precharge added in stage 2.

driven to 0. This requires that stage 1 precharge has completed and propagated through the inverter driving 0 to the input of stage 2. During this time, the precharge device in stage 2 is on and the pull-down path may still be active. To reduce cross-over current, the precharge of the second domino stage is delayed using two inverters as shown in Fig. 8.3(b). The propagation delay through the highlighted inverters in the precharge path should closely match the delay of the two gates in the previous domino stage, the dynamic gate and the inverter, to reduce power without impacting performance. The addition of delay elements in the precharge path causes the precharge to ripple.

Figure 8.4 illustrates some variations of the standard domino gate. In compound domino logic [Fig. 8.4(a)] the output inverter found in standard domino logic is replaced by a multiple-input complementary gate [2], [3]. Typically, this leads to reduced stack heights and enables increased logic complexity. This domino style can reduce charge sharing by moving some of the logic out of the pull-down network and into the output gate. Charge sharing is discussed in more detail later in this chapter.

Multiple-output domino logic (MODL) can reduce transistor count and device area in instances where intermediate functions in an evaluation tree are also required as outputs [4]. A single MODL structure can be used instead of repeating logic in several domino structures. The example circuit in Fig. 8.4(b) uses 15 transistors to generate three outputs compared to the 23 transistors required to generate the same outputs using standard domino gates.

Noise-tolerant precharge (NTP) domino logic [Fig. 8.4(c)] can be used when improved noise immunity is needed [5]. The complementary PMOS network that is



**Figure 8.4** Other forms of domino logic: (a) compound domino logic; (b) multiple-output domino logic (MODL); (c) noise-tolerant precharge (NTP) domino logic; (d) dual-rail domino logic.

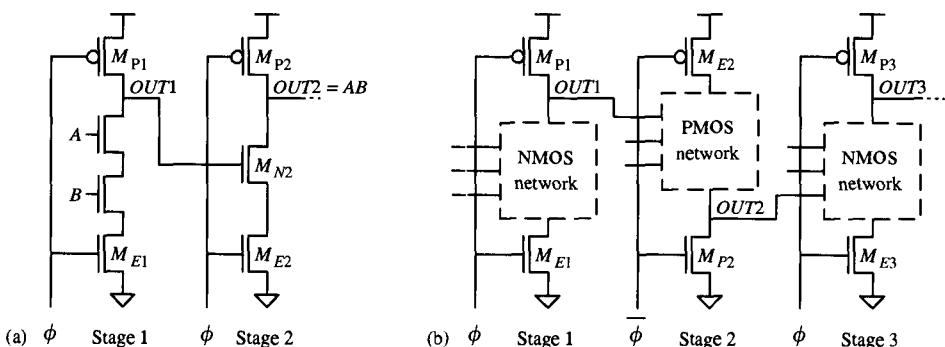
normally removed from dynamic logic is reintroduced. The PMOS network is not intended to replace the precharge device. It simply supplies enough current to the internal node in instances where it is left floating during evaluation to protect it from noise. In this example, this occurs when both inputs  $A$  and  $B$  are 0. The cost of improved noise immunity is increased area, increased power consumption, and reduced speed. Other methods of noise protections are discussed later.

Dual-rail dynamic logic has a higher device count and consumes more power than standard domino logic [6]. Since true and complementary data is generated, the power is not dependent on the input data. One of the outputs is guaranteed to switch every cycle. Dual-rail is typically used when both true and complementary outputs are required [Fig. 8.4(d)], but it requires true and complementary inputs. Standard domino logic is capable of generating noninverting outputs only.

### 8.1.2.2 Cascading Dynamic Structures

There is a problem when same polarity dynamic gates are cascaded without the intervening inverter found in domino logic. Consider the dynamic NMOS example shown in Fig. 8.5(a). As previously mentioned, the input signals to a dynamic NMOS gate must either be valid for the entire evaluation phase or monotonically transition from 0 → 1 during the evaluation phase. These input requirements are violated when dynamic NMOS gates are cascaded. When  $\phi = 0$ , transistors  $M_{P1}$  and  $M_{P2}$  are on and the output nodes  $OUT1$  and  $OUT2$  are precharged to  $V_{DD}$ . The problem occurs when the inputs to stage 1,  $A$  and  $B$  are both 1 and evaluation of stage 1 will result in transition on  $OUT1$  from 1 → 0. At the beginning of the evaluation phase when  $\phi$  transitions from 0 → 1,  $OUT1$  is still high, transistors  $M_{N2}$  and  $M_{E2}$  are on and  $OUT2$  incorrectly starts to discharge.  $OUT2$  continues to discharge until  $OUT1$  is pulled below the threshold voltage of  $M_{N2}$ . Since  $OUT2$  is a dynamic node and not actively driven to  $V_{DD}$  during the evaluation phase, the lost charge cannot be restored.

The problem with cascading dynamic gates is solved by alternating the polarity of each stage as seen in Fig. 8.5(b). This circuit style is commonly referred to as NORA (or NO RAce) logic since the race described in the previous example is eliminated [7], [8]. Each stage alternates between NMOS and PMOS dynamic gates. When  $\phi = 0$ ,  $OUT1$  and  $OUT3$  are charged to  $V_{DD}$  and  $OUT2$  is discharged to  $V_{SS}$ . Therefore, the input



**Figure 8.5** Cascading dynamic gates: (a) same polarity; (b) alternating polarity (NORA logic).

requirements for all of the dynamic stages are met. The inputs to NMOS stage 3 all start at 0 and may transition to 1 during evaluation. Likewise, the inputs to PMOS stage 2 all start at 1 and may transition to 0. The evaluation of each stage is now dependent on evaluation of the previous stage. One drawback of NORA logic is that it is highly susceptible to noise, more so than complementary logic and dynamic domino logic. The output of each dynamic stage is driven directly into an NMOS or PMOS transistor of the following stage. Therefore, any noise on an output is immediately propagated to the input of the next stage. The output need only change by a threshold voltage to turn on the input device of the next stage corrupting the data. While NORA provides less noise margin, it significantly improves the speed of the circuit.

### 8.1.3 Comparison of Dynamic Logic to Standard Complementary CMOS Logic

Some of the general design features of dynamic logic are listed below.

#### 8.1.3.1 General Design Features of Dynamic Logic

- The use of a clock in dynamic logic allows for synchronization and latching.
- In a two-phase design, dynamic logic outputs are valid for less than 50% of the cycle time. One complete phase is used for precharge, and the other phase is used for evaluation.
- Robust dynamic logic is more difficult to synthesize than complementary logic.
- Dynamic logic allows for very wide structures that cannot possibly be built in one static gate such as a 64 bit zero detect.
- NMOS dynamic logic is more commonly used than PMOS dynamic logic.

The area, speed, power consumption, and noise immunity of dynamic and complementary static logic are compared below.

- **Area:** Dynamic logic typically takes less area than the equivalent static circuit.
  - An N-input dynamic gate requires  $N + 2$  transistors, one for the precharge, one for the evaluation, and one for each of the N inputs (neglecting keepers and intermediate precharge devices which will be discussed later). An N-input complementary gate requires  $2 * N$  transistors, one PMOS and one NMOS for each input.
  - In NMOS dynamic logic, the PMOS network that exists in a complementary static gate is replaced by a single, typically smaller PMOS precharge device.
- **Speed:** Dynamic logic is typically faster than the equivalent static circuit.
  - Dynamic logic has lower input capacitance. The input gates only need to drive a PMOS or NMOS device and not both.
  - Dynamic logic has lower source/drain junction capacitance.
  - Dynamic logic switches with an input voltage change of  $V_T$ , while static logic is typically sized to switch with a change of  $V_{DD}/2$  on the input to equalize rise and fall times.
  - When dynamic logic evaluates to the precharge value, the output is available immediately at the clock edge.

- **Power:** Power dissipation for dynamic logic is generally greater than that for static logic, but each case must be examined carefully.
  - Dynamic logic has smaller input capacitance than static logic since the NMOS or PMOS network is eliminated (reduced power).
  - Dynamic logic has smaller source/drain junction capacitance (reduced power).
  - Dynamic logic increases gate capacitance on the clock node by adding clocked precharge and evaluation devices.
  - Noise on dynamic nodes can lead to voltage degradation creating short-circuit current on subsequent gates.
  - Dynamic logic typically has a high switching factor even when the inputs signals do not change from cycle to cycle.
  - Static logic has short-circuit current when both the PMOS and NMOS devices are driving the output during transitions in the input voltage. Dynamic logic can have short-circuit current when clock transitions.
  - Static logic can have glitches as input signals change at various times, resulting in increased output transitions and wasted power.
- **Noise immunity:** Static circuits always offers better noise immunity than dynamic circuits.
  - Reduced noise immunity is the biggest drawback with dynamic logic when compared to static circuits.
  - Noise on static logic typically results in speed degradation, whereas noise on dynamic logic can lead to functional failures as well.

## 8.2 DESIGN ISSUES WITH DYNAMIC LOGIC

As previously mentioned, dynamic nodes rely on storing charge on the parasitic node capacitance to retain state. Since dynamic nodes are not always actively driven, noise events can affect the amount of charge stored resulting in logic failures. Some of these noise sources are discussed below [9].

### 8.2.1 Charge Leakage

Dynamic nodes are susceptible to charge leakage through both NMOS and PMOS devices when the node is not actively driven. Consider the dynamic inverter in Fig. 8.6(a).

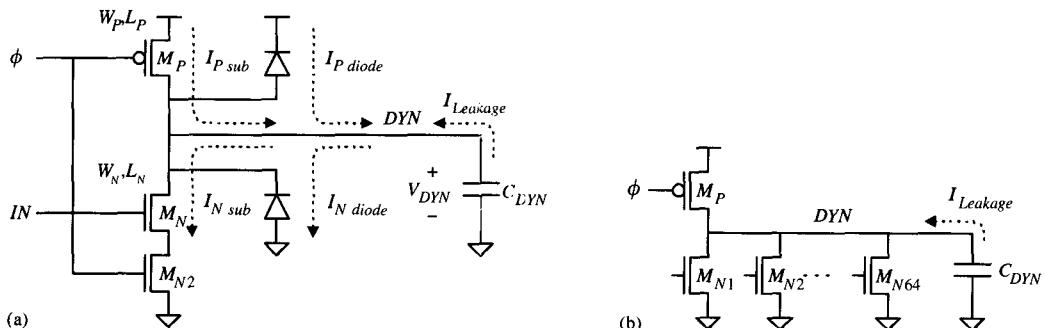


Figure 8.6 Leakage on dynamic nodes: (a) sources of leakage; (b) wide OR structure.

Assume that the dynamic node  $DYN$  has been precharged to  $V_{DD}$  and that  $IN$  is 0 when  $\phi$  goes high. Both  $M_P$  and  $M_N$  are “off,” and the output node is left floating. The ability of the output node to retain its stored charge is dependent on the magnitude of the leakage currents through the two devices. There are two separate leakage sources to consider for each device with a source/drain connection to the dynamic node.

A small amount of current flows through a device even when it is considered to be “off” (when  $V_{GS} < V_T$ ). This is commonly referred to as the subthreshold leakage current. In Fig. 8.6(a), the subthreshold leakage current through  $M_P (I_{P\ sub})$  adds charge to  $DYN$  and the subthreshold leakage current through  $M_N (I_{N\ sub})$  removes charge from  $DYN$ . The effect is exacerbated by the presence of noise on the input node when  $V_{GS}$  approaches  $V_T$ . In many NMOS dynamic circuits, the cumulative channel width ( $\sum W_N$ ) of the NMOS devices connected to the dynamic node is greater than the width ( $W_P$ ) of the PMOS precharge device. This is especially the case in structures with wide fan-in like the 64-bit dynamic NOR gate shown in Fig. 8.6(b). As a result, the net effect of the combined subthreshold leakage currents is typically removal of charge from the output node. The magnitude of the subthreshold leakage is highly dependent on the device characteristics, which can vary between NMOS and PMOS devices in the same process, as well as operating conditions.

The second source of leakage is the reverse-biased diode current. Parasitic p-n diodes are formed by the n+ drain diffusion of the NMOS device and the p-substrate, and by the p+ drain diffusion of the PMOS device and the n-well. The reverse-biased currents are given as  $I_{N\ diode}$  and  $I_{P\ diode}$ , for the NMOS and PMOS, respectively. Therefore, the total leakage current on the dynamic node in this example is

$$I_{Leakage} = (I_{N\ sub} + I_{N\ diode}) - (I_{P\ sub} + I_{P\ diode}) \quad (8.1)$$

If  $I_{Leakage}$  is a positive value, the node capacitance represented by  $C_{DYN}$  will be discharged to  $V_{SS}$  over time. After some period of time  $t_{CRIT}$ , the output voltage  $V_{DYN}$  will drop far enough below  $V_{DD}$  to cause a logic failure. The length of time it takes for the output node to drop by the critical voltage  $V_{CRIT}$  is

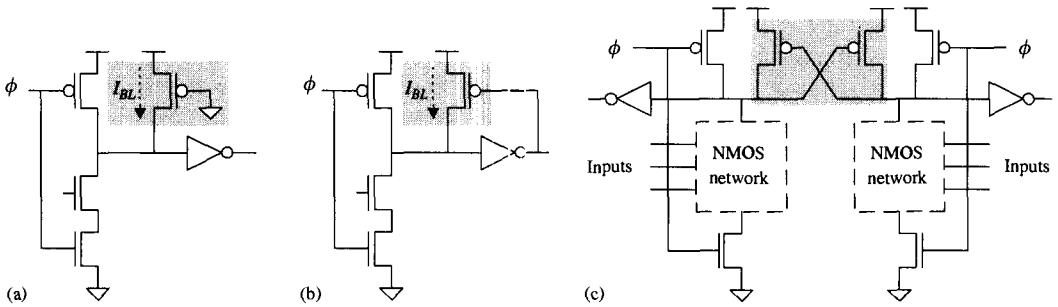
$$t_{CRIT} = (C_{DYN} * V_{CRIT}) / I_{Leakage} \quad (8.2)$$

The leakage current can limit the minimum operating frequency to  $f_{MIN}$ . A high  $f_{MIN}$  can impact the ability to test and debug a part especially during wafer probe.

$$f_{MIN} = 1 / (t_{CRIT} * \text{the number of phases per cycle}) \quad (8.3)$$

One side effect of technology scaling is that the transistor subthreshold leakage ( $I_{doff}$ ) increases. Subthreshold leakage is higher in short-channel devices where  $V_T$  is strongly influenced by gate dimensions and applied drain bias. A small increase in the channel lengths of the transistors in a dynamic circuit will reduce the leakage current and lower  $f_{MIN}$  with negligible impact on speed. In technologies with multiple threshold devices, low  $V_T$  transistors typically have higher leakage than normal  $V_T$  transistors. Low  $V_T$  devices are typically too “leaky” to be used in dynamic logic.

Leakage analysis should take into account worst-case operating condition and process parameters where leakage is maximized such as high temperature, fast process “corner” device characteristics, minimum device and interconnect capacitance, and so on. In early CMOS technologies, the reverse-biased diode current was the dominant leakage mechanism. As devices are pushed to achieve the higher performance and threshold voltages are reduced, the subthreshold leakage source is beginning to dominate.



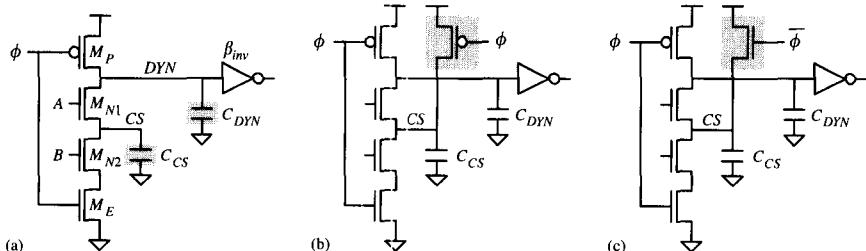
**Figure 8.7** Solutions for leakage (and charge sharing): (a) bleeder device; (b) active feedback device; (c) cross-coupled devices on dual-rail dynamic logic.

Dynamic transmission gate latches such as the one introduced in Fig. 8.2 can also exhibit leakage behavior. When  $\phi$  is low and the transmission gate is closed,  $DYN$  is left floating and the data is “latched.” The charge stored on the dynamic node can “leak” through the transmission gate if the polarities of  $DYN$  and  $IN$  are opposite when the latch is closed.  $DYN$  can leak in either directions,  $0 \rightarrow 1$  or  $1 \rightarrow 0$ .

Figure 8.7 illustrates three circuit modifications that can be made to counteract the leakage problem. In Fig. 8.7(a), a PMOS device with its gate tied to  $V_{SS}$  is added to the circuit. The current  $I_{BL}$  through the PMOS device, often called a “bleeder,” must be greater than the leakage current  $I_{Leakage}$ . The impact of process variation on transistor drive characteristics must be considered when sizing the “bleeder.” The “bleeder” in Fig. 8.7(a) is always on, supplying current even when the dynamic node is to be driven low, resulting in static power dissipation. Connecting the gate of the bleeder as shown in Fig. 8.7(b) eliminates that problem. When the dynamic node is high, the output of the inverter is low and the bleeder is “on” holding the dynamic node high. When the dynamic node is discharged, the inverter output is high and the bleeder is turned “off.” Some static power is dissipated when the dynamic node is being discharged and the output of the inverter is still high. Cross-coupled PMOS devices can be used on dual-rail dynamic logic as shown in Fig. 8.7(c). All three solutions have a negative impact on power, area, and speed, but they eliminate the minimum operating frequency constraint by making the dynamic circuit pseudo-static.

### 8.2.2 Charge Sharing

Charge sharing is another issue that can lead to failures on dynamic nodes. Consider the example shown in Fig. 8.8(a). The dynamic node  $DYN$  is driven to  $V_{DD}$  during the



**Figure 8.8** Charge sharing: (a) circuit diagram; (b) precharge internal node with PMOS; (c) precharge internal node with NMOS.

precharge phase when  $\phi = 0$ . Assume that both inputs  $A$  and  $B$  are low during precharge, and the internal stack node  $CS$  was discharged to  $V_{SS}$  during the previous evaluation phase. When  $\phi$  transitions from  $0 \rightarrow 1$  the precharge device  $M_P$  is turned off and  $DYN$  is left floating. Since  $A$  and  $B$  are both low,  $CS$  is also floating. The initial charge stored on  $DYN$  is  $(V_{DD} * C_{DYN})$  and the initial charge stored on  $CS$  is 0. Now assume that  $A$  transitions from  $0 \rightarrow 1$  during the evaluation phase, turning on  $M_{N1}$ . The charge stored on  $DYN$  is “shared” between  $C_{DYN}$  and  $C_{CS}$  resulting in a voltage loss on  $V_{DYN}$ . The final voltage on  $V_{DYN}$  after charge sharing is given by

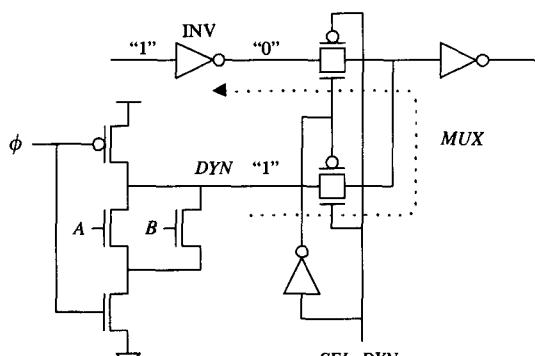
$$V_{DYN} = V_{DD} \left( \frac{C_{DYN}}{C_{DYN} + C_{CS}} \right) \quad (8.4)$$

Equation (8.4) is valid only when the change in voltage on  $DYN$  due to the charge sharing is greater than the threshold voltage of the NMOS transistor  $M_{N1}$ . The maximum voltage  $CS$  can charge to is  $(V_{DD} - V_{Tn})$ , so when the voltage drop on  $DYN$  is less the threshold voltage, Eq. (8.5) must be used instead.

$$V_{DYN} = V_{DD} - \frac{C_{CS}}{C_{DYN}} (V_{DD} - V_{Tn}) \quad (8.5)$$

Since  $DYN$  is a dynamic node, the voltage drop cannot be recovered. If the voltage drops far enough, the output of the inverter may incorrectly transition from  $0 \rightarrow 1$  causing a failure. Increasing  $\beta_{INV}(\beta_N/\beta_P)$  can improve the noise margin of the inverter, but does not guarantee that the circuit will operate properly. Examining the timing of the input signals and reordering the position of the inputs in the pull-down stack can sometimes be used to reduce charge sharing on specific nodes.

The two solutions presented in Figs. 8.7(a) and (b) to combat leakage can be used to restore the lost charge on  $DYN$ , and may provide acceptable results with charge-sharing problems. In large dynamic structures where multiple internal nodes charge-share to the dynamic node, the leakage solutions presented earlier may not be acceptable. The most reliable solution is to precharge the internal stack nodes ( $CS$ ) and eliminate the charge-share problem. In Fig. 8.8(b), the highlighted PMOS device is used to precharge the internal node  $CS$  to  $V_{DD}$  when  $\phi$  is low. Since  $CS$  and  $DYN$  are both charged to  $V_{DD}$ , there is no voltage loss on  $DYN$ . Similarly, the NMOS device in Fig. 8.8(c) is used to precharge  $CS$  to  $V_{DD} - V_{Tn}$ . Both precharge options reduce the speed performance of the dynamic gate by adding additional capacitance and charge to the  $CS$  node which must be discharged when the  $DYN$  is meant to transition from



**Figure 8.9** Charge sharing to a dynamic node through a transmission gate.

$1 \rightarrow 0$ . In addition, the precharge device increases the gate capacitance on the  $\phi$  node resulting in increased power dissipation.

Dynamic nodes can also charge share through transmission gates as seen in Fig. 8.9. Assume that the dynamic node  $DYN$  is charged to  $V_{DD}$  and that inputs  $A$  and  $B$  remain low. When  $SEL\_DYN$  is low, the transmission gate output node  $MUX$  is driven to 0 by the static inverter and  $DYN$  is floating. When  $SEL\_DYN$  goes high, the nodes  $MUX$  and  $DYN$  share charge, lowering the voltage on  $DYN$ . Additional charge is lost through the inverter INV during the transition time when  $MUX$  is being driven by both sources. To avoid this problem, refrain from connecting dynamic nodes directly to transmission gates.

### 8.2.3 Capacitive Coupling Issues

#### 8.2.3.1 Interconnect Capacitive Coupling

Dynamic nodes are susceptible to failures caused by interconnect capacitive coupling. When an unrelated signal wire running vertically or laterally adjacent to a dynamic node switches, the mutual capacitance between the two nodes can result in the addition or removal of charge from the dynamic node. The transition on the unrelated signal wire acts as an “aggressor,” forcing an unwanted voltage change on the dynamic “victim” node. If the magnitude of the mutual capacitance is significantly large compared to the total capacitance on the dynamic node, the state of the node may be lost, leading to a logic failure in the circuit.

Consider the circuit shown in Fig. 8.10. An unwanted parasitic coupling capacitance  $C_{DYN-CPL}$  exists between the dynamic node  $DYN$  and the unrelated signal wire  $DYN-CPL$ . During the precharge phase,  $DYN$  is charged to  $V_{DD}$ . If the inputs  $IN$  and  $IN2$  are kept low during the evaluation phase, transistors  $M_{IN}$  and  $M_{IN2}$  remain off and the voltage on  $DYN$  should remain at  $V_{DD}$ . However, if  $DYN-CPL$  undergoes a high-to-low-voltage transition, charge redistribution caused by the coupling capacitance will result in a voltage change on  $DYN$  of  $\Delta V_{DYN}$ .

$$\Delta V_{DYN} = (C_{DYN-CPL} * \Delta V_{DYN-CPL}) / C_{DYN} \quad (8.6)$$

$\Delta V_{DYN-CPL}$  is the magnitude of the voltage transition on  $DYN-CPL$ .  $C_{DYN}$  is the total capacitance on the dynamic node. It includes the gate, interconnect, and diffusion capacitance on the dynamic node as well as the coupling capacitance to  $DYN-CPL$ .

The magnitude of  $\Delta V_{DYN}$  that can be tolerated on the dynamic node before a logic failure occurs depends on the beta ratio  $\beta_{OUT}$  ( $\beta_{OUT} \equiv \beta_n/\beta_p$ ) and DC transfer characteristics of the output inverter. Increasing the beta ratio reduces the input-high voltage  $V_{IH}$  of the inverter [10].  $V_{IH}$  is defined as the smallest input voltage that can be

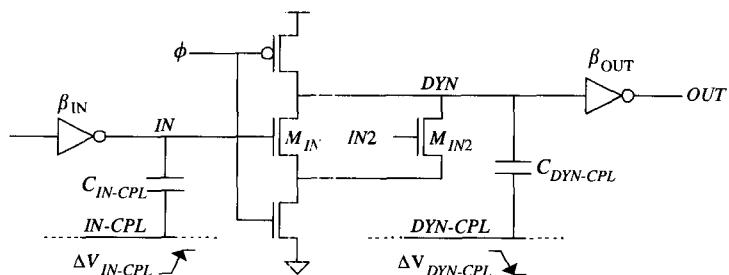


Figure 8.10 Capacitive coupling diagram.

interpreted by the output inverter as a high logic level, and is an important factor in determining the noise margin of the circuit. A lower value of  $V_{IH}$  results in a greater “high-voltage” noise margin  $NM_H$ . The negative effect of increasing the beta ratio is that the switching point of the inverter is lowered and therefore, the evaluation discharge time of the circuit is increased.

Typically, a dynamic node has more than one unrelated aggressor signal coupling to it, all of which must be accounted for in the analysis. Every node that is capacitively coupled to the dynamic node can affect its voltage and contribute to causing a logic failure. Figure 8.11(a) shows a simplified circuit diagram of a single aggressor node coupling to a dynamic node. The edge rate and exact timing of the aggressor signal transition is not a consideration in the analysis as long as it occurs when the dynamic node is floating.

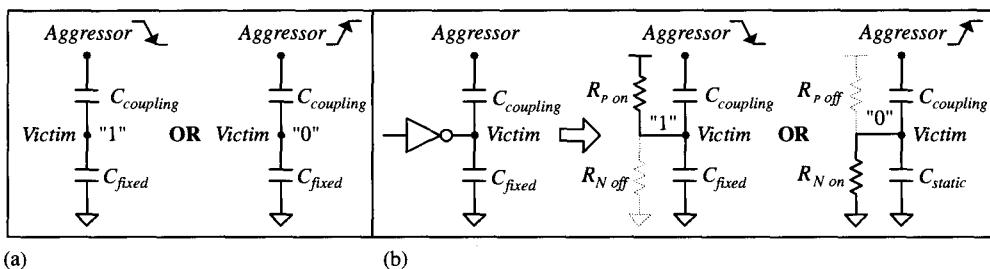
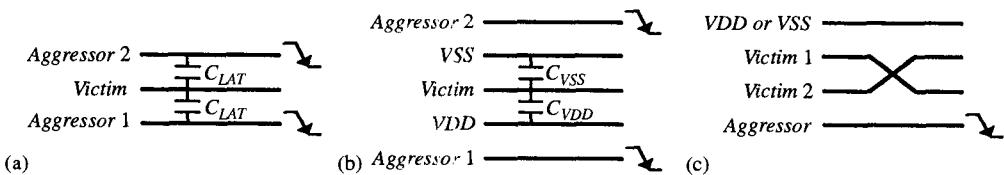


Figure 8.11 Simplified coupling circuit diagrams: (a) dynamic nodes; (b) static nodes.

Interconnect capacitive coupling to the inputs of dynamic nodes is also a concern. Consider again the circuit in Fig. 8.10. One of the inputs (*IN*) to the dynamic gate is driven by an inverter. An unrelated signal wire *IN-CPL* is coupled to *IN* through the parasitic capacitor  $C_{IN-CPL}$ . When *IN* is supposed to remain low, a low-to-high transition on *IN-CPL* can temporarily drive the voltage on *IN* above  $V_{SS}$ . For the purposes of analysis, the NMOS device of the inverter can be modeled as a resistor with a value equal to the “on” resistance of the device. The NMOS device will restore the voltage on *IN* to  $V_{SS}$ , but the voltage spike may cause a voltage drop on the dynamic node. The maximum voltage and width of the spike is influenced by the magnitude of the coupling capacitance  $C_{IN-CPL}$  as a percentage of the overall node capacitance, the edge rate of the coupling event, and the size of the inverter. Increasing the size of the NMOS device in the inverter will reduce the height and width of the spike and may be sufficient to fix the problem. Keeping the input inverter close to the dynamic gate will reduce the interconnect length and can reduce the probability of destructive coupling.

Capacitive interconnect coupling is a major concern in tightly packed datapath sections. Typically, datapath signal wires and buses are routed parallel to other signals for long distances at minimum pitch. By routing the signals at minimum pitch, the overall physical dimension of the datapath cell is reduced. However, this results in the maximum lateral coupling capacitance  $C_{LAT}$  between the adjacent signal wires. The worst-case scenario for lateral coupling occurs when the dynamic *Victim* signal is routed at minimum pitch, between two *Aggressor* signals as shown in Fig. 8.12(a). The voltage on the dynamic node, initially charged to  $V_{DD}$ , can be reduced through the lateral capacitance by high- to low-voltage transitions on the *Aggressor* signals. Several options are offered to reduce the impact of the *Aggressor* signal coupling on the dynamic *Victim* signal. Increasing the space between the *Victim* and the *Aggressor* signals reduces the lateral

capacitance ( $C_{LAT\ NEW} < C_{LAT}$ ). Routing a dynamic *Victim* node between two complementary *Aggressor* signals reduces the overall coupling. The addition and removal of charge from the *Victim* caused by opposite transitions on the complementary signals effectively cancel, resulting in little or no voltage change on the *Victim* node. In Fig. 8.12(b), the *Victim* is fully shielded by power rails. Routing signals next to power rails instead of active signals does not reduce the overall capacitance on the *Victim* node ( $C_{VDD} = C_{VSS} = C_{LAT}$ ). It does, however, reduce the percentage of coupling capacitance between the dynamic node and actively transitioning *Aggressor* nodes. In Fig. 8.12(c), the position of two adjacent *Victim* lines are swapped to take advantage of routing next to the steady-state voltage of the power rails. All of these solutions add to the size of the datapath and may have a negative impact on the performance of the circuit.



**Figure 8.12** Interconnect capacitive coupling in datapaths: (a) worst-case lateral coupling; (b) shielding; (c) swapping routing tracks.

One alternative to increasing line spacing or adding shields is to encode signal wires to reduce the number of wires that can switch. The actual number of wires is not reduced. This technique is useful in instances where dual-rail buses are routed. Two sets of dual-rail signals can be encoded to reduce the maximum number of transitioning wires from four to two per cycle [11].

It is essential to analyze the capacitive coupling on all dynamic nodes to prevent unwanted logic failures. A strict design methodology should be used to identify all nodes that exceed safe coupling limits. Limits should be set for the maximum tolerable coupling percentage on both dynamic nodes and the inputs to dynamic nodes. The beta ratio on gates that drive dynamic nodes should also be restricted to ensure acceptable noise margins. In general, reducing or eliminating the “bad” coupling capacitance on dynamic nodes reduces the probability of a circuit failure.

### 8.2.3.2 Miller and Back-gate Coupling

Gate coupling can affect the voltage on dynamic nodes. Consider the circuit shown in Fig. 8.13(a). Gate coupling from the precharge device  $M_{P1}$  can have a minor effect on the voltage on  $DYN$ . At the start of the precharge phase, when the voltage on  $\phi$  makes a transition from high to low, the voltage on the dynamic node is coupled down through the gate to drain capacitance  $C_{gd}$ . This is not an issue, because the dynamic node will charge to  $V_{DD}$  during the precharge phase recovering the lost charge. At the end of the precharge phase, when the voltage on  $\phi$  makes a transition from low to high, the voltage on  $DYN$  is coupled above  $V_{DD}$  by  $C_{gd}$ . Since  $C_{gd}$  is typically small compared to the overall capacitance on the dynamic node, the rise in voltage will be small and have an insignificant impact on the speed of the circuit. This effect is commonly referred to as miller coupling capacitance.

Gate coupling between the output of a complementary static gate and a dynamic node input signal could have a significantly greater impact on the voltage of the

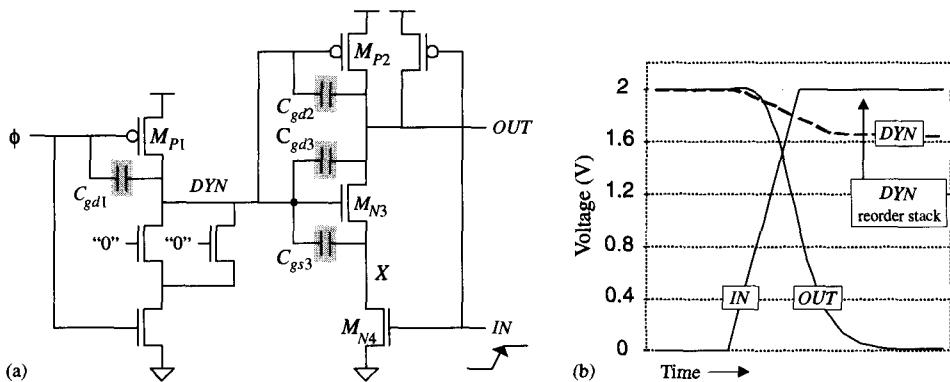


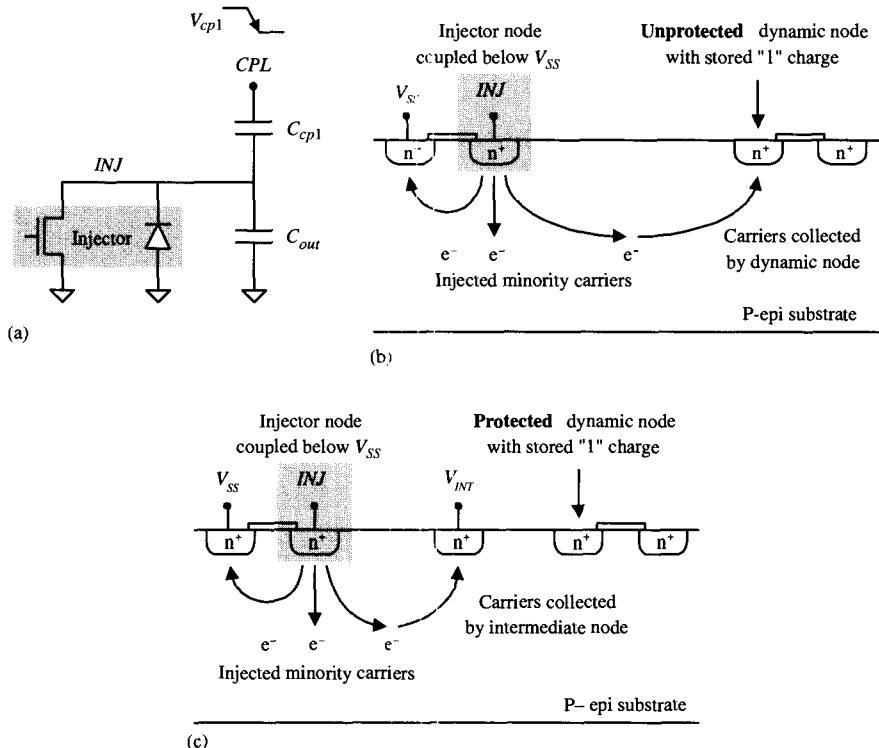
Figure 8.13 Back-gate coupling: (a) circuit diagram; (b) signal waveforms.

dynamic node than the miller coupling described above. Consider again the circuit shown in Fig. 8.13(a). A two-input NOR gate is driven by a dynamic input node *DYN* and a static input node *IN*. The dynamic node drives the NMOS transistor  $M_{N3}$  whose drain is connected to the output node *OUT*. During the precharge phase when  $\phi$  is low, *DYN* is charged to  $V_{DD}$ . If *IN* is low, *OUT* is also driven to  $V_{DD}$ , and the source of transistor  $M_{N3}$  (node *X*) is charged to  $V_{DD} - V_T'$ . Therefore, transistor  $M_{N3}$  has a  $V_{GS}$  equal to  $V_T'$  and is at the point of “turn-on” with a partially formed channel. If *IN* transitions from  $0 \rightarrow 1$ , nodes *OUT* and *X* are both discharged to  $V_{SS}$ . The  $V_{GS}$  on  $M_{N3}$  increases from  $V_T'$  to  $V_{DD}$  and the channel fully forms. If  $\phi$  is high and the precharge device is off, the increase in the gate capacitance of transistor  $M_{N3}$  results in a voltage drop on *DYN* as shown in Fig. 8.13(b). This effect is often referred to as back-gate coupling. The back-gate coupling can be eliminated by swapping the inputs to  $M_{N3}$  and  $M_{N4}$ . When the dynamic node drives  $M_{N4}$  instead of  $M_{N3}$ , the channel fully forms during the precharge phase allowing the precharge device time to charge the increased capacitance. The voltage drop on *DYN* is reduced with the reordered stack, and is solely due to the miller coupling capacitance and not the back-gate coupling.

### 8.2.4 Minority Carrier Charge Injection

Interconnect coupling events that drive nodes below  $V_{SS}$  and above  $V_{DD}$  may result in minority carrier injection into the substrate and well. A low-down coupling event on a signal node can drive the voltage of a n+ source/drain diffusion connected to that node below the p-substrate voltage. If the p-n junction formed by the substrate and n+ source/drain diffusion becomes sufficiently forward biased, minority carriers (electrons) will be injected into the substrate. These minority carriers move laterally in the substrate where they may be collected by the n+ source/drain of nearby dynamic nodes. Similarly, a high-up coupling event can drive the voltage of a p+ source/drain diffusion above the n-well voltage. Some of these injected minority carriers (holes) will flow vertically in the well, while others will flow laterally and may be collected by the p+ source/drain of a nearby dynamic node. If the charge collected by the dynamic node is significant compared to the stored charge, the state of the dynamic node may be corrupted.

Consider the circuit shown in Fig. 8.14(a). A transition on *CPL* from  $V_{DD}$  to  $V_{SS}$  forces *INJ* (initially at  $V_{SS}$ ) below  $V_{SS}$ , forward biasing the p-n junction and injecting



**Figure 8.14** Minority carrier charge injection: (a) circuit diagram; (b) cross section of unprotected dynamic node; (c) cross section of protected dynamic node.

electrons into the substrate. Figure 8.14(b) shows a cross-sectional view of the injecting node along with other n+ diffusions that can collect the injected charge. The surrounding diffusions include dynamic nodes, static nodes, and the source diffusion of the injecting NMOS device. The injecting diffusion and the substrate form the emitter and base of a parasitic NPN transistor, while the surrounding n+ diffusions form the collectors. Since the topology and distance between the n+ regions vary, the gain of each collector will vary as well. The “unprotected” dynamic node on the left in Fig. 8.14(b) is at risk of losing its stored “1” value. The reverse-biased p-n junction formed by the diffusion and substrate will attract the electrons flowing in the substrate. In addition, there are no intervening diffusions in the path between the injector and the dynamic node to reduce the number of carriers collected by the dynamic node.

The dynamic node in Fig. 8.14(c) is shielded from the injecting node by an intermediate diffusion. The “protected” dynamic node will still collect some of the injected charge, but the amount will be significantly reduced and the node should retain its stored “1” value.

Locating dynamic nodes that are vulnerable to data corruption by charge injection requires analysis of the physical layout of a design. The metal interconnect network must be extracted and analyzed to identify all nodes that can be coupled above  $V_{DD}$  or below  $V_{SS}$ . After these nodes are identified, the amount of charge injected by each injector needs to be calculated based on the magnitude of the coupling and the p-n

forward-bias characteristics. The amount of charge collected by the dynamic node can then be calculated based on the physical layout of the area surrounding the dynamic node and the injector. The minority carrier diffusion length is typically large compared to the size and distance between injecting and collecting diffusions. So all of the dynamic nodes that are less than the diffusion length from an injector must be analyzed. Minority carrier recombination can be included in the analysis, but typically, less than 1% of injected charge is lost to recombination.

When a potential problem is identified, the preferred solution is to modify the interconnect layout to eliminate or reduce the coupling causing the injection. In instances where the coupling cannot be reduced to an acceptable level, diffusion collectors tied to the power rail should be placed in the layout between the injector and the dynamic node.

Minority carrier charge injection is becoming less of an issue as supply voltages are reduced in scaled technologies. The forward-bias turn-on voltage of a p-n junction remains constant at around 0.5 V regardless of supply voltage, so a higher coupling percentage is needed to induce charge injection. For example, with a 5 V supply voltage, 10% coupling is needed on a node to forward bias the p-n junction to 0.5 V. If the supply voltage were reduced to 1 V, the same node would have to be subjected to 50% coupling to achieve the same forward bias. While this discussion focused on capacitive coupling, inductive coupling must be considered as well.

### 8.2.5 Supply Noise and Variation

Dynamic structures are sensitive to variation and noise in the power supply and ground voltages. Consider the example shown in Fig. 8.15 ignoring the effects described in the previous sections. Assume that the dynamic node  $DYN$  is precharged to  $V_{DD}$  and that inputs  $A$  and  $B$  are low. When  $\phi$  goes high, the dynamic node  $DYN$  should remain at  $V_{DD}$ . Resistance and switching noise in the ground distribution network can elevate  $V_{SS}$  at the driver above  $V_{SS}$  at the dynamic receiver forcing node  $A$  to the same elevated voltage. If the voltage on node  $A$  increases above  $V_T$ ,  $M_N$  will turn on and  $DYN$  will discharge. Even when  $0 < V_{GS} < V_T$ , the subthreshold leakage current through  $M_N$  will be greater than the leakage when  $V_{GS} = 0$  V. Reducing the noise and resistance in  $V_{SS}$  network is the ideal solution, but is often difficult to achieve especially when the distance between the driver and receiver is large. Placing the driver as close to the dynamic receiver as possible will minimize the ground variation.

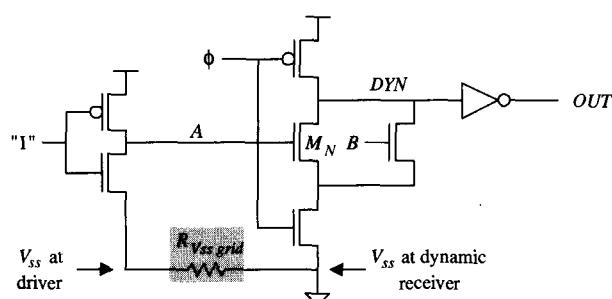


Figure 8.15 Resistance in ground network.

### 8.3 CONCLUSION

Dynamic logic offers a significant speed advantage over complementary static CMOS circuits. One drawback of dynamic logic is that it is more susceptible to noise than static logic. For static nodes, noise results in a loss in performance, since the node recovers state if the chip cycle time is increased. For dynamic nodes, however, state may be lost, leading to logic failures regardless of the cycle time. All of the noise sources discussed in this chapter are cumulative and must be accounted for in the design process. Therefore, dynamic logic requires significantly more electrical verification than static logic. The following set of design guidelines can help reduce the risk of functional failures on dynamic nodes resulting from the noise issues described.

1. **Charge leakage:** Leakage on dynamic nodes should not limit the ability to test packaged parts or wafers on the tester. Calculate the leakage on every dynamic node using worst-case conditions and determine the minimum operating frequency. In circuits where leakage is greater than acceptable, increase the device channel to reduce to subthreshold leakage or add feedback devices.
2. **Charge sharing:** Analyze dynamic nodes for charge-sharing problems. Precharge internal nodes and/or reorder input stacks when the charge sharing is above an acceptable limit.
3. **Charge sharing:** Do not drive dynamic nodes directly into transmission gates. Charge sharing can occur when the source of the transmission gate is changed.
4. **Interconnect capacitive coupling:** Set capacitive coupling limits on the inputs to dynamic gates. The limit should be set to ensure that the input voltage does not rise above  $V_T$ . Reduce the coupling or increase the size of the driver where needed.
5. **Interconnect capacitive coupling:** Set capacitive coupling limits on dynamic nodes. Reduce “bad” coupling capacitance or increase fixed or “good” capacitance on dynamic node.
6. **Back-gate capacitive coupling:** Dynamic nodes driven directly into multiple-input complementary gates should drive the transistors closest to the supply rail to prevent back-gate coupling issues.
7. **Minority carrier injection:** Dynamic nodes should be protected from potential injectors. Coupling limits should be set to eliminate the possibility of injection. In cases where injection occurs, diffusion collectors should be added to protect dynamic nodes.
8. **General:** The range of  $\beta$  ratios for gates driving dynamic nodes should be limited to ensure sufficient noise margin.
9. **General:** The range of  $\beta$  ratios for gates being driven by dynamic nodes should be limited to ensure sufficient noise margin. Increasing the  $\beta$  ratio for NMOS precharged dynamic logic will improve the noise margin and reduce the speed of the circuit.
10. **General:** The output of static gates, and not dynamic gates, should drive the input to dynamic gates. Dynamic gates directly driving into another dynamic gate provide insufficient noise margin.
11. **General:** Static gates should be in close proximity to the dynamic gates they drive. Capacitive coupling, supply resistance, and other sources of noise can raise the output voltage of the static gate and falsely discharge the dynamic node.

**REFERENCES**

- [1] R. H. Krambeck, et al., "High-Speed Compact Circuits with CMOS," *IEEE Journal of Solid-State Circuits*, vol. SC-17, no. 3, pp. 614-619, June 1982.
- [2] T. W. Houston, et al., "Compound Domino CMOS Circuit," *U.S. Patent #5015882*, May 14, 1991.
- [3] K. Bernstein, et al., *High Speed CMOS Design Styles*. Kluwer, Norwell, MA, 1998.
- [4] I. S. Hwang and A. L. Fisher, "Ultrafast Compact 32-bit CMOS Adder in Multiple-Output Domino Logic," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 2, pp. 358-369, April 1989.
- [5] H. Yamada, et al., "A 13.3 ns Double-precision Floating Point ALU and Multiplier," *Proceedings of the 1995 International Conference on Computer Design*, pp. 466-470.
- [6] L. G. Heller, et al., "Cascode Voltage Switch Logic: A Differential CMOS Logic Family," *1984 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 16-17.
- [7] N. Goncalves and H. De Man, "NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures," *IEEE Journal of Solid-State Circuits*, vol. SC-18, no. 3, pp. 261-266, June 1983.
- [8] V. Friedman and S. Liu, "Dynamic Logic CMOS Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 2, pp. 263-266, April 1984.
- [9] P. Larsson and C. Svensson, "Noise in Digital Dynamic CMOS Circuits," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 655-662, June 1994.
- [10] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd Ed. Addison-Wesley, Reading, MA, 1984, pp. 68-71.
- [11] C. Heikes and G. Colon-Bonet, "A Dual Floating Point Coprocessor with an FMAC Architecture," *1996 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 354-355.

Ted Williams  
*MorphICs*

## 9.1 INTRODUCTION

*Pipelining* is the fundamental concept of segmenting combinational logic into stages, and defining the interactions between those stages so that multiple computations progress simultaneously. Throughput can be increased compared to just plain combinational (state-free) logic, by enabling new computations (tokens of data) to be initiated before earlier ones have finished. For proper operation, there must be a defined way that the tokens stay separated and yet progress in time through the stages. This definition of the boundaries also defines the “state” of the system, and the rules for state changes. These definitions differ for synchronously clocked pipelines, wave-pipelines, and self-timed pipelines—the subject of this chapter.

A synchronously clocked pipeline can define the token boundaries with either registers (also called flip-flops) or with latches. Registers define the state transitions crisply, based on the edge of the clock control signal. Latches can either be passing (transparent) or holding (opaque), based on the level of the clock control signal. Discounting the possibility of “time-borrowing,” state transitions are defined by when latches close (become opaque). Also, for the moment, discounting the realities of clock skew, the key distinguishing feature of a synchronous pipeline is that all of its state elements transition simultaneously, and the clock period must be long enough to accommodate the slowest stage. Each of the combinational segments must therefore be designed to be of equal length for optimum performance, as the overall latency is the number of stages times the longest combinational length (plus the propagation delays through the latches or registers).

In stark contrast, wave-pipelining does not add latches or registers in order to create state boundaries. Instead, successive data tokens are simply introduced into the same block of combinational logic, with the plan that they remain separated and can be collected successively, intact, at the output. For this to work, the delays through all possible paths through the combinational logic must be matched, considering both minimum and maximum delays, creating two-sided timing constraints. These constraints are a significant burden, because they must encompass all the possible variability in fabrication process spread, model inaccuracies, and simulation tool inaccuracies. Wave-pipelining does have the advantage that the overall latency is just the sum of the combinational stage latencies, without having to multiply the longest stage delay times the number of tokens present in the pipeline. Additional margin can be added by increasing the time between introductions of new tokens, but this

performance loss may exceed the potential gain made possible by removing the propagation delay of latches or registers.

Self-timed pipelines have the best attributes of both synchronous pipelines and wave-pipelines. Not only can they be designed to have the safety and tolerances of synchronous pipelines, but in addition, self-timed pipelines, like wave-pipelines, can achieve lower overall latency, and increased throughput, even when stage delays are unequalized. A self-timed pipeline controls the movement of tokens across boundaries by local handshaking between neighboring stages, so that all tokens need not cross boundaries simultaneously. This has the advantage that the overall latency is simply the sum of the stage delays, and not the worst stage delay times the number of stages. Also, there doesn't have to be allowance for clock skew since, by construction, the pipeline does not need to synchronize the actions across all boundaries. Unlike wave-pipelining, the control signals in a self-timed pipeline can be designed to move tokens across boundaries exactly when data computations complete. Thus, the logical function will be correct for any actual gate delays, a property known as "speed independence."

Self-timed pipeline structures are a subset of the design styles used in asynchronous design. In the last decade, there has been a resurgence of interest in asynchronous styles motivated not only by potential performance advantages, but also by the ease and flexibility of modular composition, avoidance of the difficulty and power dissipation required for low-skew full-chip clock distribution, possible power savings in specific situations such as those with varying data rates, and even because asynchronous design, which has less synchronization of events at a specific clock frequency, typically has lower peaks of ground-bounce, IR-drop, and generated or radiated electromagnetic interference. Other articles [2], [16] have recently surveyed the broader set of asynchronous design issues and accomplishments; this chapter focuses specifically on self-timed pipelines for datapaths. Even more specifically, the objective here is to focus on native and self-contained datapaths built using precharged logic blocks that can also serve as implicit state-storing latches. Other useful approaches, not emphasized by the analysis presented in this chapter, are hybrids [15], [31] that use various mechanisms to generate local control signals that are then used, as a regularly repetitive clock would be, in circuits built out of traditional combinational logic and ordinary latches or registers.

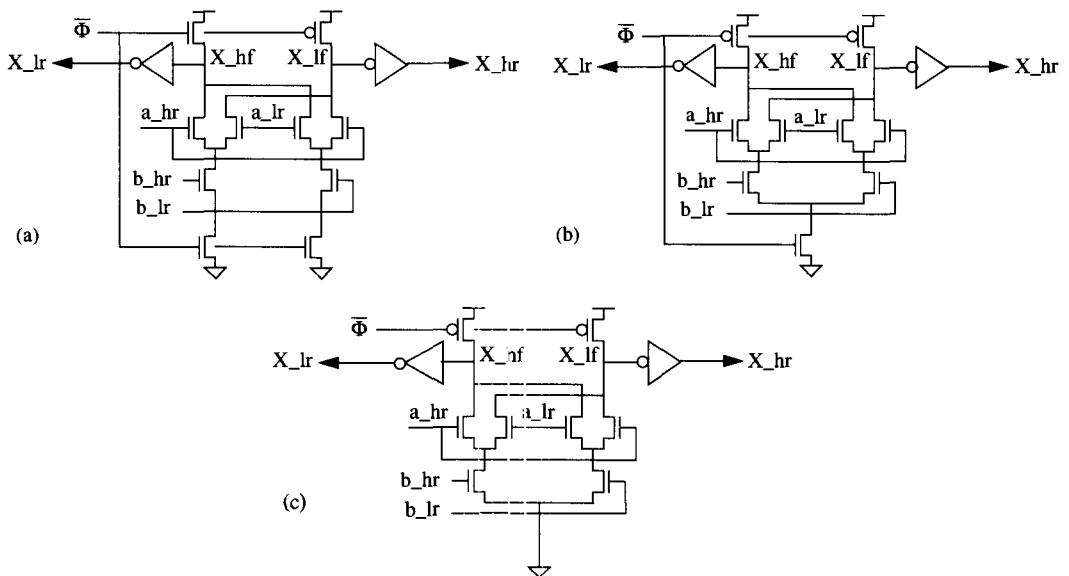
This chapter is divided into sections describing how to design, use, and interface self-timed pipelines. It catalogues many of the typical styles and covers issues of practical interest including quantified performance comparison, margin analysis, testing, and power consumption. Section 9.2 begins by discussing the circuit choices for the logic blocks for the individual stages, and Section 9.3 defines the specialized terms used in describing flow through self-timed pipelines. Section 9.4 describes the varieties of control interconnections, the key concept in distinguishing self-timed pipelines. Section 9.5 goes into detail about how the local characteristics of stages and their handshaking affect a pipeline's overall latency and throughput, including the issue of pipeline bubbles. It is shown that a self-timed pipeline is really a special case of a self-timed recirculating ring, and for that reason it makes sense to start with the generalized analysis. Section 9.6 gives example applications of how these concepts can be used to create timing domains that are distinct from an enclosing synchronous clock; to achieve multipumping of logic blocks; and to interface between different clock domains. Section 9.7 describes some voltage and delay margin issues, methods for handling modeling and block abstractions needed for wafer test pattern generation, and the factors affecting power consumption.

## 9.2 INDIVIDUAL STAGES

The starting point for a stage of a self-timed pipeline is an ordinary precharged gate. Although it is certainly possible to build a self-timed pipeline using non-precharged fully-complementary static gates, there is a natural symbiosis in using a precharged gate for the logic block because it can also act as an implicit latch. By implicit, we mean that, without adding any additional transistors, it can hold its output even after the input signals are removed until the precharge signal is again asserted. Table 9.1 describes the four “conditions” of a precharged gate.

**Table 9.1**

Control input	Data inputs	Meaning or status
Control = 0	Data inputs reset or active	Actively precharging
Control = 1	Data inputs reset	Waiting for data = holding precharge
Control = 1	Data inputs active	Actively evaluating
Control = 1	Data inputs reset	Waiting for precharge = holding evaluated outputs



**Figure 9.1** (a) Precharged gate for XOR/XNOR with separate series-evaluate transistors; (b) Precharged gate for XOR/XNOR with merged series-evaluate transistors; and (c) Precharged gate for XOR/XNOR without series-evaluate transistors.

Figure 9.1(a) shows a typical precharged gate. To prevent false discharges, precharged gates require monotonically (changing in one direction only) rising inputs during the period the pull-down stacks are enabled for evaluation. When each precharged stack is grouped with an output inverter, then the outputs of the inverters are also monotonically rising, and are suitable as inputs to another precharged gate. When taken together, each precharged stack and its output inverter are logically noninverting, so in general both “polarities” (logical true and complement) of each signal need to be generated. The

combination of both polarities is called a “dual-monotonic” or “dual-rail” pair. It is possible to merge portions of the pull-down stacks of multiple outputs, and particularly for the two stacks of a gate generating a dual-monotonic pair. Figure 9.1(b) shows a variation of Fig. 9.1(a), with shared “series-evaluate” or “footer” transistors. Especially when the stacks are merged, this same structure is sometimes called Dual-Cascode Voltage Switch Logic [10]. Another good idea, particularly useful for “address decode” signals that need to generate a unary one-hot result anyway, is to pair together two signal bits using four wires so that, out of each group of four wires, only one of the four wires transitions to indicate the value of both signal bits. This technique can both save power and reduce noise due to fewer cross-coupling aggressor transitions [9].

When a sequence of precharged gates (grouped with their output inverters) is concatenated together *with the same precharge signal*, it is called a domino chain. Within each precharged gate, the purpose of a “series-evaluate” or “footer” transistor is to prevent crowbar current due to fighting during precharging, if some inputs are still active. It is also possible to omit the series-evaluate transistor, as shown in Fig. 9.1(c). Within a domino chain, if all gates have series-evaluate transistors, then precharging of all the stages in the chain occurs in parallel, but where there are no series-evaluate transistors, then precharge ripples down the domino chain, just like the serial rippling of an evaluation waveform. Sometimes, the maximum ripple length through stages can be usefully shorter, if the height of some pull-down networks are already tall. This happens when the N : P resistance ratio of a precharge stack is large enough that its output voltage during precharge can still rise enough to be considered at “logic high,” even without adding a series-evaluate transistor to the bottom of the stack. This typically occurs when the fan-in is three or more, through all possible paths in a pull-down network. In either case, where the serial rippling of precharge can be tolerated, or when the N : P ratio is large enough, the omission of a series-evaluate transistor can be advantageous, both because it can improve the evaluate-delay by lessening the height of the stack, and because it saves area.

There are several varieties for the pull-up head, and its combinations with “keeper” transistors. Sometimes designers will only instantiate the PFET transistor of a keeper inverter, since its pulling-up is in the direction that recovers from a precharged high value from being degraded through charge-sharing. But instantiating the NFET transistor of a keeper inverter is useful too, because it keeps an evaluated dynamic node tied to ground, relevant in the case where the inputs have already been de-asserted, and therefore helps it recover from cross-coupling or power-supply bounce conditions which might otherwise degrade its level.

Precharged gates provide circuit designers with an easy opportunity to skew the ratio of the devices in order to speed the evaluation transitions and improve performance along the critical forward paths. These single-edge optimizations are a key part of the self-timed pipeline style SRCMOS [3]. Single-edge optimization can be applied not only to precharged gates, but also to fully-complementary static gates as well. In the limit, this could buy up to a factor of two in performance because nets may only have to charge transistors of mostly one polarity. For example, the inverter following a precharged net could have huge PFET transistors, and minuscule NFET transistors, to emphasize the speed of only the rising evaluate transitions. Performance is improved due to the increased drive-strength for the transitions that matter, reduced capacitive load of the smaller transistors that drive the slow transitions, and because the highly skewed gates will have lower cross-over current opposing the active transitioning. On the other hand, highly skewed gates will have lower noise margin, and we have to be

particularly careful in self-timed pipelines to keep the reset transitions out of the critical paths. If the reset transitions become too slow due to these single-edge optimizations, then they will undesirably enter into critical paths, or functional failures could even result if violations occur in assumptions made about reset races. SRCMOS emphasizing single-edge tuning is therefore very design-verification intensive, and performance improvements in more complicated structures often prove to be elusive after all issues and paths are taken into account.

The clock is just one possible source for the signal that controls when each precharged block precharges and when it evaluates. When the clock is used directly, this means that each precharged block is precharging for one clock phase, and enabled for evaluation in the other clock phase, meaning the precharged block can process one token of data per clock cycle. Glitches or non-monotonic changes on the inputs to precharged blocks could cause evaluation to the wrong result. To avoid this, inputs to a precharged block must come either directly from latches or registers, or from logic that is stabilized before the precharged block gets enabled for evaluation. Figure 9.2 shows the generally allowed arrangements for combining precharged logic with latches, highlighting the restrictions that are required when using a clock as the precharge control signal. In contrast, if edge-triggered registers were used, no static logic could be placed between register outputs and precharged block inputs, because such combinational logic might generate non-monotonic outputs. Using latches instead of registers lessens this constraint because combinational logic inserted between latch outputs and precharged block inputs can be allowed time to stabilize that is accounted for as time-borrowing from the previous clock phase. Another advantage of using latches is that there can be precharged logic in both clock phases, since precharged blocks can be chosen to precharge in the appropriate clock phases. However, even with latches there are still significant constraints when using the clock as the precharge control signal. Figure 9.3 shows an example of how a subsequent latch or register could capture the wrong data if clock skew delayed its capture of the correct data. An incorrect data value would be captured if the latch turned opaque too late, allowing the precharged block's output to reset before good data was captured and latched. Many of these

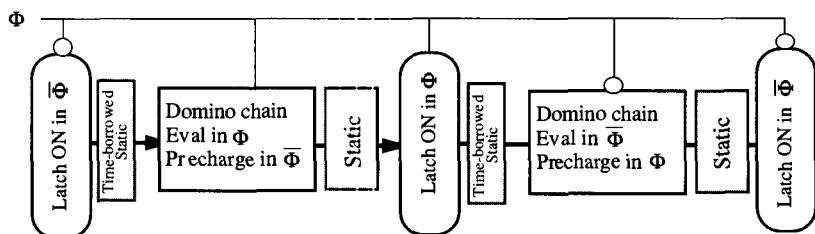


Figure 9.2 Correct combination of precharged logic, static logic, and latches.

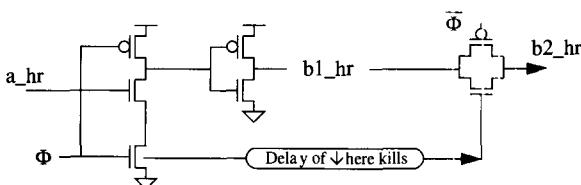


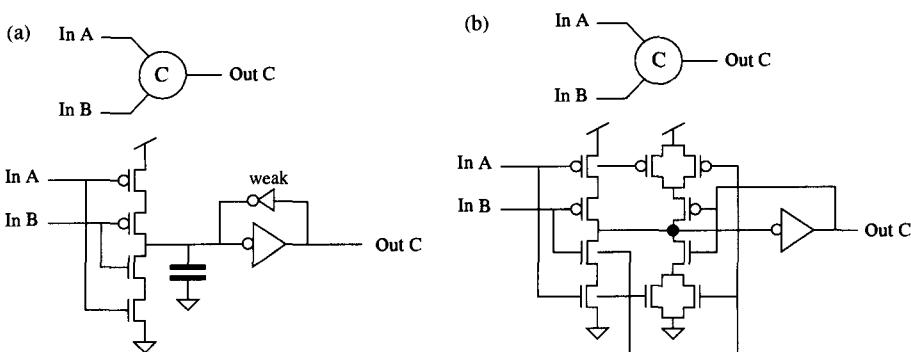
Figure 9.3 Clock skew between a precharged gate and downstream latches can cause wrong data capture.

disadvantages of using the clock as the control signal for precharged logic can be removed or mitigated by moving toward self-timed design, and generating the precharge control signal instead of just using the clock.

### 9.3 DEFINITIONS

The constraints on the precharged control signal for a gate are that it must be removed to start evaluation, and it cannot be asserted again until all the consumers of the gate's output have acted on its data. The beauty of precharged logic is that once a gate's output has evaluated, the output will be stable, even if the inputs are de-asserted, because it is only the assertion of the precharge signal that will cause the output to reset. So, the fundamental concept behind a self-timed pipeline is that we can construct the precharge control signal to give us more choices than just using a phase of the clock. In general, the control signals can be formed either by using "completion-detectors," "matched-delays," or a combination of both. Completion detectors take advantage of the observation that a dual-monotonic pair not only communicates a data value, but also communicates "when" the value is valid. By using a simple OR gate on a dual-monotonic pair we can generate a completion signal for that value. To combine the completion signals from multiple bits of data, the function we want is one that waits until all bits are valid before indicating completion of the group, but also waits until all bits are reset before indicating that the group is reset. This function that waits for all inputs to transition high before setting the output high, and waits for all inputs to transition low before setting the output low is called a "C-element" [14], and Figs. 9.4(a) and (b) shows both a dynamic and static version of a two-input C-element. It turns out that C-elements are the basic building blocks of control signal logic for self-timed pipelines because they enable the control signals to be generated such that the sequencing will be correct for any combination of gate delays, making it "speed-independent" logic.

Although it is also possible to use edge-sensitive (two-phase) signaling, as promoted in [22], we will continue with the styles mentioned in the previous sections using embedded-completion, dual-rail monotonic signaling [20] for each bit, in order for the completion detectors in a level-sensitive (four-phase, or four-state signaling) circuit to distinguish between successive data tokens in the pipeline. This signaling requires stages to be reset (precharged) between separate data elements, and the "space" in the pipeline occupied by these stage(s) that are reset between data elements we call a



**Figure 9.4** (a) Dynamic two-input C-element; and (b) static two-input C-element.

*reset spacer*. Every *token* is therefore composed of one data element and one reset spacer. Since the stage configurations may contain a varying number of series latches, the parameter  $S$  specifies the (possibly fractional) number of stages required to contain the data element and reset spacer pair held statically back-to-back.  $S$  is thus the “static spread” in stages between tokens, which is equal to 2 divided by the number of series latches in a stage. The number of latches includes both explicit latches and the implicit latch provided a function-block is precharged. The reason  $S$  is called the *static* spread is because it measures the number of stages each token (data + accompanying reset space) occupies if the pipeline was filled up to the point of stopping all flow. Remember (unlike a synchronous pipeline), it is possible for the pipeline to “fill up” if the “end” of the pipeline (the final output stage) stopped receiving acknowledgment handshakes, or to “empty out” if no new data tokens were introduced at the “head” of the pipeline.

The local handshaking control signals in a self-timed pipeline keep tokens distinct by enforcing that a token only flows forward into the unoccupied slot. Such an empty stage can be described as a stage containing a “hole” or *bubble*. Bubbles flow backward as they are displaced by data tokens flowing forward. It is important to understand the difference between a reset spacer and a bubble: a reset spacer flows forward and is paired with a data element in every token; bubbles flow backward and the number of bubbles is independent of the number of tokens.

A stage configuration for a pipeline has particular parameters characterizing its local performance and the delays associated with its control interconnections. Unlike a synchronous pipeline, in which the delay from the output of one stage to the output of the next is equal to the period of a global clock, in a self-timed pipeline, the latencies are independent quantities called the per-stage latencies, and are not tied to any global signal. The forward latency,  $L_f$ , is the delay from new valid data outputs at one stage to new valid data outputs from the following stage. The reverse latency,  $L_r$ , is the delay from the acknowledgment of a stage’s output to the acknowledgment of its predecessor’s output. The forward latency can be measured or analyzed independently by observing a data token flowing forward through an initially empty pipeline. Likewise, the reverse latency can be measured or analyzed independently by observing the delays bubbles experience when flowing backwards through a pipeline initially packed with data. Because any packing will consist of alternating data elements and reset spacers, we define  $L_b$  to be the average delay of a bubble displacing a data element and displacing a reset spacer. Considering just the average is sufficient because bubbles will always displace equal numbers of data and reset elements. Each stage has a local minimum cycle time,  $P$ , which includes the delays of all the transitions necessary for a stage to reset and become enabled again for the evaluation of the next token. The per-stage latencies and cycle time for particular stage configurations can be determined in detail from the individual gates by the dependency graph analysis method in [26], [29], or can just be observed from normal circuit simulation.

## 9.4 SELF-TIMED CONTROL INTERCONNECTIONS

Figure 9.5 shows an example of a three-stage segment of the simplest speed-independent self-timed pipeline. Each stage uses a C-element to combine the completion detector outputs from neighboring stages such that a stage precharges when both of two conditions are met. These two conditions are (1) that the users of a stage’s outputs (the “successors”) have finished evaluating, and (2) that the stages inputs (from the

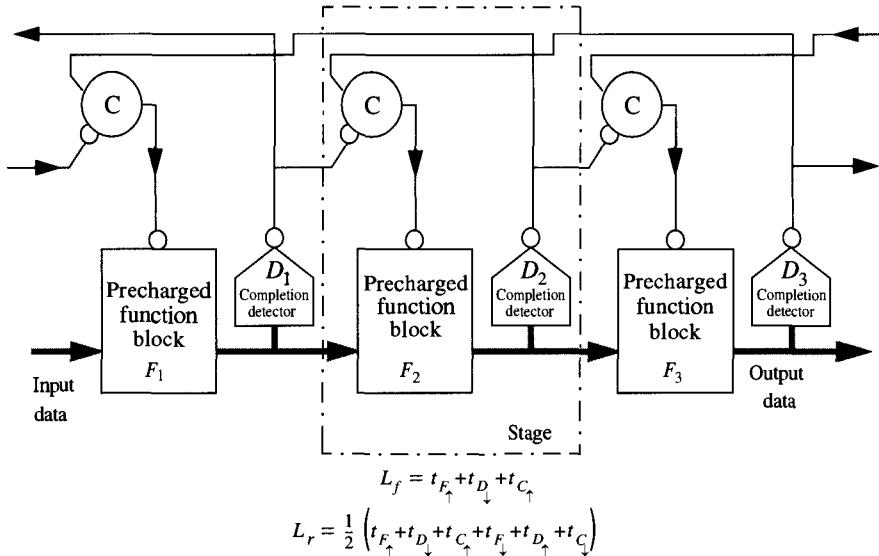


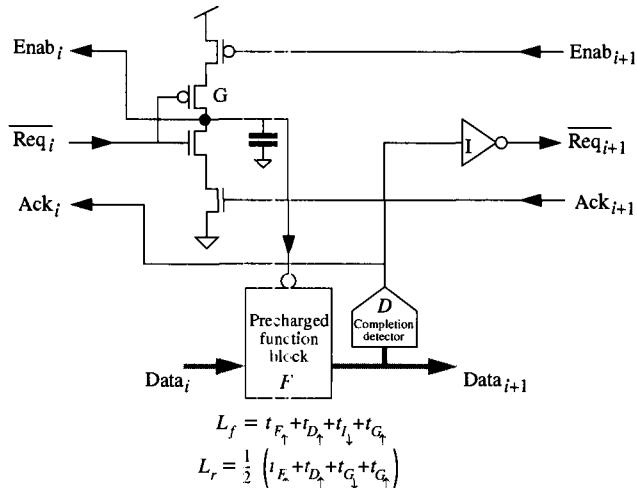
Figure 9.5 Three-stage self-timed speed-independent pipeline.

“predecessors”) have themselves finished precharging. Likewise, the same C-element enforces that a stage is enabled for evaluation when (1) its predecessors have finished evaluating, and (2) its successors have finished precharging, so that a token of data does not run into the one ahead. The up-arrows and down-arrows in the figure refer to the propagation delays, respectively, of rising and falling output transitions.

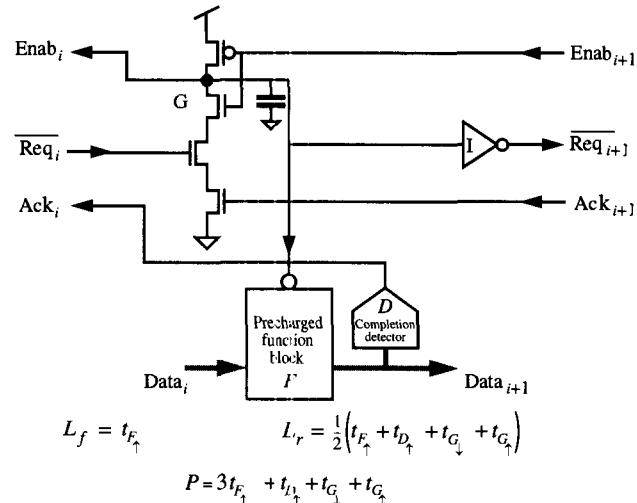
The disadvantage of the simple control connections using the stage pattern in Fig. 9.5 is that tokens progress forward down the pipeline too slowly. The control connections enforce logical correctness, but at the expense of some performance, namely, because the forward latency  $L_f$  includes series propagation delays through a completion detector and C-element in every stage. The theme of good self-timed pipeline design is how to go from such known good construction toward better performance without introducing hazards that cause failure, or too much engineering tuning to dance on the edge of potential hazards.

A first step in removing unnecessary control dependencies is to change the connections to not delay evaluation of a stage until after its successor has finished resetting. It suffices to prevent race-through of tokens simply to check that the successor has *started* resetting, as long as the first gate has a footer transistor or enough ratioing to ensure precharge wins even if valid data are still present at the inputs. We can therefore modify the control logic in each stage to that shown in Fig. 9.6, which lowers the reverse latency by removing the terms for the function block reset delay. This control logic still maintains speed-independence: the correct sequencing is ensured for any combination of rising and falling delays of all the gates.

We can remove more dependencies, and make the control logic less likely to enter into the critical paths we care about by making another observation and one assumption. The observation is that since the data signals are monotonically rising during evaluation (meaning they contain embedded completion), we don’t need to make the enabling of a block’s evaluation dependent on the detection of valid data at its inputs. The assumption that we need to make is that each stage’s predecessor resets no slower



**Figure 9.6** Self-timed pipeline stage with lower reverse latency, still speed-independent.



**Figure 9.7** Minimum-latency self-timed pipeline stage, with zero latches or control logic in the path of forward-flowing data tokens.

than its successor can evaluate. This is usually quite reasonable because internal domino chains within a stage can reset in parallel versus evaluating in series. (But, we must also verify that resetting didn't get too slow because of ratioing for single-edge optimizations.) With this one assumption, we can change the control logic to make the forward latency of a stage equal to just the function block evaluation time. Figure 9.7 shows more aggressive minimum-latency stage control logic that enables each stage for evaluation even before new data arrive, so control delays are completely removed from the forward propagation path. This achievement of “zero-delay-overhead” [27] as compared with the raw function block delays is one of the motivations for using self-timed pipelines as compared with synchronous pipelines that introduce latch or register delays.

Several other variations of control logic connections have been used that have even more assumptions that need to be validated during design. Although the designers of these control styles often think that they are achieving “higher performance” as compared with the connections of, for example, Fig. 9.6 or Fig. 9.7, what they actually gain instead is usually a tighter spacing of data tokens in the pipeline, which can improve throughput, but not latency. Section 9.4 will describe these trade-offs in more detail, and it is indeed the comparison of the different control styles that necessitates clear analysis and distinction between latency and throughput.

Conceptually, instead of fully checking whether both predecessors and successors have finished both the precharge and evaluation appropriate for an “advancement” of tokens, control logic can instead just check “some” of the information. The idea of “precharging again as soon as a block has finished” motivates the name “postcharge” with the idea that it emphasizes that precharging is triggered “after” evaluation, rather than being done “before.” Of course, it’s all the same thing, since one stage’s “after” is another stage’s “before.” Typically, the name “postcharge” refers to using simple inverted or direct connections to subsequent signals, as shown in Figs. 9.8 and 9.9, which initiate precharge when a stage or a successor stage has finished evaluating. When a precharge signal is taken from the stage’s own output, as in Fig. 9.9, there is an unchecked race that the successor stage must act on the data before the data go away. Such races are further aggravated, of course, by the reality that signals do not transition instantly, and races lead to runt pulses and partial transitions, which can leave signals hanging at an intermediate voltage level.

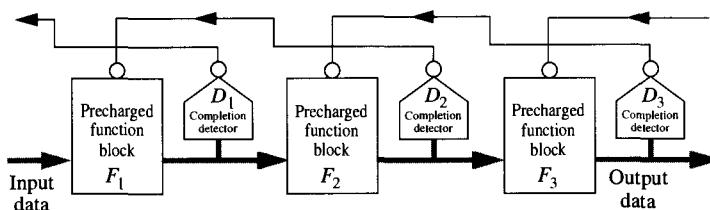


Figure 9.8 “Postcharge” connections with reset from successor stage.

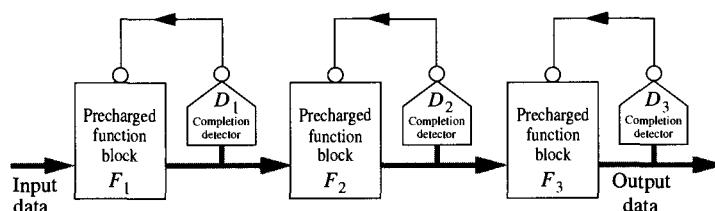


Figure 9.9 “Postcharge” connections with reset from each stage’s own outputs.

Another variation in the local control connections is taken by “delayed-reset logic,” shown in Fig. 9.10, which introduces the additional concept of a signal supplied from the head of the pipeline that defines when to precharge (reset) a stage, along with each stage feeding a delayed version of its reset signal forward to its successor [25], [12]. In delayed-reset logic the sequencing is “evaluate,” “reset,” and “recovery” where the reset action is initiated by forward-flowing signals. The “recovery” action (which

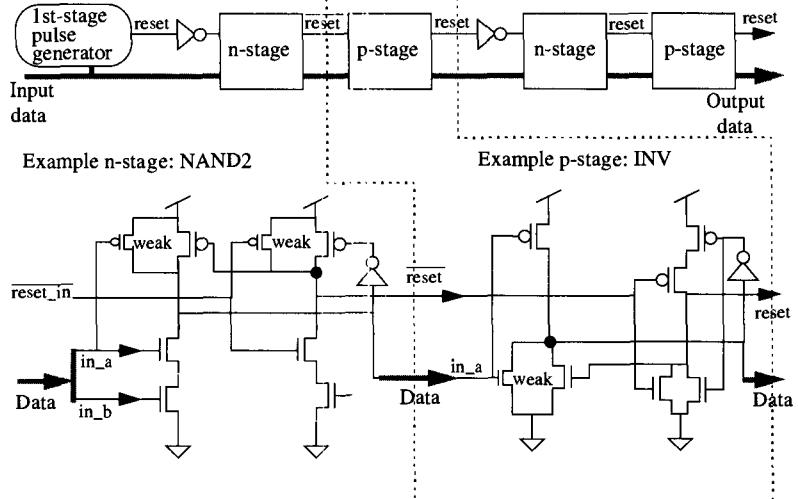


Figure 9.10 “Delayed-reset” stage style, with forward-flowing reset initiation.

removes reset, to enable the next evaluation) is then initiated by the backwards connection that contains a race similar to the one in the postcharge logic of Fig. 9.9. Sourcing the reset signal from the pipeline head has the disadvantage that there must be a pulse generator calibrated to the width needed by the slowest resetting block downstream, without any active feedback from completion detectors. On the other hand, an advantage of a forward-flowing reset initiation is that it conveniently allows the choice of holding the evaluated states arbitrarily, which can aid in testing or probing. In contrast, most other self-timed pipeline control interconnection styles, including SRCMOS [3], initiate resetting based on backwards flowing control from successor stages, instead of interlocking with a separate signal sourced from the pipeline head.

It is clear that there are many variations of how to reset/precharge/postcharge dynamic logic in self-timed pipelines. The next section answers how we quantitatively can compare these choices.

## 9.5 OVERALL PIPELINE LATENCY AND THROUGHPUT

We can predict the latency and throughput of an ensemble of pipeline stages based on the local properties of those stages, and the total number of stages and tokens. When a self-timed pipeline has its output fed back around to its input so that data recirculates fully under self-timed control, this structure is called a self-timed ring, and can be useful for computation of iterative functions. It turns out that, both analytically and experimentally, a self-timed pipeline can be considered a special case of a self-timed ring, and for that reason, we derive the analysis of latency and throughput for the more general case of a ring. Since every data token consists of one data element and one reset spacer, the serial distance in pipeline or ring stages occupied by a data token is the same space as for two bubbles (remember, as defined in Section 9.2, bubbles are the “empty slots” that flow backwards as data and reset spacers flow forward). A pipeline or ring with  $N$  stages and  $K$  tokens therefore contains  $2((N/S) - K)$  bubbles, where again  $S$  is the number of stages occupied by each token, accounting both for its data element and

reset spacer. If there is not at least one bubble, then the self-timed handshaking would not be able to “move” any tokens forward. The number of tokens and bubbles in a *pipeline* can change if the input or output rates fluctuate, but in a ring, the number of tokens and bubbles remains fixed after the ring is initialized.

When self-timed stages form a ring, the ensemble has a total latency and total cycle time. Input and output operations “exchange” new tokens for processed tokens so the number of tokens kept circulating in the ring is held constant. The total latency, denoted by  $\lambda$ , is the delay between the introduction of a new data token into the ring and the removal of the corresponding processed token after the number of iterations necessary for the token to have passed through, in all,  $G$  function evaluation stages. If the number of stages in the ring is increased, then a given token will need to loop around the ring fewer times before it is completed. Since all the other tokens in a ring get exchanged with new data during the time it takes for one token to complete  $G$  function evaluations, the overall throughput of the ring is given by

$$T = \frac{K}{\lambda} \quad (9.1)$$

While this same equation is indeed true for both synchronous and self-timed pipelines, in self-timed pipelines/rings there is a fundamental additional complication (a key subject of this section), namely that latency can be degraded by having too few stages, if it turns out performance is dominated by bubble propagation instead of data propagation. So, in a ring holding a fixed number of tokens, do not think of latency and throughput as trading off with each other, but instead consider both latency and throughput as trading off with the number of stages, to which the ring area is proportional.

Table 9.2 summarizes the parameters defined so far. While this chapter considers these parameters to be constants for particular circuit choice styles, similar conclusions and design principles also hold even if delays are considered to be random variables (changing for example, due to data dependencies), as was done for a simplified pipeline model in [1].

**Table 9.2** Summary of Parameter Definitions

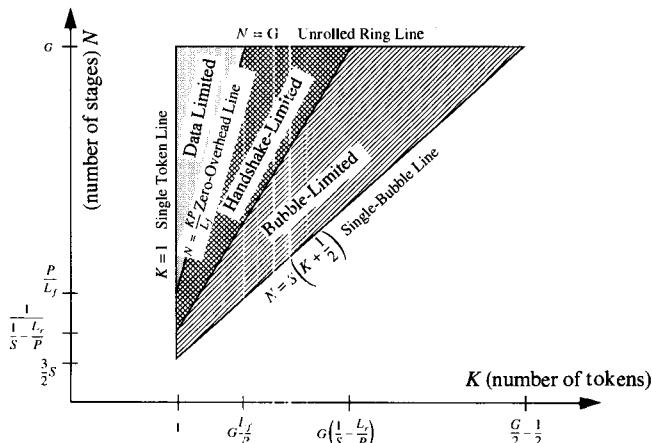
Name	Description
$G$	Number of function evaluations for given problem
$N$	Number of stages in pipeline/ring
$K$	Number of tokens in pipeline/ring
$S$	Spread between statically packed tokens
$L_f$	Per-stage forward latency of tokens
$L_r$	Per-stage reverse latency of bubbles
$P$	Local cycle time of stages
$\lambda$	Total latency of pipeline/ring
$T$	Throughput of pipeline/ring

### 9.5.1 Ring Performance Graphs

Analyzing the performance of a ring means determining the total latency and total cycle time as functions of the local parameters  $L_f$ ,  $L_r$ ,  $P$ , and  $S$ , which describe the particular stage configuration chosen. Such an analysis is relevant to guide design choices for the stage configuration and number of stages. For example, during the

design of the commercially shipping microprocessor in [30], this analysis was used to determine how many stages (and therefore area) were needed for one-token and two-token rings, allowing a decision between these alternatives.

The performance of a self-timed ring can be limited by different causes. The number of stages and number of tokens in a ring determine which of the causes predominates. The possible limiting considerations define regions of values for  $N$  and  $K$  in which different relations expressing ring performance apply. Therefore, the most basic performance diagram, as shown in Fig. 9.11, is a graph, having  $N$  and  $K$  as the axes, that defines the applicable regions for particular equations specifying total latency and throughput. Three lines enclosing the entire valid region of ring operation bound the possible values for  $N$  and  $K$  in a self-timed ring. The top line, for which  $N = G$ , is the *unrolled ring line* because it represents the degenerate case of a ring completely unrolled into a pipeline for accomplishing the given  $G$  function evaluations. The left edge of the valid region in Fig. 9.11 is the  $K = 1$  *single-token* line. The right edge is the diagonal *single-bubble* line where  $N = S(K + 1/2)$ . Values of  $K > (N/S - 1/2)$  are not possible because a self-timed ring must have at least one bubble for data to circulate at all. There are three possible regions of operation within the triangle formed by the three boundary lines. One region, marked *data-limited* is where the token flow rate is limited by the forward latency; the other, marked *bubble-limited* is where the token flow rate is limited by the reverse latency. These names apply because limitation by forward latency occurs when stages wait for new data, and limitation by reverse latency occurs when stages must wait for new bubbles. The third region is called *handshake-limited* because the local cycle time constrains the performance within it.



**Figure 9.11** Performance regions in self-timed rings.

We'll now give examples of some key equations giving the overall latency and throughput for each region. Within the *data-limited* region, so few data tokens are in the ring that there are plenty of bubbles, and therefore the performance is determined entirely by the forward latency of the tokens. The total latency within this region is

$$\lambda = GL_f \quad (9.2)$$

and from Eq. (9.1) the throughput is thus

$$T = \frac{K}{GL_f} \quad (9.3)$$

Within the *bubble-limited* region, so many data tokens are in the ring that the low supply of bubbles limits the rate at which data can flow. Since the backward flow rate of bubbles is specified by the reverse latency of the stages, the total throughput is

$$T = \frac{1}{GL_r} \left( \frac{N}{S} - K \right) \quad (9.4)$$

which is just the number of bubbles in the ring times the rate at which they flow.

The *handshake-limited* region exists if  $P > S(L_f + L_r)$ , meaning the local stage cycle time can limit the total performance. Within this region, the throughput is

$$T = \frac{N}{GP} \quad (9.5)$$

If  $P \leq S(L_f + L_r)$ , it means the control logic in the stages is fast enough that the ring's performance is not limited by local control logic handshaking considerations. In this case, the *handshake-limited* region degenerates into a single line called the *max flow* line, on which the flow rates of tokens and bubbles are matched for maximum performance. The intersection of (9.3) and (9.4) determines this line has the equation

$$N = KS \left( 1 + \frac{L_r}{L_f} \right) \quad (9.6)$$

The *handshake-limited* region's boundary with the *data-limited* region is where (9.3) and (9.5) intersect, which is the line with equation

$$N = K \frac{P}{L_f} \quad (9.7)$$

This line is called the *zero-overhead* line because only rings to its left operate without any overhead caused by control logic.

We have verified the performance equations in this section by computer simulations [26] of several specific pipeline configurations with varying numbers of stages and tokens.

### 9.5.2 Performance Region Edges

The edges of the performance graphs deserve special attention because they specify both desirable and undesirable limiting cases. The points where the three diagonal lines in Fig. 9.11 cross the vertical  $K = 1$  *single-token* line are particularly significant. The lowest point at  $[K = 1, N = (3/2)S]$  is the point giving the absolute minimum number of stages,  $(3/2)S$ , in a ring. A ring with fewer stages cannot be self-timed with embedded completion signaling, since it would not provide sufficient space for a single data element, reset spacer, and bubble to circulate.

The desired region of operation for the lowest latency is the *data-limited* region, where  $\lambda = GL_f$ . The best throughput, or the smallest area for a given throughput, occurs on the boundary between the *data-limited* region and the *handshake-limited* region, which the *zero-overhead* line specifies. The best design goal is therefore the lowest point on this line able to achieve the desired throughput. If there is not a specific constraint on the throughput, then the unique desired operating point is where the

*zero-overhead* line intersects the *single-token* line at  $[K = 1, N = (P/L_f)]$  because this point achieves zero overhead with the fewest stages. Rings with a single token are therefore the preferred case when latency or area is the most important consideration. If there is a specific need for higher throughput, then other points on the *zero-overhead* line can satisfy it with the additional area cost of more stages and tokens. Multiple token rings may also be called for if there is a specific need for buffering or delaying more data tokens in the FIFO queue formed by a self-timed ring.

The top edge of Fig. 9.11 is the *unrolled ring* line, on which there are  $G$  stages to accomplish the  $G$  function evaluations, and iteration is not necessary. This line, therefore, elegantly describes a self-timed *pipeline* as the special case of a ring that is fully unrolled. A self-timed pipeline's total latency and throughput are characterized by the same three data-, handshake-, and bubble-limited regions of operation. The *data-limited* region describes a pipeline limited by its input rate. The *bubble-limited* region describes a pipeline limited by its output rate since bubbles are introduced at the output. The *handshake-limited* region describes a pipeline limited by local cycle time constraints along its length. Since  $N = G$  on the *unrolled ring* line, the maximum throughput of a self-timed pipeline is  $1/P$ , and the pipeline achieves this rate when the number of tokens it contains in the steady state is within the range given by

$$N = \frac{L_f}{P} \leq K \leq N\left(\frac{1}{S} - \frac{L_r}{P}\right) \quad (9.8)$$

The lower right *single-bubble* diagonal edge of Fig. 9.11 is the least desirable condition for a self-timed ring. Along this edge, the total throughput is  $T = 1/(2GL_r)$  and the total latency is  $\lambda = 2GL_rK$ . The factor of 2 occurs because the single bubble must make two cycles around the ring for all of the tokens to advance once: one cycle for the data portion, and one cycle for the reset spacer portion of each token.

Figure 9.11 illustrates values of  $N$  and  $K$  that dictate the different regions of performance. Slices through these diagrams can more clearly illustrate the resultant latency and throughput characteristics for specific values of  $N$  or  $K$ . The end points of the slices correspond to the edges discussed in the previous section. For example, Fig. 9.12 shows throughput versus  $K$  for a fixed value of  $N$ . The graph has three line

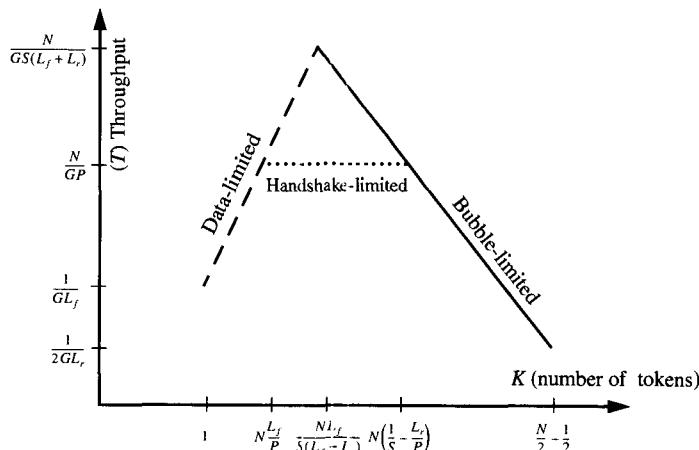


Figure 9.12 Throughput versus  $K$  (tokens) while  $N$  (stages) is constant.

segments corresponding to the three regions in Fig. 9.12. The dashed line segment shows the *data-limited* region, in which the latency and throughput are given by Eqs. (9.2) and (9.3). The solid line segment shows the *bubble-limited* region, in which the throughput is given by Eq. (9.4). Where there is a dotted line segment from the *handshake-limited* region, it clips the attained throughput to the value given by Eq. (9.5).

For a given number of tokens, a ring's latency can be improved by increasing the number of stages up to  $K(P/L_f)$ . Likewise, for a given number of stages, Fig. 9.12 illustrates how increasing the number of tokens increases the throughput while the ring is still within the *data-limited* region, but adding too many tokens causes the ring to enter the *bubble-limited* region. When there are so many tokens that there is room for just a single bubble in the ring, the throughput degrades to a level even lower than the throughput for just a single token because the bubble must circulate twice in order for all the data elements and all the reset spacers to advance once. The trapezoidal shape of this graph was observed by early pioneers such as Molnar and Sutherland building self-timed circuits decades ago, but they did not have the generalized theory to explain it until [27].

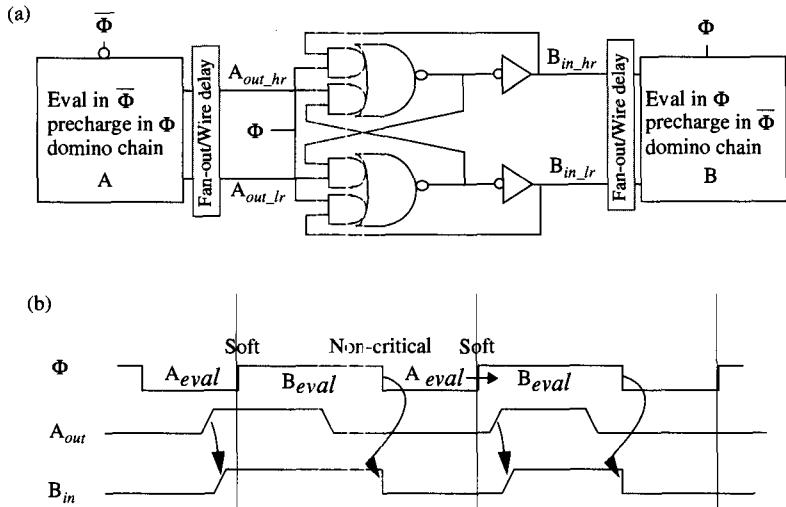
The coefficient of the *zero-overhead* line, the value  $P/L_f$ , is a critical parameter, specifying the minimum number of stages necessary per token to achieve zero overhead. This value, which can also be called the desired “wavelength,” includes the stages occupied by a flowing token's data element and its accompanying reset spacer, along with bubble space sufficient to allow it to flow unimpeded by the other tokens. Tables collected in [26] provide values of the wavelength for a variety of stage configurations. The preferred stage configuration in Fig. 9.7 typically requires from three to six stages per token to achieve *zero-overhead* operation, depending on the evaluation delay of the function blocks relative to their precharge delay and the completion detector delay. Using this principle, a self-timed ring can achieve a total latency equal to just the sum of the raw function block delays of the stages through which data must pass, without any added delays due to latches or control logic in the forward-flowing data path.

## 9.6 APPLICATIONS

Another key advantage motivating the use of self-timed pipelines is their ability to accommodate the propagation delays of logic blocks that don't line up nicely with a chosen clock period. Without self-timing, precharging can only be initiated or removed at the start of each clock phase, but this may leave significant unused time, or even worse, not enough time to complete the desired logic operation within the phase. Delaying clock edges is one way to “adjust” the positions in time of the clock edges to where they are needed, but these “delayed” clock edges are, in fact, just a simple form of self-timing that is matching the chosen delay to the logic requirements. Matched delays can work fine if there is enough margin to account for the variabilities and inaccuracies in both sides of the matched paths, where the race is between the path delaying the clock, and the logic path whose propagation delay creates the original need. But, matched paths are not as “robust” as using logic that checks appropriate completion conditions, which led us to the safer forms of self-timed pipelines discussed in the previous sections.

Self-timed pipelines can operate within a synchronous system by initiating an operation at a clock edge, and returning a result back into the synchronous system either one or more clock cycles later. Typically, if the result is returned within one or two clock cycles, then the designer ensures through analysis that the result from the

self-timed pipeline will be available by the required time, just as the timing of any path through synchronously clocked logic must be analyzed to ensure compliance with the desired clock period. An example of such a usage is in [8] where the paths through a three-cycle floating-point multiplier pipeline are made independent of the position of the intermediate clock edges. The self-timed pipeline allows the division of pipeline stages to be made more flexibly, as long as the total latency fits within the three cycles allotted in the enclosing synchronous system. This design uses mousetrap dual-monotonic latches, shown in Fig. 9.13(a) and (b), as the division between some of the pipeline stages. These latches allow a dual-monotonic signal to arrive either before or after each control signal edge. When the  $\Phi$  signal is high, the latch accepts a new dual-monotonic input, which can already be present before  $\Phi$  rising, or the latch will stay reset until the data do indeed arrive. The cross-coupling of the latch stores the data value even when the data input from the preceding stage resets. When  $\Phi$  goes low, the latch resets, but this transition is designed to not be part of the important forward-flowing data critical paths. Thus, the exact timing (skew) of neither  $\Phi$  edge enters into the critical paths.



**Figure 9.13** (a) Mousetrap latch makes intermediate clock edges noncritical; and (b) timing diagram for mousetrap latch.

Self-timed pipelines that can operate in a variable number of clock cycles, and signal their completion early enough to the scheduling circuits in the enclosing synchronous logic, can provide great benefit by being tuned for the best average case performance instead of degrading the average case just to be able to occasionally handle some rare worst cases. The design of the RAPPID (“Revolving Asynchronous Pentium Processor Instruction Decoder”) [18] exemplifies a factor of three performance advantage to a previous synchronous implementation of the same function.

Even more generalized is the self-timed ring used for floating-point division in [28], [30] where the division operation takes nominally seven cycles, but the exact number is not fixed in the hardware, and is instead programmable, so that the tuning and choice of number of synchronous cycles used can be based on measurement rather than hardcoded during chip design. Making variable the number of synchronous cycles

enclosed can allow the embedded self-timed pipeline to be tolerant to changes, either in its performance or in the performance of the other portions of the design. For example, if there is some other path on a chip that turns out to be slower than expected under particular fabrication conditions, and this necessitates slowing the clock frequency for the whole chip, then the number of clocks allowed for an embedded self-timed pipeline portion can be reduced, and still get the same absolute number of nanoseconds. Thus, performance can be improved relative to a design that would have taken a fixed number of cycles, even if the cycles had to become slower for outside reasons.

This latter design example [28], as well as [19] also illustrate the principle of “multi-pumping,” or using self-timed pipelines to internally iterate logic, so that a given logic gate can be reused more often than once per external clock cycle, or at a different granularity (such as using it three times every two clock cycles). Being able to stimulate a combinational logic block independently of the clock period is a valuable feature and motivation of self-timed pipelines.

Since self-timed pipelines also “decouple” the timing relationship of their input and output, another valuable use is to act as a buffer between different clock domains, where there is potentially an asynchronous (either nonintegral, or nondeterministic) relationship between the clocks. Such a usage of self-timed pipelines is at the heart of the long history of “mesh-routing chips” and the basis of the products of Myricom [21] and shipped by Hewlett-Packard, which perform network routing and clock resynchronization between computers with potentially different and asynchronous clocks. Of course, this usage, and any usage where a completion signal at the output of a self-timed pipeline is “sampled” to decide on which clock cycle to take some action in an enclosing synchronous system, is theoretically subject to metastability at the point of sampling. It is a fundamental principle that it is not possible with 100% certainty to correctly make a binary decision in finite time from sampling a signal that has a possibility of being in the middle of a transition at the point of sampling. Rather, we can compute a probability that the decision is settled (to one of the binary values) based on the probability of the signal being sampled not being “right in the middle” and the amount of waiting time allowed for feedback in the sampling circuit to pull the decision to one of the rails. The probability improves exponentially with increased waiting time, counted up to the net where the sampled data fans out to more than one destination. This is why a common strategy for the sampling of asynchronous signals in general (and the completion signals of self-timed pipelines in particular) is to use two registers before branching the signal to affect other combinational logic. But there is nothing magic about two registers; all that matters is the settling time allowed for the feedback to resolve, and the same probability of success for a binary decision could be obtained by using a single register clocked at a lower repetition frequency.

Taken to the extreme, one could avoid the sampling to get back into synchronous clock cycles by building an entire chip using self-timed handshakes. The big problem with making all interactions dependent on C-elements waiting for request-acknowledge handshakes is the handling of cases where the dataflow does not guarantee if, or on which path, an event will come. In this case, the control must include arbiters, which are blocks that have multiple inputs, each with a request-acknowledge protocol, and can make a decision (output change) while potentially considering either input but without waiting for events on both. There have been attempts at designing entire microprocessors [23], [4], [5], [13] using self-timed pipelines with arbiters, but these have struggled hard in trying to achieve a performance advantage relative to comparable synchronous logic.

The complexity of requiring arbiters in general is a big reason why self-timed pipelines are much better suited for unidirectional dataflow macros, such as memory arrays, caches, and arithmetic circuits, whose state flow is simple. Still, sophisticated efforts have produced some good results. The most promising of the self-timed processors is AMULET3 [6], which gained performance from tuning and well-chosen matched delays without full completion detectors. The commercially successful project in [24] demonstrates a multimedia processor that attains significant power-savings in a design with widely varying data rates by taking advantage of the data-driven nature of self-timed pipelines. On the other hand, despite its good performance, it could likely be argued, for this and other designs too, that the designers simply “chose” to do the design using self-timing, and could have likely achieved similar results in a synchronous design if they had chosen to use extensive gated clocking and to handle synchronous pipeline advancement control logic and bypassing with a sufficient amount of design attention.

## 9.7 MARGIN, TESTING, AND POWER ISSUES

A speed-independent circuit will function correctly for any combination of actual delays, but as we introduce optimizations to increase performance, we must ensure that any timing hazards introduced will have sufficient margin at all process corners and operating conditions. In a synchronous circuit, some timing constraints (the “set-up” relationships) can be relaxed by slowing down the clock speed, whereas other constraints (the “hold” relationships) cannot. Constraints that have been introduced into a self-timed pipeline design by making assumptions must be verified to have sufficient “delay margin” just like the “hold” constraints in synchronous designs.

Delay margin is a form of noise margin, and likewise, conceptually must account both for known “aggression sources” as well as allow for calculation inaccuracies. Calculable aggression sources affecting delay include cross-coupling, power-supply droops and bounces (both due to resistive and inductive effects), and changes in transistor currents due to process spread, and operating voltage and temperature conditions. Inaccuracies affecting delay include miscalculations of parasitic capacitances from extraction tools, and timing inaccuracies from circuit simulation tools or from the use of approximations in modelling timing. Typically, simulation tools do not have direct ways of verifying that all places where races have been introduced actually have quantifiably enough delay margin. This is one reason why the methodology for the design of self-timed pipelines needs to start with speed-independent constructions, and races should be introduced only where doing so has high leverage in performance improvement, and moreover, only under the control and scrutiny of an insightful and rigorous designer.

Traditional static timing analysis doesn’t help self-timed pipeline designs because the interesting cases have feedback (cycles in the logic dependency graph) that can’t be handled by the common static timing analysis algorithms. Vector-based simulation only verifies a particular choice of data patterns, process, and parasitics. Further, cases with only negligible “delay margins” can be hidden or missed because the usual simulation tools only seek to produce each output, without analyzing sensitivities. Tools like SIMIC [7] do calculate delay margins at all gates where the simulation results would differ (create a pulse, or leave a different value at an output node) as a result of having a different fan-in branch win a close race. In this type of tool, chosen criteria can be set for delay margins, just like designers typically choose how much margin to build into setup&hold checks at latches.

Sufficient delay margins in self-timed circuits can also be evaluated using normal deterministic simulators (SPICE, STAR-SIM, TIMEMILL, or IRSIM), by varying parasitic capacitances according to some criteria and resimulating each case to verify continued correct functionality. Ideally, every capacitance variation could be resimulated at every transistor process corner, but this exhaustive simulation is infeasible in practice. Where specific designer attention is affordable for specific individual race verification, the correct verification, of course, increases all parasitics that would slow a path needed to be faster, and decreases all parasitics that would slow another path that is needed to be slower, and then resimulation can verify that the path pairs still have the required relative ordering. But, there are other techniques that attempt to catch problems without requiring specific designer attention to individual races. Some good parasitic capacitance variance strategies are: to scale all nodes up or down by a constant factor (while leaving the transistor gate capacitances unchanged, so relative dependencies are stressed), or to statistically scale nodes up or down by constant factors (e.g., each node has a 50% chance of being multiplied by 1.5 or 0.3), or to scale nodes within a particular hierarchical block or whose names match a particular criteria (such as containing the string “reset”) by a factor, leaving other nodes unchanged.

Wafer-probe testing of circuits containing self-timed pipelines has many issues in common with the more general issues associated with the testing of any precharged circuits, and some unique issues due to the pipelined aspect. We discuss first the testing issues applicable to all precharged circuits, including those whose precharge control signal is just a phase of the clock. Typically, wafer-probe test patterns are derived from automatic test-pattern generation (ATPG) software, which generates patterns that will find faults that can be sensitized (controlled) and propagated to an output (observed) according to a model of failure called the “stuck-at” model, which assumes that the problem is a particular net shorted to either power (stuck at one) or ground (stuck at zero). This model is used because it is well behaved and tractable, but it does not specifically address the realities of precharged circuits that are designed to operate with conditions where some outputs transiently hold their value, without being actively driven. For example, if it so happened that a particular net in a circuit only changed twice during the entire test sequence, where the first correct value was high (and could therefore detect a stuck at zero fault), and the second correct value was low (and could therefore detect a stuck at one fault), and if it so happened that the initial value (occurring when power was first applied) on that node was high, but that the transistor that would have precharged it high was faulty, then the test pattern would not detect the fault. In this type of case, the test coverage of a chip containing precharged gates can be increased by running the test pattern sequence twice, because the test pattern sequence (which, for full stuck-at coverage, must set every net to both states sometime during its run) would then be assuredly testing both directions of *transition*. However, even running the stuck-at patterns twice does not guarantee detection of other failure modes possible in precharged circuits, such as weak resistive faults which may leak away charge and degrade noise margin.

To supplement stuck-at fault testing, another testing technique, called  $I_{ddq}$  testing, seeks to find faults caused by shorts or excessive leakage by measuring total power-supply current while inputs are not changing, and normal static CMOS logic would therefore be drawing only infinitesimal current. Dynamic logic can also be tested with  $I_{ddq}$  testing [17], if leakage doesn’t dominate. To make this work, as was done successfully in [30], the control signals for the stages of a self-timed pipeline just have to have a

mode that forces a quiescent state with no contention, and actively precharges all precharged stacks so that no nodes are undriven (potentially leaking).

The additional testing issues that result from using self-timed pipelines, beyond those of all precharged circuits, result from the possible concealment of state. While it is possible to add scan paths into the state points of self-timed pipelines [11], this has rarely been done in practice, both because of the complexity, and because the very reasons for using self-timed pipelines, namely high-speed and fine granularity, may oppose adding scan paths into the latches. Instead, it is often better to model a self-timed pipeline for testing as just one long combinational logic block that lets every applied test vector flow fully from input to output down the self-timed pipeline. This works because test patterns applied at wafer-probe time typically are applied at a much lower repetition rate than a product's eventual clock frequency, and thus will allow enough time for data to flow through all of the stages of a self-timed pipeline. For example, in [28], even though the self-timed pipeline (with iteration) would take seven cycles at the chip's normal operating frequency, the test patterns were generated to allow the self-timed pipeline for floating-point division to complete its 54 bit operation on every test vector pattern. In arithmetic circuits where nearly all faults are easily controllable and observable, this works well, but more general applications do get more benefit from adding scan paths into the pipeline latches.

Self-timed pipelines are data driven, and in some cases, this can be a power savings. If blocks end up switching less frequently than they would with a single global clock, then the self-timed aspect means that it consumes less power. If there is logic at the head of a pipeline that only propagates new data values into the pipeline when the outputs will be needed, then there can be fewer transitions than if every stage were transitioning to new data on every clock cycle. On the other hand, the isolation of downstream logic blocks from unnecessary transitions is a general power-saving technique, which is not unique to self-timed pipelines. Likewise, a potential power cost with the precharged logic from which self-timed pipelines are built is that dual-monotonic logic blocks require an output transition (which consumes energy) even if the data value is the same as a previous one. Although it is again not unique to self-timed pipelines, this issue can make precharged logic consume more power than purely complementary static logic that has only a 50% probability (or less) of needing to consume energy to switch outputs for each new data item. Depending on the timing of the precharge control signal and whether or not a series-evaluation device is instantiated, there may or may not be fighting (additional cross-over, or “crowbar” current) at the start of precharge. An additional advantage of the speed-independent styles of self-timed pipelines is that the control logic that ensures correct function for any delays also ensures that precharge is removed before evaluation, so this also lessens power consumption by guaranteeing no fighting. The bottom line for power consumption is that the issues are highly application dependent, and it would be misleading to overgeneralize as to which factor dominates.

## 9.8 CONCLUSION

Self-timed pipelines allow designers to control precharge logic blocks with more flexibility than just using a phase of the clock. It is also possible to avoid the synchronization points of clocked registers, and to enable combinational logic to stretch across multiple clock cycles without having to go through explicit latches at every cycle

boundary, thus avoiding both clock setup and skew penalties. For more generalized uses involving control logic or choices in dataflow, the potential advantages in average timing performance or power consumption vary dramatically depending on the particular application and on the sophistication of the design techniques used.

Self-timed rings are a generalization of self-timed pipelines that are applicable for iterative computations, such as arithmetic, which are functions that have significant internal computational load without needing intermediate control interaction. Realizing that a self-timed pipeline is a special case of a self-timed ring also unifies their analysis, and makes it clear why it is important to distinguish between the forward and reverse latency through stages. Carefully planning the “spacing” between launched data tokens is equivalent to controlling the number of bubbles in a fixed length of stages.

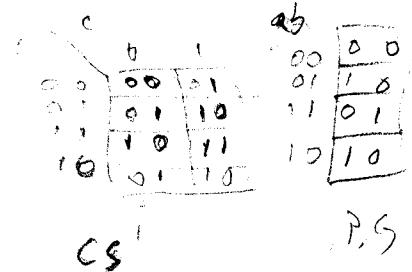
## REFERENCES

- [1] Robert Berks and Jo Ebergen, “Response Time Properties of Asynchronous Linear Pipelines,” *Proceedings of TAU97: ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Austin, Texas, pp. 179–188, Dec. 1997.
- [2] Erik Brunvand, Steven Nowick, and Kenneth Yun, “Practical Advances in Asynchronous Design and Asynchronous/Synchronous Interfaces,” *Proceedings of TAU99: ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Monterey, California, pp. 33–38, March 1999.
- [3] Terry and Barbara Chappell, et al., “A 2ns Cycle, 3.8ns Access 512kb CMOS ECL SRAM with a Fully Pipelined Architecture,” *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1577–1585, Nov. 1991.
- [4] Steve Furber, “Computing without Clocks: Micropipelining the ARM Processor,” in Graham Birtwistle and Al Davis, editors, *Asynchronous Digital Circuit Design*, Springer “Workshops in Computing” book series, pp. 211–262, 1995.
- [5] Jim Garside, “Amulet2e,” Hotchips Symposium Record, Stanford, California, pp. 257–274, Aug. 1996.
- [6] Jim Garside, Steve Furber, et al., “Amulet3 revealed,” *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Barcelona, Spain, April 1999.
- [7] Gary Gendel, et al., “Tools for Validating Asynchronous Digital Circuits,” *Proceedings of ASYNC94: International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Salt Lake City, Utah, pp. 12–21, Nov. 1994.
- [8] Craig Heikes, “A 4.5 mm<sup>2</sup> Multiplier Array for a 200MFLOP Pipelined Processor,” *ISSCC Digest of Technical Papers*, FA18.1, San Francisco, California, pp. 290–291, Feb. 1994.
- [9] Craig Heikes and Glenn Colon-Bonet, “A Dual Floating-Point Coprocessor with FMAC Architecture,” *ISSCC Digest of Technical Papers*, paper SP22.1, San Francisco, California, Feb. 1996, pp. 354.
- [10] Lawrence Heller G., et al., “Cascode Voltage Switch Logic: A Differential CMOS Logic Family,” *ISSCC Digest of Technical Papers*, San Francisco, California, 1984.
- [11] Ajay Khoche and Erik Brunvand, “Testing Micropipelines,” *Proceedings of ASYNC94: International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Salt Lake City, Utah, pp. 239–246, Nov. 1994.
- [12] Lavi Lev, et al., “A 64-b Microprocessor with Multimedia Support,” *IEEE Journal of Solid-State Circuits*, pp. 1227–1238, vol. 30, Nov. 1995.
- [13] Alain Martin, A. Lines, et al., “The design of an Asynchronous MIPS R3000 Microprocessor,” *Proceedings of Advanced Research in VLSI*, pp. 164–181, Sept. 1997.

- [14] David E. Muller, "Asynchronous logics and applications to information processing," *Proceedings of Symposium on Applications Switching Theory Space Technology*, pp. 289–297, 1963.
- [15] Steve Nowick, "Automatic synthesis of Burst-mode Asynchronous Controllers," Ph.D. Dissertation, Stanford University, *Computer Sci. Tech. Report CSL-TR-95-686*, 1995.
- [16] Steve Nowick, Mark Josephs, and Kees van Berkel, Special Issue on Asynchronous Circuits and Systems, *Proceedings of the IEEE*, vol. 87, no. 2, Feb. 1999.
- [17] Marly Roncken, "Defect-Oriented Testability for Asynchronous ICs," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 363–375, Feb. 1999.
- [18] Shai Rotem, Ken Stevens, et al., "CAD requirements for High Performance Asynchronous Circuits," *Proceedings of the 36th Design Automation Conference*, New Orleans, Louisiana, June 1999.
- [19] Mark Santoro and Mark Horowitz, "SPIM: A pipelined  $64 \times 64b$  Iterative Multiplier," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 487–493, April 1989.
- [20] Charles L. Seitz, "System Timing" in C. Mead and L. Conway, editors, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, Chap. 7, 1980.
- [21] Jakov N. Seizovic, "Pipeline Synchronization," *Proceedings of ASYNC94: International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Salt Lake City, Utah, pp. 87–96, Nov. 1994.
- [22] Ivan Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, pp. 720–738, 1989.
- [23] Ivan Sutherland, "The Counterflow Pipeline Processor," *Proceedings of ASYNC94: International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Salt Lake City, Utah, Nov. 1994.
- [24] H. Terada, S. Miyata, and M. Iwata, "Ddmpls: Self-timed super-pipelined data-driven multi-media processors," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 282–296, Feb. 1999.
- [25] Dennis Wendell, et. al., "A 3.5 ns,  $2K \times 9$  Self-timed SRAM," *ISSCC Digest of Technical Papers*, San Francisco, California, pp. 138–139, Feb. 1990.
- [26] Ted Williams, "Latency and Throughput Tradeoffs in Self-timed Asynchronous Pipelines and Rings," Stanford University, *Computer Sci. Tech Report CSL-TR-90-431*, Aug. 1990.
- [27] Ted Williams, "Self-timed Rings and their Application to Division," Ph.D. Dissertation, Stanford University, *Computer Sci. Tech. Report CSL-TR-91-482*, 1991.
- [28] Ted Williams and Mark Horowitz, "A Zero-Overhead Self-Timed 160 ns 54b CMOS Divider," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1651–1661, Nov. 1991.
- [29] Ted Williams, "Performance of Iterative Computation in Self-Timed Rings," *Kluwer Journal of VLSI Signal Processing*, vol. 7, pp. 17–31, Feb. 1994.
- [30] Ted Williams, "SPARC64: A 64b 64-Active-Instruction Out-of-Order-Execution MCM Processor," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 1215–1226, Nov. 1995.
- [31] Kenneth Yun, "A High-performance Asynchronous SCSI controller," *Proceedings of ICCD: International Conference on Computer Design*, Austin, Texas, pp. 44–49, Oct. 1995.

Vojin G Oklobdzija  
University of California

$$S_i =$$



## 10.1 INTRODUCTION

Digital computer arithmetic is an aspect of logic design with the objective of developing appropriate algorithms in order to achieve efficient utilization of the available hardware [1]–[4]. Because the hardware can only perform a relatively simple and primitive set of Boolean operations, arithmetic operations are based on a hierarchy of operations that are built upon the simple ones. Since ultimately, speed, power, and chip area are the most often-used measures of the efficiency of an algorithm, there is a strong link between the algorithms and technology used for its implementation.

## 10.2 HIGH-SPEED ADDITION: ALGORITHMS AND VLSI IMPLEMENTATION

First we will examine a realization of a one-bit adder, which represents a basic building block for all the more elaborate addition schemes.

### 10.2.1 Full Adder

Operation of a full adder is defined by the Boolean equations for the sum and carry signals:

$$\begin{aligned} s_i &= a_i \bar{b}_i \bar{c}_i + \bar{a}_i b_i \bar{c}_i + \bar{a}_i \bar{b}_i c_i + a_i b_i c_i = a_i \oplus b_i \oplus c_i \\ c_{i+1} &= \bar{a}_i b_i c_i + a_i \bar{b}_i c_i + a_i b_i \bar{c}_i + a_i b_i c_i \end{aligned}$$

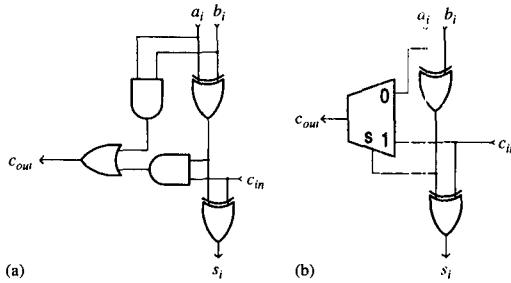
where  $a_i$ ,  $b_i$ , and  $c_i$  are the inputs to the  $i$ th full adder stage, and  $s_i$  and  $c_{i+1}$  are the sum and carry outputs from the  $i$ th stage, respectively. From the above equation we realize that the realization of the sum function requires two XOR logic gates.

The carry function is further rewritten defining the carry-propagate  $p_i$  and carry-generate  $g_i$  terms:

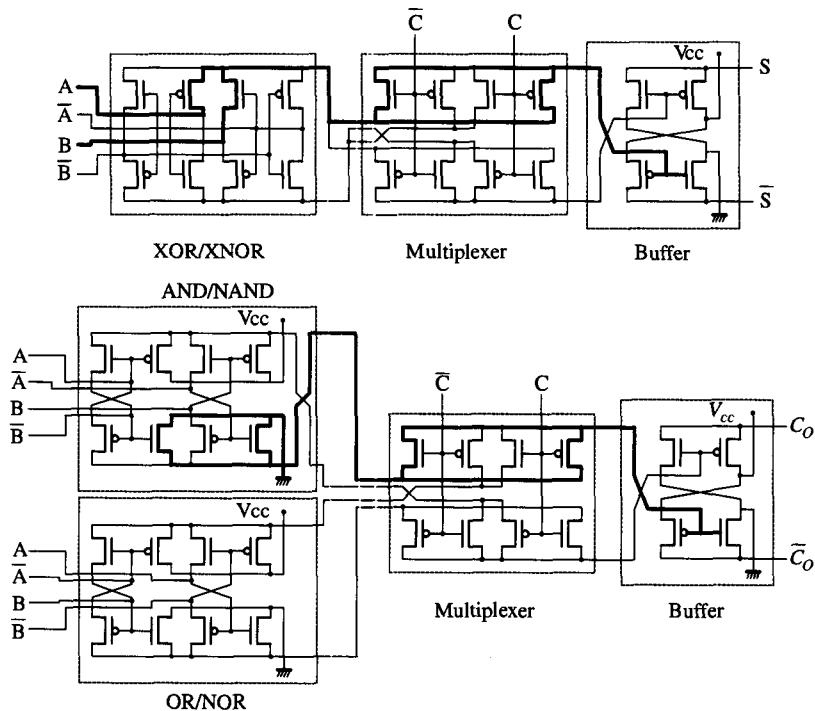
$$p_i = a_i \oplus b_i, \quad g_i = a_i \bullet b_i$$

At a given stage  $i$ , a carry is generated if  $g_i$  is true (i.e., both  $a_i$  and  $b_i$  are ONEs), and if  $p_i$  is true, a stage propagates an input carry to its output (i.e., either  $a_i$  or  $b_i$  is a ONE). The logical implementation of the full adder is shown in Fig. 10.1(a). For this implementation, the delay from either  $a_i$  or  $b_i$  to  $s_i$  is two XOR delays and the delay from  $c_i$

to  $c_{i+1}$  is two gate delays. Some technologies, such as CMOS, implement the functions more efficiently by using pass-transistor circuits. For example, the critical path of the carry-in to *carry-out* uses a fast pass-transistor multiplexer [8] in an alternative implementation of the full adder shown in Fig. 10.1(b).



**Figure 10.1** Full-adder implementation: (a) regular; (b) using multiplexer in the critical path.

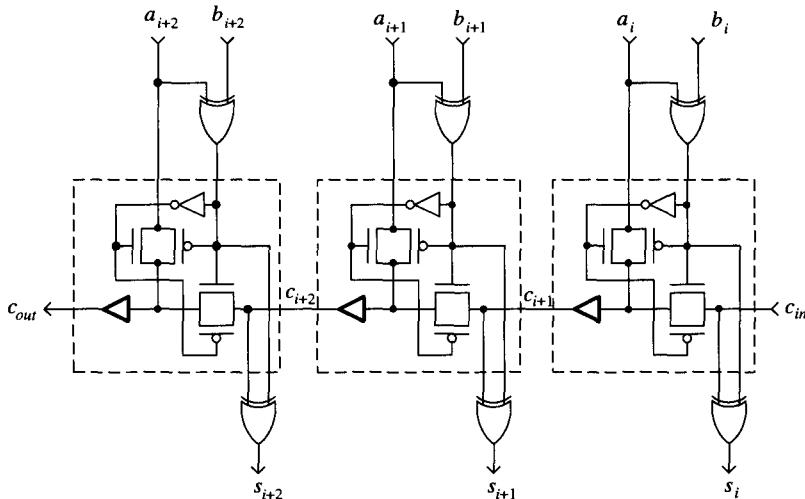


**Figure 10.2** Pass-transistor realization of a full adder in DPL [11].

The ability of pass-transistor logic to provide an efficient multiplexer implementation has been exploited in CPL and DPL logic families [10], [11]. Even an XOR gate is more efficiently implemented using multiplexer topology. A full-adder cell, which is entirely multiplexer based as published by Hitachi [11], is shown in Fig. 10.2. Such a full-adder realization contains only two transistors in the *Input-to-Sum* path and only one transistor in the  $C_{in}$ -to- $C_{out}$  path (not counting the buffer). The short critical path is a factor that contributes to a remarkable speed of this implementation.

### 10.2.2 Ripple Carry Adder

A ripple carry adder for  $N$ -bit numbers is implemented by concatenating  $N$  full adders as shown on Fig. 10.3. At the  $i$ th bit position, the  $i$ th bits of operands A and B and a carry signal from the preceding adder stage are used to generate the  $i$ th bit of the sum,  $s_i$ , and a carry,  $c_{i+1}$ , to the next adder stage. This is called a *ripple carry adder* (RCA), since the carry signal “ripple” from the least significant bit position to the most significant [3]–[4]. If the ripple carry adder is implemented by concatenating  $N$  full adders, the delay of such an adder is  $2N$  gate delays from  $C_{in}$ -to- $C_{out}$ .



**Figure 10.3** Carry-chain of an RCA implemented using multiplexer from the standard cell library [8].

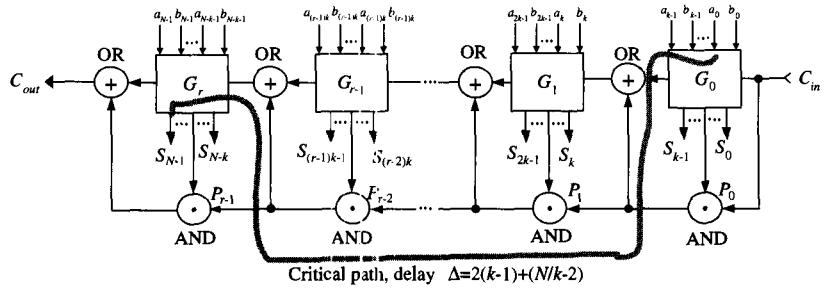
The path from the input to the output signal that is likely to take the longest time is designated as a “*critical path*.” In the case of an RCA, this is the path from the least significant input  $a_0$  or  $b_0$  to the last sum bit  $s_n$ . Assuming a multiplexer-based XOR gate implementation, this critical path will consist of  $N + 1$  pass transistor delays. However, such a long chain of transistors will significantly degrade the signal, thus some amplification points are necessary. In practice, we can use a multiplexer cell to build this critical path using the standard cell library as shown in Fig. 10.3 [8].

### 10.2.3 Carry Skip Adder

Since the  $C_{in}$ -to- $C_{out}$  represents the longest path in the ripple carry adder, an obvious attempt is to accelerate carry propagation through the adder. This is accomplished by using carry-propagate  $p_i$  signals within a group of bits. If all the  $p_i$  signals within the group are  $p_i = 1$ , the conditions exist for the carry to bypass the entire group:

$$P = p_i \bullet p_{i+1} \bullet p_{i+2} \bullet \dots \bullet p_{i+k} - 1$$

The *carry skip adder* (CSKA) divides the words to be added into groups of equal size of  $k$  bits. The basic structure of an  $N$ -bit carry skip adder is shown in Fig. 10.4. Within the group, carry propagates in a ripple-carry fashion. In addition, an AND gate is used to



**Figure 10.4** The basic structure of a CSKA:  $N$  bits,  $k$  bits per group,  $r = N/k$  groups.

form the group propagate signal  $P$ . If  $P = 1$  the condition exists for carry to bypass (skip) over the group as shown in Fig. 10.4.

The maximal delay of a carry skip adder is encountered when carry signal is generated in the least-significant bit position, rippling through  $k - 1$  bit positions, skipping over  $N/k - 2$  groups in the middle, rippling to the  $k - 1$  bits of the most significant group and being assimilated in the  $N$ th-bit position to produce the sum  $S_N$ :

$$\begin{aligned}\Delta_{CSA} &= (k-1)\Delta_{rca} + \left(\frac{N}{k}-2\right)\Delta_{SKIP} + (k-1)\Delta_{rca} \\ &= 2(k-1)\Delta_{rca} + \left(\frac{N}{k}-2\right)\Delta_{SKIP}\end{aligned}$$

Thus, CSKA is faster than RCA at the expense of a few relatively simple modifications. The delay is still linearly dependent on the size of the adder  $N$ ; however, this linear dependence is reduced by a factor of  $1/k$  [3].

#### 10.2.4 Variable Block Adder

The idea behind *variable block adder* (VBA) is to minimize the longest critical path in the carry chain of a CSKA, while allowing the groups to take different sizes [2], [7]. Such an optimization in general does not result in an enhanced complexity as compared to the CSKA. A carry chain of a 32-bit VBA is shown in Fig. 10.5.

The first and last blocks are smaller, and the intermediate blocks are larger. That compensates for the critical paths originating from the ends by shortening the length of the path used for the carry signal to ripple in the end groups, allowing carry to skip over larger groups in the middle.

There are two important consequences of this optimization:

1. First, the total delay is reduced as compared to CSKA.
2. Second, the delay dependency is not a linear function of the adder size  $N$  as in CSKA. This dependency follows a square root function of  $N$  instead.

For an optimized VBA, it is possible to obtain a closed form solution expressing this delay dependency, which is given as

$$\Delta_{VBA} = c_1 + \sqrt{c_2 N + c_3}$$

where  $c_1, c_2, c_3$  are constants. It is also possible to extend this approach to multiple levels of carry skip as done in [7]. A determination of the optimal sizes of the blocks on

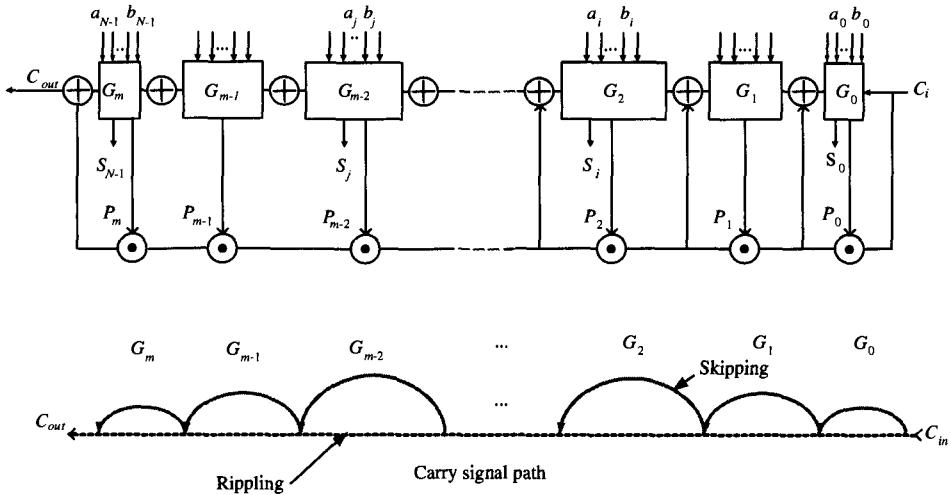


Figure 10.5 Carry chain of a 32-bit variable block adder.

the first and higher levels of skip blocks is a linear programming problem which does not yield a closed form solution. Such types of problems are solved with the use of dynamic programming techniques. The speed of such a multiple-level VBA adder surpasses single-level VBA and that of fixed group *carry lookahead adder* (CLA) [15]. There are two reasons why this is possible:

1. First, the speed of the logic gates used for CMOS implementation depends on the output load *fan-out*, as well as the number of inputs *fan-in*. CLA implementation is characterized with the large *fan-in* which limits the available size of the groups. On the other hand VBA implementation is simple. Thus, it seems that CLA has passed the point of diminishing returns as far as an efficient implementation is concerned. This example also points to the importance of modeling and incorporating appropriate technology parameters into the algorithm. Most of the computer arithmetic algorithms developed in the past use a simple *constant gate delay* model.
2. Second, a fixed-group CLA is not the best way to build an adder. It is a sub-optimal structure that after being optimized for speed, consists of groups that are different in size yielding a largely irregular structure [15].

There are other advantages of VBA adder that are a direct result of its simplicity and efficient optimization of the critical path. Those advantages are exhibited in the lower area and power consumption while retaining its speed. Thus, VBA has the lowest energy-delay product as compared to the other adders in its class [9].

### 10.2.5 Carry Lookahead Adder

A significant speed improvement in the implementation of a parallel adder was introduced by a CLA adder developed by Weinberger and Smith in 1958 [13]. The CLA adder is theoretically one of the fastest schemes used for the addition of two numbers, since the delay to add two numbers depends on the logarithm of the size of the operands:

$$\Delta \approx \log[N]$$

The CLA adder uses modified full adders (modified in the sense that a carry output is not formed) for each bit position and lookahead modules which are used to generate carry signals independently for a group of  $k$  bits. In the most common case  $k = 4$ . In addition to carry signal for the group, lookahead modules produce group carry-generate (G) and group carry-propagate (P) outputs that indicate that a carry is generated within the group, or that an incoming carry would propagate across the group.

Extending the carry equation to a second stage in a ripple carry adder we obtain

$$\begin{aligned} c_{i+2} &= g_{i+1} + p_{i+1}c_{i+1} \\ &= g_{i+1} + p_{i+1}(g_i + p_ic_i) \\ &= g_{i+1} + p_{i+1}g_i + p_{i+1}p_ic_i \end{aligned}$$

This carry equation results from evaluating the carry equation for the  $i + 1$ -st stage and substituting  $c_{i+1}$ . Carry  $c_{i+2}$  exits from stage  $i + 1$  if

- a. a carry is generated in the stage  $i + 1$  or
- b. a carry is generated in stage  $i$  and propagates across stage  $i + 1$  or
- c. a carry enters stage  $i$  and propagates across both stages  $i$  and  $i + 1$ , etc.

Extending the carry equation to a third stage yields

$$\begin{aligned} c_{i+3} &= g_{i+2} + p_{i+2}c_{i+2} \\ &= g_{i+2} + p_{i+2}(g_{i+1} + p_{i+1}g_i + p_{i+1}p_ic_i) \\ &= g_{i+2} + p_{i+2}g_{i+1} + p_{i+2}p_{i+1}g_i + p_{i+2}p_{i+1}p_ic_i \end{aligned}$$

Although it would be possible to continue this process indefinitely, each additional stage increases the size (i.e., the number of inputs) of the logic gates. Four inputs (as required to implement the  $c_{i+3}$  equation) is frequently the maximum number of inputs per gate for current technologies. To continue the process, the carry lookahead scheme utilizes group generate and group propagate signals over four-bit groups (stages  $i$  to  $i + 3$ ),  $G_j$  and  $P_j$ , respectively:

$$G_j = g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_{i+1}g_i$$

and

$$P_j = p_{i+3}p_{i+2}p_{i+1}p_i$$

The carry equation can be expressed in terms of the four-bit group generate and propagate signals:

$$c_{i+j} = G_j + P_j c_i$$

Thus, the carry-out from a four-bit wide group  $c_{i+4}$  can be computed in four gate delays: one gate delay to compute  $p_i$  and  $g_i$  for  $i = i$  through  $i + 3$ , a second gate delay to evaluate  $P_j$ , the second and the third to evaluate  $G_j$ , and the third and fourth to calculate carry signals  $c_{i+1}$ ,  $c_{i+2}$ ,  $c_{i+3}$ , and  $c_{i+4}$ . Actually, if not limited by fan-in constraints,  $c_{i+4}$  could be calculated concurrently with  $G_j$  and will be available after three gate delays.

In general, a  $k$ -bit lookahead group requires  $0.5(3k + k^2)$  logic gates, where  $k$  is the size of the group. In a recursive fashion, we can create a “group of groups” or a “super-group.” The inputs to the “supergroup” are  $G$  and  $P$  signals from the previous level. The “supergroup” produces  $P^*$  and  $G^*$  signals indicating that the carry signal will be propagated across all of the groups within the “supergroup” domain, or that the carry will be generated in one of the groups encompassed by the “supergroup.” Similarly to the group, a “supergroup” produces a carry signal out of the “supergroup” as well as an

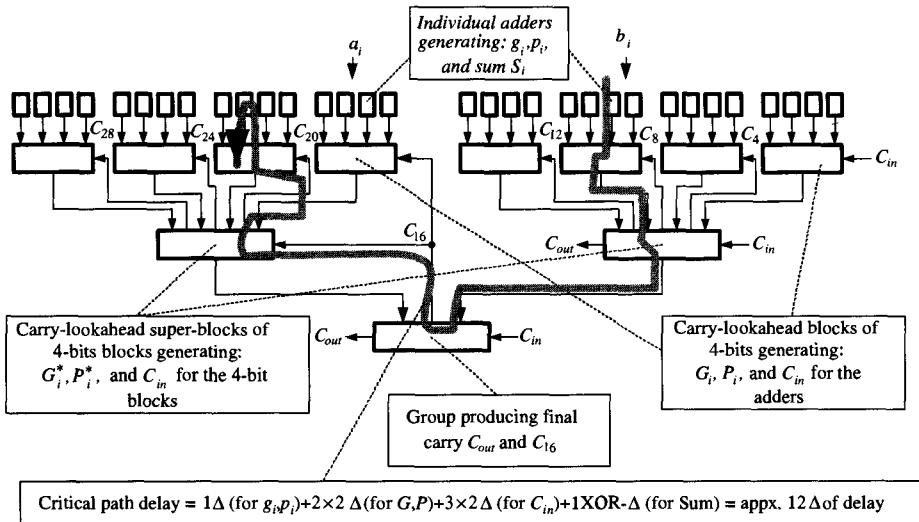


Figure 10.6 A 32-bit carry lookahead adder.

input carry signal for each group in the level above:

$$\begin{aligned} G_k^* &= G_{j+3} + P_{j+3}G_{j+2} + P_{j+3}P_{j+2}G_{j+1} + P_{j+3}P_{j+2}P_{j+1}G_j \\ P_k^* &= P_{j+3}P_{j+2}P_{j+1}P_j \\ C_4(j+1) &= G_j + P_jC_4j \end{aligned}$$

A construction of a 32-bit carry lookahead adder is illustrated in Fig. 10.6.

As opposed to RCA or CSA, the critical path in the CLA travels in a vertical direction rather than a horizontal one as shown in Fig. 10.6. Therefore the delay of CLA is not directly proportional to the size of the adder  $N$ , but to the number of levels used. Because the groups and supergroups in the CLA resemble a tree structure the delay of a CLA is thus proportional to the  $\log$  function of the size  $N$ .

CLA delay is evaluated by recognizing that an adder with a single level of carry lookahead (four-bit words) contains three gate delays in the carry path. Each additional level of lookahead increases the maximum word size by a factor of  $k$  and adds two additional gate delays. Generally the number of lookahead levels for an  $N$ -bit adder is  $\lceil \log_k N \rceil$  where  $k+1$  is the maximum number of inputs per gate. Since a  $k$ -bit group carry lookahead adder introduces three gate delays per CLA level, and there are two additional gate delays: one for  $g_i$  and  $p_i$ , and the other for the final sum  $s_i$ , CLA delay  $\Delta$  is

$$\Delta_{CLA} = 1 + 2(\lceil \log_k N \rceil - 1) + 1 = 2 \log_k \lceil N \rceil$$

This  $\log$  dependency makes CLA one of the theoretically fastest structures for addition [2]–[4]. However, it can be argued that the speed efficiency of the CLA has passed the point of diminishing returns given the *fan-in* and *fan-out* dependencies of the logic gates and inadequacy of the delay model based on counting the number of gates in the critical path. In reality, CLA is indeed achieving lesser speed than expected, especially when compared to some techniques that consume less hardware for the implementation as shown in [7], [8].

One of the simple schemes for addition that was very popular at the time when transition into MOS technology was made, is *Manchester carry chain* (MCC) [6], [38].

MCC is an alternative switch-based technique implemented using pass-transistor logic. The speed realized using MCC is impressive because of its simplicity and the properties of the pass-transistor logic. MCC does not require a large area for its implementation, consuming substantially less power as compared to CLA or other more elaborate schemes. A realization of the MCC is shown in Fig. 10.7. Due to the RC delay properties of the MCC the signal needs to be regenerated by inserting inverters at appropriately chosen locations in the carry chain.

In the same way a CLA can be built using MCC for the implementation of the lookahead group. Further, pass-transistor MCC structure can be incorporated in the logic group of the circuit technology used for CLA realization. One such example is shown in Fig. 10.8(a) representing a four-bit group of a 64-bit CLA built using CMOS

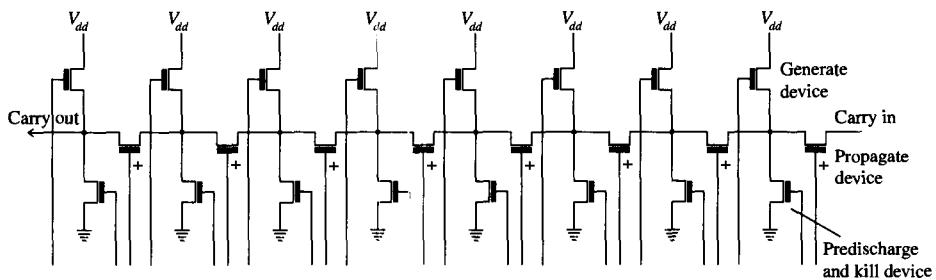


Figure 10.7 Manchester carry chain realization of the carry path.

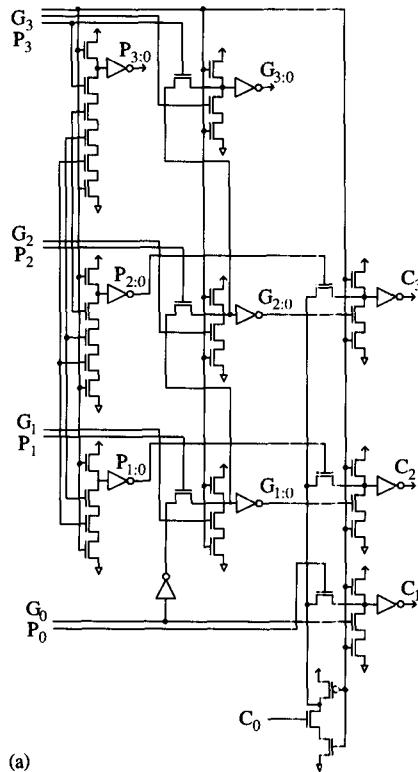
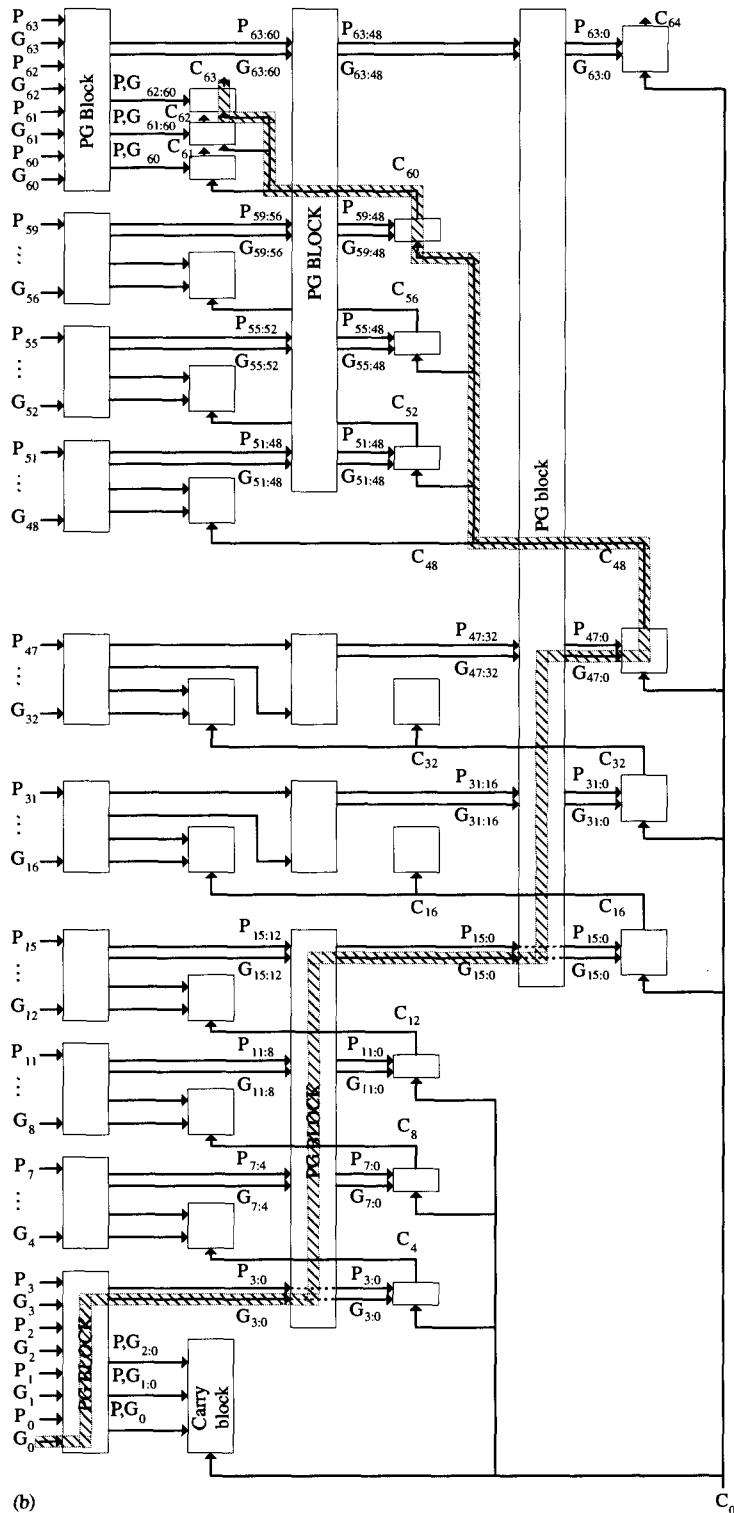


Figure 10.8 (a) CMOS Domino realization of a 64-bit CLA. (b) Critical path in Motorola's 64-bit CLA [14].



Critical path: A, B -  $G_0 \cdot G_{15:0} \cdot G_{47:0} \cdot G_{31:16} \cdot G_{15:0} \cdot G_{15:0} \cdot G_{11:0} \cdot G_{11:0} \cdot G_{7:4} \cdot G_{7:4} \cdot G_{3:0} \cdot G_{3:0} \cdot S_{63}$

Figure 10.8 (Cont.)

Domino logic [14]. Each CLA group is implemented as a separate CMOS Domino function. This adder built by Motorola using  $1.0\text{ }\mu\text{m}$  CMOS technology achieved a remarkable speed of  $4.5\text{ nS}$  at  $V_{DD} = 5\text{ V}$  and  $25^\circ\text{C}$ . The critical path of this design is shown in Fig. 10.8(b). Using selection technique and careful analysis of the critical path the same adder was extended to 96 bits at the same speed of  $4.5\text{ nS}$ . As with RCA, the carry lookahead adder complexity grows linearly with the word size (for  $k = 4$ , this occurs at a 40% faster rate than the RCA).

### 10.2.6 Recurrence Solver-Based Adders

The class of adders based on solving recurrence equations were first introduced by Bilgory and Gajski [17] and Brent and Kung [18] and were based on the previous work by Kogge and Stone [16]. They realized that if  $C_{in} = 0$  can be assumed, the carry lookahead equations can be written in a simple form of a recurrence:

$$(g, p) \bullet (g, p) = (g + pg', pp')$$

where an operator  $\bullet$  termed “black” operator was introduced. By application of this recurrence equation various topologies of an adder can be obtained with three desirable properties:

1. A good layout.
2. The fan-out can be controlled and limited to no more than two.
3. Trade-offs between *fan-in*, *fan-out* and hierarchical layout topologies can be achieved.

The above properties were the cause for the relative popularity of the “recurrence-solver” schemes. In essence, “recurrence solver” based adders are simply a variation of many possible different CLA topologies [2]. An example of a “recurrence solver” adder is shown in Fig. 10.9.

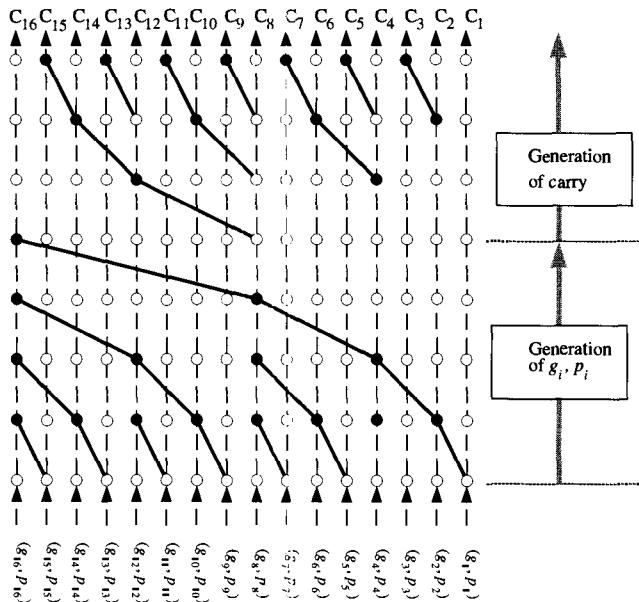


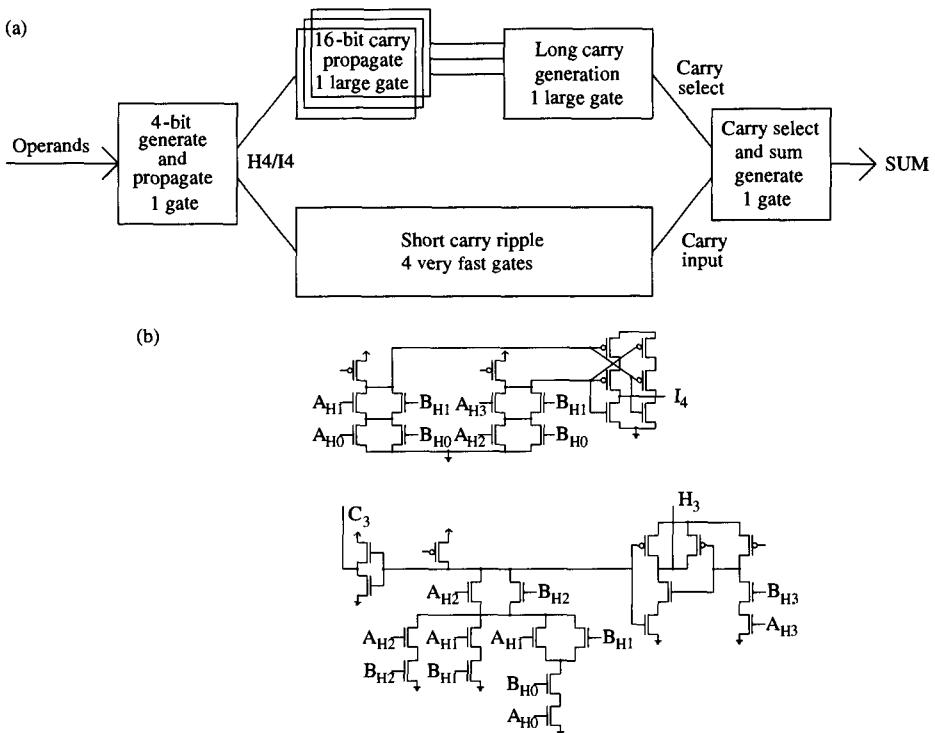
Figure 10.9 Recurrence solver adder topology.

### 10.2.7 Ling Adder

Ling adder is a scheme developed at IBM to take advantage of the ability of the ECL technology to perform wired-OR operation with a minimal additional delay [19]. Ling redefined the equations for sum and carry by encoding pairs of digit positions:  $(a_i, b_i, a_{i-1}, b_{i-1})$ . To understand the advantage of Ling adder, we will consider the generation of  $C_3$  carry-out bit using conventional CLA and using modified Ling equations. Without the wired-OR function,  $C_3$  can be implemented in three gate delays. The expansion of those equations will yield 15 terms and a maximum fan-in of 5. Ling equations on the other hand will perform the same evaluation (of Ling's modified carry  $H_3$ ) using eight terms with the maximal fan-in of 4. Thus, in a particular IBM's ECL technology (for which this adder was developed) with the limitation of fan-in of 4 for the wired-OR term, Ling's adder yields substantial advantage. Thus Ling adder can realize a sum delay in

$$\Delta = \left\lceil \log_r \frac{N}{2} \right\rceil + 1$$

The Ling adder was also found to be adequate for realizations using CMOS technology. The advantage of high-gain and fan-in capabilities of dynamic CMOS combined with the dual rail DCVS logic were used in Hewlett-Packard's sub-nanosecond adder which was design in 0.5  $\mu\text{m}$  CMOS technology [20]. The organization of this adder is shown in Fig. 10.10(a) while the circuits used for generation of  $H_4$  and  $I_4$  terms are shown in Fig. 10.10(b).



**Figure 10.10** (a) Organization of a 64-bit Ling adder realized in CMOS technology [20]. (b) Circuit used for generation of  $H_4$  and  $I_4$  terms [20].

### 10.2.8 Conditional-Sum Addition

The theoretically fastest scheme for addition of two numbers is *conditional-sum addition* (CNSA) proposed by Sklanski in 1960 [3], [5], [21]. The essence of the CNSA scheme is in the realization that we can add two numbers without waiting for the carry signal to be available. Simply, the numbers are added in two instances: one assuming  $C_{in} = 0$  and the other assuming  $C_{in} = 1$ . The conditionally produced results:  $Sum_0$ ,  $Sum_1$  and  $Carry_0$ ,  $Carry_1$  are selected by a multiplexer using an incoming carry signal  $C_{in}$  as a multiplexer control. Similarly to the CLA the input bits are divided into groups that are in the case of CNSA added “conditionally.”

It is apparent that while building CNSA the hardware complexity starts to grow rapidly starting from the *least significant bit* (LSB) position. Therefore, in practice, the full-blown implementation of the CNSA is not found. However, the idea of adding the *most significant* (MS) portion of the operands conditionally and selecting the results once the carry-in signal is computed in the *least significant* (LS) portion is attractive. Such a scheme (which is a subset of CNSA) is known as “*carry select adder*” (CSLA) [22].

### 10.2.9 Carry Select Adder

The *carry select adder* (CSLA) divides the words to be added into blocks and forms two sums for each block in parallel (one with a carry in of ZERO and the other with a carry in of ONE) [2], [3], [5], [22]. As shown in an example of a 16-bit carry select adder in Fig. 10.11, the carry-out from the previous LS four-bit block controls a multiplexer that selects the appropriate sum from the MS portion. The carry-out is computed using the equation for carry-out of the group, since the group-propagate signal  $P_i$  is the carry-out of an adder with a carry-input of ONE and the group-generate  $G_i$  signal is the carry-out of an adder with a carry-input of ZERO. This speeds-up the computation of the carry signal necessary for selection in the next block. The upper 8 bits are computed conditionally using two CSLAs similar to the one used in the LS eight-bit portion. The delay of this adder is determined by the speed of the LS  $k$ -bit block (four-bit RCA in the example, Fig. 10.11) and delay of multiplexers in the MS path. Generally this delay is

$$\Delta = \delta_{MUX} \lceil \log_k N \rceil + \delta_{k\text{-bit-adder}}$$

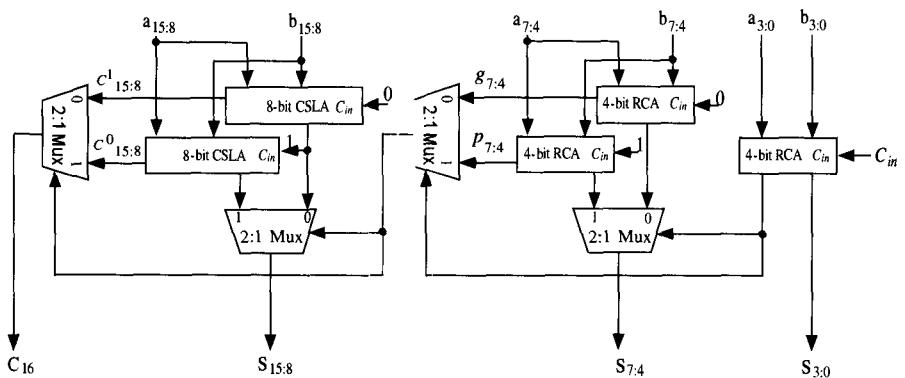


Figure 10.11 Sixteen-bit of a carry select adder.

### 10.2.10 DEC “Alpha” 21064 Adder

The 64-bit adder used in the first 200 MHz Digital’s Alpha 21064 RISC microprocessor employed a combination of techniques in order to reach 5 nS cycle required from the 0.75  $\mu\text{m}$  CMOS technology of implementation [23]. There were four different techniques used on the various levels of this 64-bit adder:

1. On the eight-bit level the Manchester carry chain technique was used. MCC seems to be the most effective for the short adders, especially when the word length is below 16 bits. The carry chain was further optimized by tapering down each chain stage in order to reduce the load caused by the remainder of the chain. The chain was predischarged at the beginning of the operation, and three signals were used: propagate P, generate G, and carry-kill (assimilate) K. The local carry signals were amplified using ratioed inverters. There were two MCC employed: one that assumes  $C_{in} = 0$  and other that assumes  $C_{in} = 1$ .
2. Carry-lookahead addition (CLA) was used on the least significant 32 bits of the adder. The CLA section was implemented as a distributed differential circuit producing the carry signal that controls the most significant 32-bit portion of the adder.
3. Conditional sum addition (CSA) was used for the most significant 32 bits of the adder. There were six eight-bit select switches used to implement conditional summation on the eight-bit level.
4. Finally, the carry select (CSLA) method was used to produce the most significant 32 bits of the 64-bit word. The selection of the final result was done using NMOS byte carry-select switches.

The block diagram of DEC “Alpha’s” adder is shown in Fig. 10.12 [23].

## 10.3 MULTIPLICATION

### 10.3.1 Algorithm

In microprocessors multiplication operation is performed in a variety of forms in hardware and software depending on the cost and transistor budget allocated for this particular operation. In the beginning stages of computer development any complex operation was usually programmed in software or coded in the micro-code of the machine. Some limited hardware assistance was provided. Today it is more likely to find full hardware implementation of the multiplication in order to satisfy growing demand for speed and due to the decreasing cost of hardware [2]–[5]. For simplicity, we will describe a basic multiplication algorithm which operates on positive  $n$ -bit long integers  $X$  and  $Y$  resulting in the product  $P$  which is  $2n$  bit long:

$$P = XY = X \times \sum_{i=0}^{n-1} y^i r^i = \sum_{i=0}^{n-1} X \times y^i r^i$$

This expression indicates that the multiplication process is performed by summing  $n$  terms of a *partial product*  $P_i$ . This product indicates that the  $i$ th term  $P_i$  is obtained by

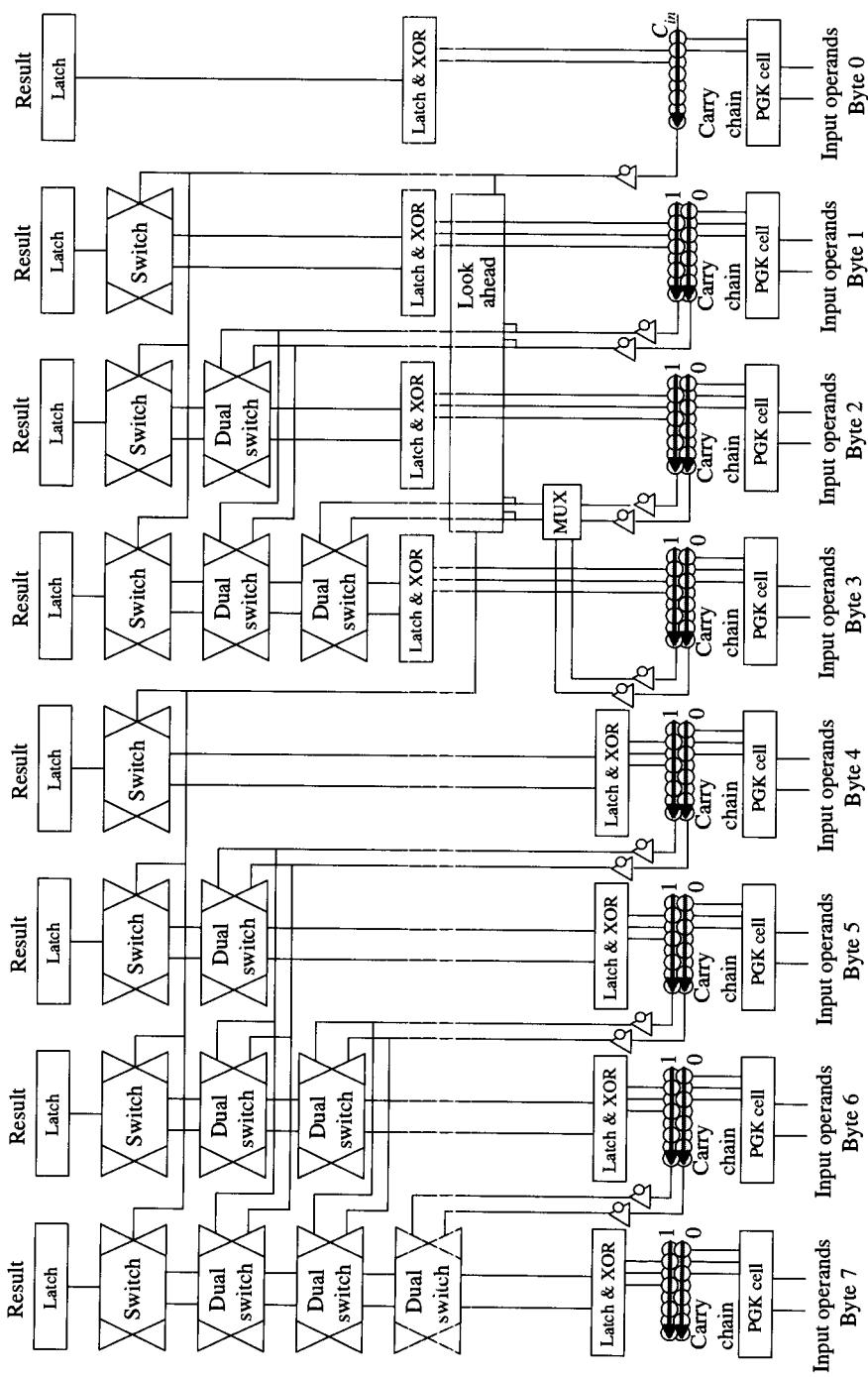


Figure 10.12 Block diagram of DEC “Alpha” 64-bit adder [23].

simple arithmetic left shift of  $X$  for the  $i$  positions and multiplication by the single digit  $y_i$ . For the binary radix ( $r = 2$ ),  $y_i$  is 0 or 1 and multiplication by the digit  $y_i$  is very simple to perform. The addition of  $n$  terms can be performed at once, by passing the *partial products* through a network of adders or sequentially, by adding *partial products* using an adder  $n$  times. The algorithm to perform the multiplication of  $X$  and  $Y$  can be described as [5]:

$$\begin{aligned} p^{(0)} &= 0 \\ p^{(j+1)} &= \frac{1}{r}(p^j + r^n X y_j) \quad \text{for } j = 0, \dots, n-1 \end{aligned}$$

It can be easily proved that this recurrence results in  $p^{(n)} = XY$ .

### 10.3.2 High-Performance Multipliers

The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past, multiplication was generally implemented via a sequence of addition, subtraction, and shift operations.

#### 10.3.2.1 Parallel Multipliers

An alternative approach to sequential multiplication involves the combinational generation of all bit products and their summation with an array of full adders. This approach uses an  $n \times n$  array of AND gates to form the bit products, an array of  $n \times n$  adders (and half adders) to sum the  $n^2$  bit products in a carry-save fashion. Finally a  $2n$  CPA is used in the final step to finish the summation and produce the result [2]–[5].

#### 10.3.2.2 Wallace/Dadda Multiplier

In his historic paper C. S. Wallace introduced a way of summing the partial product bits in parallel using a tree of carry-save adders which became generally known as the “Wallace Tree” [2], [25]. This method was further refined by Dadda [26].

With the Wallace method, a three-step process is used to multiply two numbers:

1. The bit products are formed.
2. The bit product matrix is “reduced” to a two-row matrix by using a carry-save adder (known as the Wallace Tree).
3. The remaining two rows are summed using a fast carry-propagate adder to produce the product.

Although this may seem complex, it yields multipliers with delay proportional to the logarithm of the operand size  $n$ .

A suggestion for improved efficiency of addition of the partial product was published by Dadda [26]. In his historic 1965 paper, Dadda introduces a notion of a *counter* structure which will take a number of bits  $p$  in the same bit position (of the same “weight”) and output a number  $q$  which represents the count of ones at the input.

Dadda has introduced a number of ways to compress the partial product bits using such a counter, which later became known as “*Dadda’s counter*.”

This process is shown for an  $8 \times 8$  *Dadda multiplier* in Fig. 10.13 [2]. An input  $8 \times 8$  matrix of dots (each dot represents a bit product) is shown as matrix 0. Columns having more than six dots (or that will grow to more than six dots due to carries) are reduced by the use of half-adders (each half-adder takes in two dots and outputs one in the same column and one in the next more significant column) and full adders (each full adder takes in three dots and outputs one in the same column and one in the next more significant column) so that no column in matrix 1 will have more than six dots. Half-adders are shown by a “crossed” line in the succeeding matrix, and full adders are shown by a line in the succeeding matrix. In each case the right-most dot of the pair that are connected by a line is in the column from which the inputs were taken for the adder. In the succeeding steps reduction to matrix 2 with no more than four dots per column, matrix 3 with no more than three dots per column, and finally matrix 4 with no more than two dots per column is performed. The height of the matrices is determined by working back from the final (two-row) matrix and limiting the height of each matrix to the largest integer that is no more than 1.5 times the height of its successor. Each matrix is produced from its predecessor in one adder delay. Since the number of matrices is logarithmically related to the number of bits in the words to be multiplied, the delay of the matrix reduction process is proportional to  $\log(n)$ . Since the adder that reduces the final two-row matrix can be implemented as a carry lookahead adder (which also has logarithmic delay), the total delay for this multiplier is proportional to the logarithm of the word size [2], [4].

An extensive study of the use of “*Dadda’s counters*” was undertaken by Stenzel and Kubitz in 1977. In their paper [27] they have also demonstrated a parallel multiplier built using ROM to implement [5], [4] counters used for partial product summation.

The quest for making the parallel multiplier even faster continued for almost 30 years. However, the pursuit for inventing a fastest “*counter*” did not result in a structure yielding faster partial product summation than the one that uses *full-adder* (FA) cell, or “*3:2 counter*.” Therefore the Wallace Tree was widely used in the implementation of the parallel multipliers.

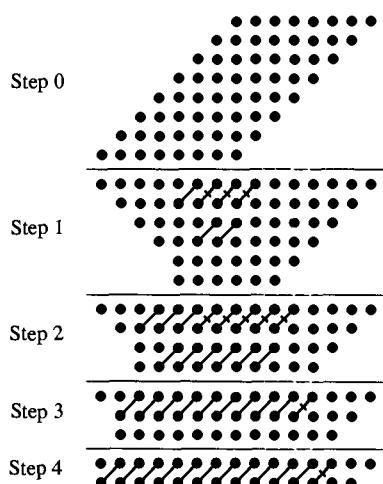


Figure 10.13 An  $8 \times 8$  Dadda multiplier example [26].

### 10.3.2.3 4:2 Compressor

In 1981 Weinberger disclosed a structure that he called “4-2 carry-save module” [28]. This structure contained a combination of FA cells in an intricate interconnection structure that was yielding faster partial product compression than the use of 3:2 counters.

The structure actually compresses five partial product bits into three; however, it is connected in such a way that four of the inputs are coming from the same bit position of the weight  $j$  while one bit is fed from the neighboring position  $j - 1$  (known as carry-in). The output of such a 4:2 module consists of one bit in the position  $j$  and two bits in the position  $j + 1$ . This structure does not represent a counter (though it became erroneously known as “4:2 counter”) but a “compressor” that would compress four partial product bits into two (while using one bit laterally connected between adjacent 4:2 compressors). The structure of a 4:2 compressor is shown in Fig. 10.14. The efficiency of such a structure is higher (it reduces the number of partial product bits by one half at each stage). The speed of such a 4:2 compressor has been determined by the speed of 3 XOR gates in series, in the redesigned version of 4:2 compressor [36], making such a scheme more efficient than the one using 3:2 counters in a regular Wallace Tree. The other equally important feature of the use of 4:2 compressor is that the interconnections between 4:2 cells follow a more regular pattern than in case of the Wallace Tree.

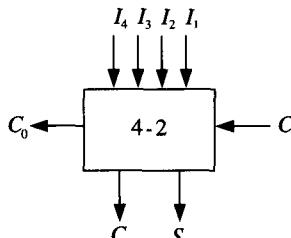


Figure 10.14 A 4:2 compressor arrangement [28].

### 10.3.2.4 TDM

The further work in improving the speed of a multiplier by optimizing the *partial product reduction tree* (PPRT) was extended by Oklobdzija, Villeger, and Liu [30]. Their approach was to optimize the entire PPRT in one pass; thus the name *three-dimensional optimization method* (TDM). The important aspect of this method is in the sorting of fast inputs and fast outputs. It was realized that the most important step is to properly interconnect the elements used in the PPRT. Thus, appropriate counters (3:2 adders in a particular case) were characterized in a way that identifies delay of each input to each output. Interconnecting of the PPRT was done in a way in which signals with large delays are connected to “*fast inputs*” and signals with small delays are connected to “*slow inputs*” in a way that minimizes the critical paths in the PPRT. An example of this method is illustrated in Fig. 10.15, producing a 3 XOR gate delay 4:2 compressor, without resorting to a redesign as done in [36]. It was further proven that TDM indeed produces an optimal PPRT and that further optimization is not possible [37], [30]. An example of TDM generation of PPRT is shown in Fig. 10.16.

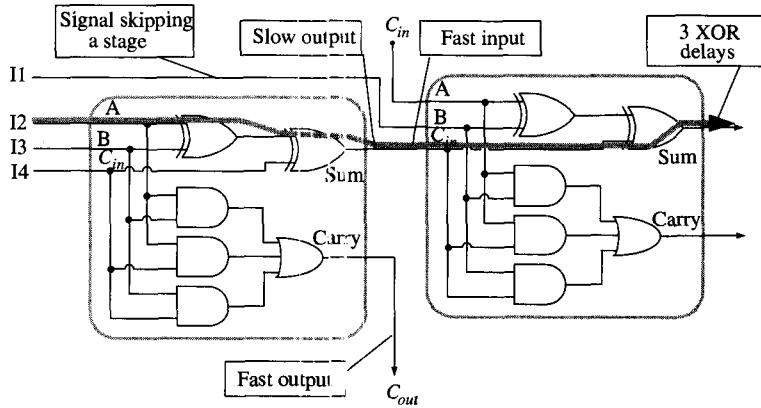


Figure 10.15 An example of TDM method producing a balanced 4:2 compressor [30].

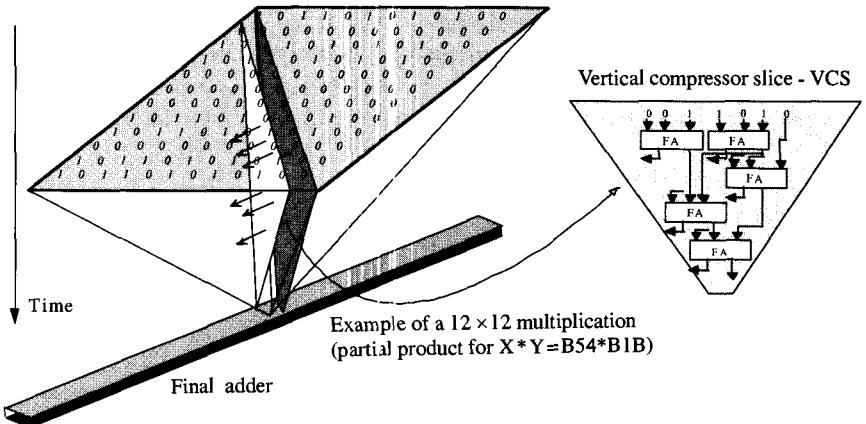


Figure 10.16 Generation of the partial product reduction tree in TDM multiplier [30].

### 10.3.2.5 Booth Recoding Algorithm

One of the best known variations of the multiplication algorithm is the “*Booth Recoding Algorithm*” described by Booth in 1951 [31]. This algorithm allows for the reduction of the number of partial products, thus speeding up the multiplication process. Generally speaking, the Booth algorithm is a case of using the redundant number system with the radix  $r$  higher than  $r = 2$  [1]. Earlier 2’s complement multipliers required data-dependent correction cycles if either operand is negative. The Booth algorithm can be used for both sign-magnitude numbers as well as 2’s complement numbers with no need for a correction term or a correction step.

A modification of the Booth algorithm was proposed by MacSorley in which a triplet of bits is scanned instead of two bits [32]. This technique has the advantage of reducing the number of partial products by roughly one half. This method is actually an application of a sign-digit representation in radix 4 [1]. The *Booth-MacSorley algorithm*, usually called the *modified Booth algorithm* or simply the *Booth algorithm*, can be

generalized to any radix. However, a three-bit recoding (case of radix 8) would require the following set of digits to be multiplied by the multiplicand: 0,  $\pm 1$ ,  $\pm 2$ ,  $\pm 3$ ,  $\pm 4$ . The difficulty lies in the fact that  $\pm 3Y$  is computed by summing (or subtracting)  $Y$  to  $\pm 2Y$ , which means that a carry propagation occurs. The delay caused by the carry propagation renders this scheme to be slower than a conventional one. Consequently, only the two-bit (radix 4) Booth recoding is used.

Booth recoding necessitates the internal use of 2's complement representation in order to efficiently perform subtraction of the partial products as well as additions. However, floating-point standard specifies sign magnitude representation which is also followed by most of the nonstandard floating-point numbers in use today. The Booth algorithm [31] is widely used for 2's complement multiplication, since it is easy to implement.

Booth recoding is performed within two steps: *encoding* and *selection*. The purpose of the encoding is to scan the triplet of bits of the multiplier and define the operation to be performed on the multiplicand, as shown in Table 10.1.

TABLE 10.1 Booth Recording

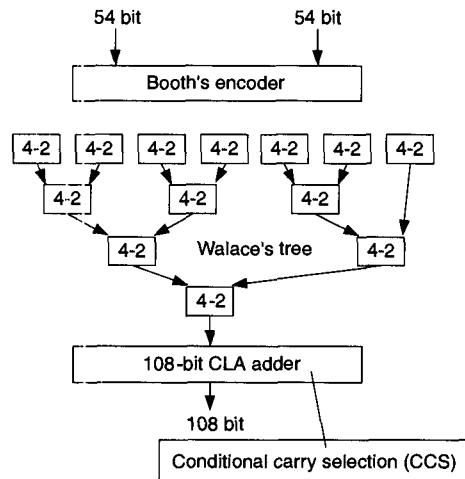
$x_{i+2}x_{i+1}x_i$	Add to Partial Product
000	+0Y
001	+1Y
010	+1Y
011	+2Y
100	-2Y
101	-1Y
110	-1Y
111	-0Y

The advantage of Booth recoding is that it generates roughly one half of the partial products as compared to the multiplier implementation, which does not use Booth recoding. However, the benefit achieved comes at the expense of increased hardware complexity. Indeed, this implementation requires hardware for the *encoding* and for the *selection* of the partial products ( $0, \pm Y, \pm 2Y$ ).

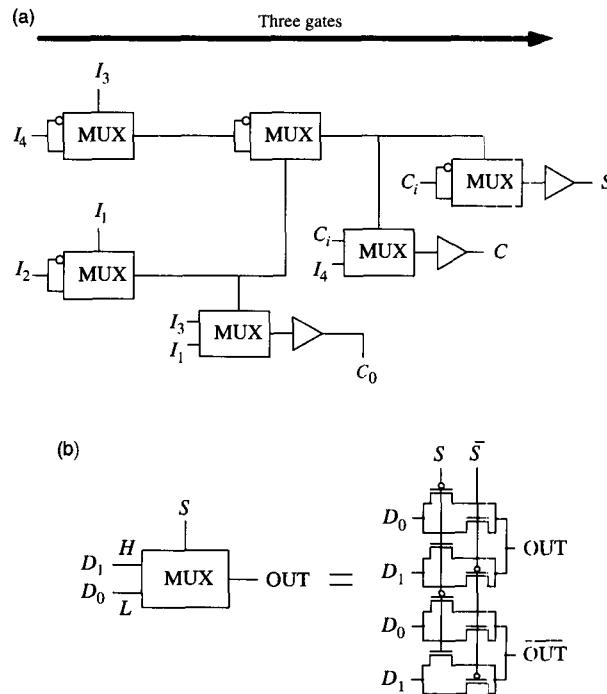
#### 10.3.2.6 Hitachi's DPL Multiplier

Hitachi's DPL multiplier was the first one to achieve under 5 ns speed for a double-precision floating-point mantissa imposed by the increasing demands on the operating frequency of modern microprocessors [12], [33]. This multiplier is of a regular structure including: (a) a Booth recoder, (b) a partial product reduction tree (Wallace Tree) and (c) a final carry propagate adder (CPA) as shown in Fig. 10.17.

The key to performance of Hitachi's multiplier is in the use of DPL circuits and the efficiency with which DPL can realize 4:2 compressor. The structure of Hitachi's 4:2 compressor is shown in Fig. 10.18(a). The realization of the 4:2 function consists entirely of DPL multiplexers, which introduce only one pass-transistor delay in the critical path as shown in Fig. 10.18(b). Indeed later studies [35] recognized this structure as one of the fastest partial product reduction tree (PPRT) realizations. For larger-size multipliers this PPRT may start showing degraded performance because of the long pass-transistor chain, which is equal to the number of 4:2 compressors used in the PPRT.



**Figure 10.17** Organization of Hitachi's DPL multiplier [33].



**Figure 10.18** (a) Hitachi's 4:2 compressor structure and (b) DPL multiplexer circuit [12].

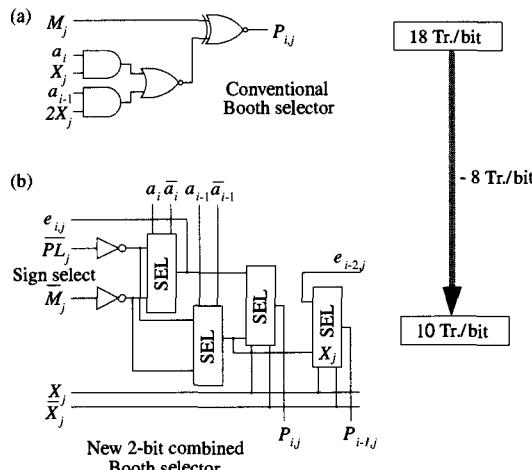
### 10.3.2.7 Inoue's Multiplier

The high-speed multiplier published by Inoue, et al. [35] employs two novel techniques in achieving very fast (4.1 ns delay)  $54 \times 54$ -bit parallel multiplier implemented in  $0.25\text{ }\mu\text{m}$  CMOS technology. The first novelty introduced is in a new design of the *Booth Encoder* and *Booth Selector* for generation of partial products. The encoding used in Inoue's multiplier is shown in Table 10.2.

**TABLE 10.2** Truth Table for Second-order Modified Booth Encoding

Inputs				Usual			Sign Select			
$b_{j+1}$	$b_j$	$b_{j-1}$	Func.	$X_j$	$2X_j$	$M_j$	$X_j$	$2X_j$	$PL_j$	$M_j$
0	0	0	0	0	0	0	0	1	0	0
0	0	1	+A	1	0	0	1	0	1	0
0	1	0	+A	1	0	0	1	0	1	0
0	1	1	+2A	0	1	0	0	1	1	0
1	0	0	-2A	0	1	1	0	1	0	1
1	0	1	-A	1	0	1	1	0	0	1
1	1	0	-A	1	0	1	1	0	0	1
1	1	1	0	0	0	1	0	1	0	0

$X_j$  – partial product,  $PL_j$  – positive partial product,  $M_j$  – negative partial product,  $B$  – multiplier (encoded),  $A$  – multiplicand,  $P = A \times B$



**Figure 10.19** (a) Regular Booth selector and (b) modified Booth selector [35].

There are two bits used for generation of sign of the partial product:  $M_j$  (for negative) and  $PL_j$  (for positive). Though this may seem to be redundant at first sight, it allows for a simpler implementation of the Booth encoder which does not require an XOR gate in the critical path. The equations for the Booth selector using regular and modified Booth encoding are listed:

$$P_{i,j} = (a_i \cdot X_j + a_{i-1} \cdot 2X_j) \oplus M_j \quad (i = 0, 1, 2, \dots, n-1 \quad j = 0, 2, 4, \dots, n-4, n-2) \quad \dots \text{regular Booth encoder} \quad (\text{a})$$

$$P_{i,j} = (a_i \cdot PL_j + \bar{a}_i \cdot M_j)X_j + (a_{i-1} \cdot PL_j + \bar{a}_{i-1} \cdot M_j) \cdot 2 \cdot X_j \quad (i = 0, 1, 2, \dots, n-1 \quad j = 0, 2, 4, \dots, n-4, n-2) \quad \dots \text{modified Booth encoder} \quad (\text{b})$$

Modified equations (b) obtained from Table 10.2 yield simpler Booth selector implementation than the regular case. The modified Booth selector is shown in Fig. 10.19(b) versus the regular Booth selector shown in Fig. 10.19(a).

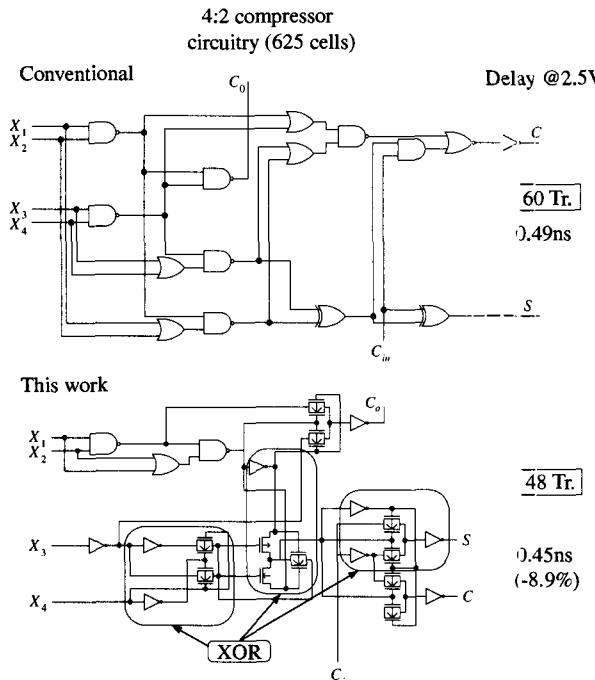


Figure 10.20 Pass-transistor implementation of the 4:2 compressor [35].

The modified Booth selector requires 10 transistors per bit as compared to the regular Booth selector which requires 18 transistors per bit for its implementation. The modification shown in Table 10.2 yields 44% reduction in the transistor count for the Booth selector of the  $54 \times 54$ -bit multiplier. Because the total number of transistors used for the Booth encoder in a  $54 \times 54$ -bit multiplier is only 1.2% of the total, modification of the Booth encoder resulting from Table 10.2 does not result in significant transistor savings. However, the use of the new Booth encoder resulted in a slight improvement in speed.

The second novelty in Inoue's multiplier is the pass-transistor implementation of the 4:2 compressor, which is shown in Fig. 10.20.

Inoue realized that there are  $2^6$  possible implementations of the 4:2 compressor. Out of the total number of  $2^6$  they have chosen the one that yields the minimal transistor count yet maintaining the speed within 5% of the fastest possible realization. This resulted in 24% savings in transistor count in the partial product reduction tree as compared to the earlier designs [34]. The transistor savings more than offset the 5% speed degradation by yielding more area and power-efficient design. It could be argued that the area improvement resulted in a better speed in the final implementation, which the simulation tools were not able to show.

## 10.4 CONCLUSION

In the past, a thorough examination of the algorithms with respect to particular technology has been only partially done. The merit of the new technology is to be evaluated by its ability to efficiently implement the computational algorithms. In

other words, the technology is developed with the aim to efficiently serve the computation. The reverse path—evaluating the merit of the algorithms—should also be taken. Therefore, it is important to develop computational structures that fit well into the execution model of the processor and are optimized for the current technology. In such a case, optimization of the algorithms is performed globally across the critical path of its implementation.

The ability to integrate 100 millions of transistors onto silicon has changed our focus and the way we think. Measuring the quality of the algorithm by the minimum number of devices used has simply vanished from the picture. However, new concerns such as power have entered it.

## REFERENCES

- [1] A. Avizienis, "Digital Computer Arithmetic: A Unified Algorithmic Specification," *Symposium on Computers and Automata*, Polytechnic Institute of Brooklyn, April 13–15, 1971.
- [2] Earl E. Swartzlander, *Computer Arithmetic*, vols. 1 and 2. IEEE Computer Society Press, Piscataway, NJ, 1990.
- [3] K. Hwang, *Computer Arithmetic: Principles, Architecture and Design*. John Wiley and Sons, New York, 1979.
- [4] S. Waser and M. Flynn, *Introduction to Arithmetic for Digital Systems Designers*. Holt, Rinehart and Winston, 1982.
- [5] M. Ercegovac, "Digital Systems and Hardware/Firmware Algorithms," Chapter 12 in *Arithmetic Algorithms and Processors*. John Wiley and Sons, New York, 1985.
- [6] T. Kilburn, D. B. G. Edwards, and D. Aspinall, "Parallel Addition in Digital Computers: A New Fast "Carry" Circuit," *Proceedings IEE*, vol. 106, Pt. B, p. 464, September 1959.
- [7] V. G. Oklobdzija and E. R. Barnes, "Some Optimal Schemes for ALU Implementation in VLSI Technology," *Proceedings of 7th Symposium on Computer Arithmetic*, June 4–6, 1985, University of Illinois, Urbana, IL.
- [8] V. G. Oklobdzija, "Simple And Efficient CMOS Circuit For Fast VLSI Adder Realization," *Proceedings of the International Symposium on Circuits and Systems*, pp. 1–4, 1988.
- [9] C. Nagendra, et al., "A Comparison of the Power-Delay Characteristics of CMOS Adders," *Proceedings of the International Workshop on Low-Power Design*, 1994.
- [10] K. Yano, et al., "A 3.8 ns CMOS 16 × 16-b Multiplier Using Complementary Pass-Transistor Logic," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 388–395, April 1990.
- [11] M. Suzuki, et al., "A 1.5 ns 32b CMOS ALU in Double Pass-Transistor Logic," *Digest of Technical Papers, 1993 IEEE Solid-State Circuits Conference*, San Francisco, February 24–26, 1993.
- [12] N. Ohkubo, et al., "A 4.4-ns CMOS 54 × 54-b Multiplier Using Pass-transistor Multiplexer," *Proceedings of the Custom Integrated Circuits Conference*, San Diego, CA, May 1–4, 1994.
- [13] Weinberger and J. L. Smith, "A Logic for High-Speed Addition," *National Bureau of Standards, Circulation* 591, pp. 3–12, 1958.
- [14] Naini, D. Bearden, and W. Anderson, "A 4.5 nS 96-b CMOS Adder Design," *IEEE 1992 Custom Integrated Circuits Conference*, 1992.
- [15] B. D. Lee and V. G. Oklobdzija, "Improved CLA Scheme with Optimized Delay," *Journal of VLSI Signal Processing*, vol. 3, pp. 265–274, 1991.
- [16] P. M. Kogge and H. S. Stone, "A Parallel Algorithms for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Transactions on Computers*, vol. C-22, no. 8, Aug. 1973, pp. 786–793.
- [17] A. Bilgory and D. D. Gajski, "Automatic Generation of Cells for Recurrence Structures," *Proceedings of the 18th Design Automation Conference*, Nashville, TN, 1981.

- [18] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, March 1982, pp. 260–264.
- [19] H. Ling, "High Speed Binary Adder," *IBM Journal of Research and Development*, vol. 25, no. 3, May 1981, p. 156.
- [20] S. Naffziger, "A Sub-Nanosecond  $0.5\text{ }\mu\text{m}$  64 b Adder Design," *1996 IEEE International Solid-State Circuits Conference*, Digest of Technical Papers, San Francisco, February 8–10, 1996, pp. 362–363.
- [21] Sklanski, "Conditional-Sum Addition Logic," *IRE Transaction on Electronic Computers*, EC-9, pp. 226–231, 1960.
- [22] O. J. Bedrij, "Carry-Select Adder," *IRE Transaction on Electronic Computers*, June 1962.
- [23] D. Dobberpuhl, et al., "A 200 MHz 64-b Dual-Issue CMOS Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, Nov. 1992.
- [24] S. D. Pezaris, "A 40 ns 17-bit array multiplier," *IEEE Transactions on Computers*, vol. C-20, pp. 442–447, April 1971.
- [25] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEE Transactions on Electronic Computers*, EC-13, pp. 14–17, 1964.
- [26] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, March 1965.
- [27] W. J. Stenzel, "A Compact High Speed Parallel Multiplication Scheme," *IEEE Transaction on Computers*, vol. C-26, pp. 948–957, Feb. 1977.
- [28] A. Weinberger, "4:2 Carry-Save Adder Module," *IBM Technical Disclosure Bulletin*, vol. 23, Jan. 1981.
- [29] J. Fadavi-Ardekani, "M  $\times$  N Booth Encoded Multiplier Generator Using Optimized Wallace Trees," *IEEE Transactions on VLSI Systems*, vol. 1, no. 2, June 1993.
- [30] V. G. Oklobdzija, D. Villegier, and S. S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," *IEEE Transaction on Computers*, vol. 45, no. 3, March 1996.
- [31] A. D. Booth, "A Signed Binary Multiplication Technique," *Quarterly J. Mechan. Appl. Math.*, vol. IV, 1951.
- [32] O. L. MacSorley, "High Speed Arithmetic in Binary Computers," *Proceedings of IRE*, vol. 49, no. 1, Jan. 1961.
- [33] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 54 $\times$ 54-b Multiplier Using Pass-Transistor Multiplexer," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, March 1995, pp. 251–257.
- [34] G. Goto, et al., "A 4.1 nS Compact 54  $\times$  54-b Multiplier Utilizing Sign-Select Booth Encoders," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, Nov. 1997, pp. 1676–1682.
- [35] A. Inoue, R. Ohe, S. Kashiwakura, S. Mitari, T. Tsuru, T. Izawa, and G. Goto, "A 4.1 nS Compact 54  $\times$  54b Multiplier Utilizing Sign Select Booth Encoders," *1997 IEEE International Solid State Circuits Conference*, Digest of papers, San Francisco, 1997, p. 416.
- [36] M. Nagamatsu, et al., "A 15 nS 32  $\times$  32-bit CMOS Multiplier with an Improved Parallel Structure," *IEEE Custom Integrated Circuits Conference 1989*. Digest of technical papers.
- [37] P. Stelling, C. Martel, V. G. Oklobdzija, and R. Ravi, "Optimal Circuits for Parallel Multipliers," *IEEE Transaction on Computers*, vol. 47, no. 3, pp. 273–285, March 1998.
- [38] C. Mead and L. Conway, *Introduction to VLSI systems*. Addison-Wesley, Reading, MA, 1980.

PART  
**IV** | **CLOCKING**

Hamid Partovi  
*Advanced Micro Devices, Inc.*

The clocked storage element, a level-sensitive latch-pair or an edge-triggered flip-flop, is probably the single most debated and analyzed circuit structure in modern microprocessor designs. This distinction is not undeserved. Clocked storage elements partition every pipeline stage on the processor. They hold the current state and prevent the next state from entering a stage. They synchronize concurrent logic with different delays. Their design is tightly coupled to the clocking strategy and circuit topology, and since all timing paths start at and end with clocked storage elements, their multigate latency weighs heavily on cycle time.

Design considerations, and sometimes conflicting requirements, such as latency, load on the clock, handling clock uncertainties, power efficiency, the ability to incorporate logic, race and noise robustness, make the selection and the design of these elements highly complex.

The chapter begins with an overview of clocking and recommends a clocking strategy most suitable for high-performance designs. A simple latch-pair and a simple flip-flop are then used to define the timing characteristics of these elements. Consequently, the properties of sequential logic, including the effects of storage-element characteristics and clock uncertainties on cycle time and race, are developed. At this point, directives for robust design are outlined and the basic storage elements are modified to adhere to these guidelines.

A number of modern implementations of clocked storage elements are then presented and further design guidelines are set forth. These directives establish a common framework based on which one can define the figures-of-merit metrics for an accurate comparison of the clocked storage elements. The discussion of clocked storage elements is, at this point, extended to cover dynamic circuits. Related issues such as monotonic signaling as well as crossing the boundaries between static and dynamic circuit domains are covered. The chapter concludes with a comparison of the presented elements to propose a latching strategy for operating frequencies in the 1 GHz range.

## **11.1 ON CLOCKING STRATEGY**

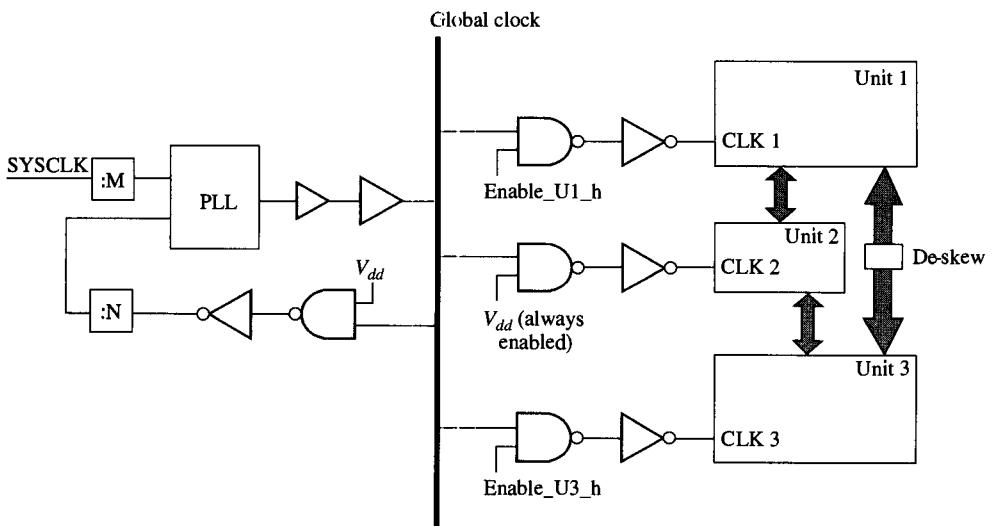
While sequential circuits can be implemented asynchronously, synchronous or clock-based design methodology presents a simple, structured, and deterministic approach to synchronize, order, and orchestrate multitudes of events in complex digital designs. In synchronous sequential circuits, switching events in various stages of the pipeline take place concurrently in response to a clock stimulus. New sets of inputs to pipeline stages

are sampled by storage elements and new computations ensue, changing the state of the sequential network. Once complete, results of computations await the next clock transition in order to advance to their next pipeline stage. In other words, the next cycle of operation cannot begin unless all current computations have completed and the system has come to rest. The *cycle time*,  $t_{CY}$ , or the clock period at which the sequential network will operate, must thus accommodate the longest delay of any stage in the network.

From this aspect, it is clear that uncertainties in the clock signal, or its deviation from the ideal periodic nature, will affect the cycle time and hence the performance of the system. The nonideal nature of the clock could also create conditions where the next state of a pipeline stage may race into a clocked storage element and corrupt its current state. Far worse than the effect on system performance, the disruption in ordering mechanism afforded by the clock causes a functional failure that cannot be corrected by manipulation of the clock period.

A major decision in the design of microprocessors is the choice of its clocking strategy. Most high-performance processors of the last decade have used either a single- or two-wire non-overlapped clocking. The increasing operating frequency and high functional integration of microprocessors, however, have made it exceedingly more difficult to *globally* produce, match, and distribute multiwire non-overlapping clocks. Moreover, the avoidance of race, which is the primary benefit of non-overlapping clocks, can now be guaranteed for single-wire clocking with advanced CAD tools for timing analysis. As a result, *the dominant trend in clocking is the generation and distribution of a single-wire global clock*.

To minimize clock skew, the load and routing parasitics of the global clock must be reduced. Additionally, in order to manage power dissipation, it is desirable to conditionally enable/disable different functional units of the processor. Both these goals can be achieved if the global clock is locally qualified and buffered, thus producing various clock domains (Fig. 11.1).



**Figure 11.1** Example of a clock network.

Since signals in different clock domains will communicate, the clock skew caused by buffering the common global clock, in addition to other uncertainties, must be carefully analyzed. This, at times, may lead to the necessity of explicit de-skewing circuits to ensure race-free operation between clock domains. Furthermore, to simplify the analysis of race *within* a clock domain, the single-wire global clock may be used to locally generate non-overlapping clocks.

## 11.2 THE NONIDEAL NATURE OF CLOCK SIGNALS

As discussed earlier, the nonideal nature of the clock can affect both the functionality and the performance of a sequential network. In this section, three components leading to the clock signal uncertainty, namely jitter, skew, and duty cycle will be studied.

### 11.2.1 Jitter

The periodicity of a clock signal is affected by the deviation of its edges from their expected transition time. The time difference between the actual transition and the expected transition of the clock is called jitter.

The global clock in microprocessors, GCLK, is generated by a *phase-locked loop* (PLL) from a system clock, SYSCLK, as shown in Fig. 11.1. The undesirable jitter characteristics of the clock are caused by noise sources such as *temporal* variations in the supply voltage and mismatches in the PLL circuitry.

Jitter comprises deterministic and random components. The former is usually a nonharmonic frequency modulation of the clock signal, exhibited as weak sidebands around the center clock frequency. The latter is a zero-mean random variable adding skirts to the center clock frequency and is known as phase noise in the frequency domain [1].

Jitter is quantified in one of two ways. If measured between *consecutive* edges of the global clock, it is called *cycle-to-cycle* or *short-term* jitter,  $t_{JS}$ ; if measured in reference to the system clock or a periodic signal of the same frequency, it is called *absolute* or *long-term* jitter,  $t_{JL}$ . While long-term jitter, which includes static phase error, accumulated phase error and the jitter of the system clock, can have peak-to-peak values of hundreds of picoseconds, short-term jitter can be as low as 20 ps [2].

Long-term jitter and short-term jitter affect the microprocessor (and its peripheral components) differently. Long-term jitter impacts the interface circuitry that operates between domains controlled by the system and global clocks. It limits the interface bandwidth. In addition, it must be considered in the calculation of the minimum delay restriction across these domains to avoid hold-time violation. Short-term jitter, in contrast, only affects the microprocessor cycle time and does not influence the minimum delay restriction between clocked storage elements.

The clock signal may in addition suffer from variations in its duty cycle. Caused by the deviation of clock-tree transistors from their nominal characteristics over process, temperature and supply voltage, variations in duty cycle add further uncertainty to the relative placement of the opposite edges of the clock. These variations affect the timing of level-sensitive sequential logic *much in the same manner as does jitter*; they are, however, largely absent in the edge-triggered timing considerations. Thus, when discussing the level-sensitive clocking, *the jitter of an edge referenced to the opposite edge will also include the variation in duty cycle*.

### 11.2.2 Skew

The time difference between *temporally equivalent* edges of two periodic signals of the same frequency and with nominally no phase difference, is called skew,  $t_{SK}$ . Similar to jitter, clock skew is composed of deterministic and random components.

In Fig. 11.1, buffers generating the local clocks vary in distance from the global clock generator, and thus observe different transition times of the GCLK. In addition, mismatches in local buffers, the load they drive, and routing parasitics further contribute to the *deterministic* components of skew between local clocks. *Spatial* differences in supply voltage, substrate noise, and coupling from other sources, produce variations in the latency of local buffers that make up the *random* components of skew.

It is very important to make the following distinction. Both jitter,  $t_{JS}$ , and clock skew,  $t_{SK}$ , affect the consecutive edges of the local clocks and hence reduce (or increase) the effective cycle time. In contrast, jitter *does not* impact the temporally equivalent or concurrent edges of the local clocks and, as will be discussed later, does not play a role in minimum delay restriction.

## 11.3 THE BASIC LATCH-PAIR

Figure 11.2(a) shows the basic level-sensitive latch: it comprises a transmission gate driven by a control signal, CLK, and followed by a buffer. The latch passes the data input,  $D$ , to its output,  $Q$ , when CLK is asserted; when de-asserted, the input is ignored and the latch *dynamically* holds the most recent input value sampled at the time CLK was de-asserted. The latch is called a transparent high latch (THL) if its CLK is high

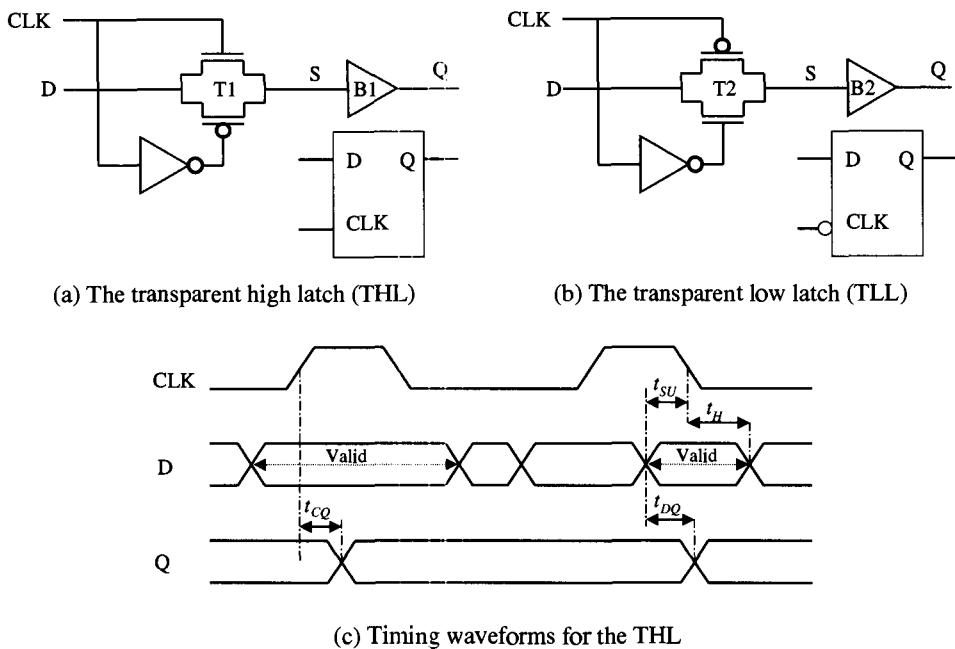


Figure 11.2 The level-sensitive latch-pair.

when asserted as in Fig. 11.2(a); it is called a transparent low latch (TLL) if its CLK is low when asserted [Fig. 11.2(b)].

Waveforms of Fig. 11.2(c) show the timing characteristics of the THL. If the data is set up prior to the rising edge of CLK, the *latch latency* is defined by the delay from the rising CLK edge to  $Q$  becoming valid,  $t_{CQ}$ . If the data input transitions when CLK is high, i.e., when THL is transparent, the latency is defined from the transition of data to  $Q$  becoming valid,  $t_{DQ}$ . To successfully transfer and hold the data as CLK falls, i.e., as the latch becomes opaque, a minimum set-up of data input to the falling CLK edge,  $t_{SU}$ , is required. Furthermore, the data must be *held* stable for a minimum time,  $t_H$ , after the falling CLK edge to ensure that the proper data has been captured.

An interesting interpretation of latch behavior is depicted in Fig. 11.4(a). It includes the latch latency,  $t_{DQ}$ , as a function of the data-to-clock delay,  $t_{DC}$ , for the THL. Latency approaches infinity when the minimum set-up time is violated, i.e., the latch fails to capture its input data. As  $t_{DC}$  is increased,  $t_{DQ}$  plateaus at a minimum. Once  $t_{DC}$  is greater than the assertion period of the clock,  $W$ , data sets up prior to and awaits the assertion edge of the clock, and hence  $t_{DQ}$  monotonically increases as a linear function of  $t_{DC}$ .

The figure further illustrates that the uncertainties in the clock-edge affect latch latency for large and small data-to-clock set-up times. In contrast, if the latch input changes well within the clock assertion period and away from the edges,  $t_{DQ}$  will not be influenced by edge uncertainty. As will be described later, this characteristic provides another property of level-sensitive latches called *slack-passing* or *time-borrowing*.

## 11.4 THE BASIC FLIP-FLOP

In contrast to a level-sensitive latch which responds to the changes at its input for as long as CLK is asserted, a flip-flop samples its input within a brief period called the *aperture window* around the *assertion edge* of a control signal, CLK. During this period, the flip-flop can be thought of as transparent; it updates its output with the new value and advances its current state to the next state. At any other time the flip-flop is opaque and ignores any change at its input. If the flip-flop captures its input data on the positive edge of CLK, it is called positive edge-triggered; likewise, it is called a negative edge-triggered flip-flop if it samples the data on the negative edge of the CLK signal.

Figure 11.3 includes the positive edge-triggered master-slave flip-flop and its timing waveforms. The circuit comprises a *slave* transparent high latch (THL) preceded by a *master* transparent low latch (TLL), both of which are gated by a CLK signal.

The circuit operates as follows. When CLK is low, the master TLL is transparent, sampling the changes at its input,  $D$ . These changes, however, are ignored by the slave THL as it is opaque; as a result  $Q$  holds its state. On the rising edge of CLK, TLL becomes opaque and holds its state at IS. Meanwhile, THL becomes transparent, updating  $Q$  by sampling its input IS. Although the THL remains transparent for as long as the clock is asserted, its input IS, which is the output of the *now opaque* TLL, will not change again. Thus, the output of the circuit,  $Q$ , is subject to being updated only once per cycle when *triggered by the positive edge of the clock*.

The master-slave flip-flop of Fig. 11.3 is subject to *internal race*. The delay introduced by the inversion of the clock causes a time window on the *falling* edge of the clock where both latches are transparent. According to the figure, it can be seen that P1 of transmission gate T1 is activated before P2 of T2 is turned off. If the clock inversion delay is larger than the delay through transmission gates T1 and T2 and inverter B1,

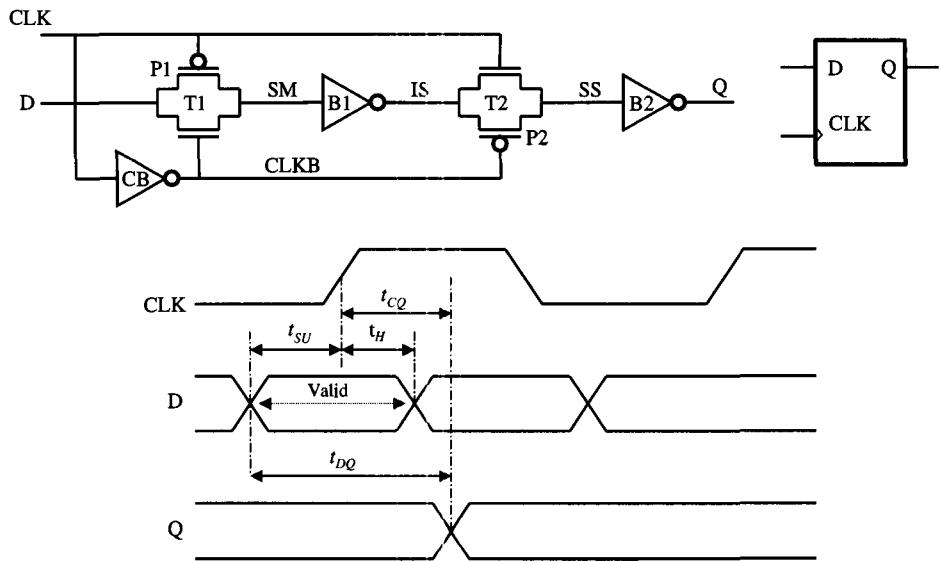


Figure 11.3 The master-slave flip-flop and its timing waveforms.

bad data will sneak through to the state node of the slave latch, SS, and disturb the state of the flip-flop. The inclusion of inverter B1, although logically redundant, is intended to reduce the sensitivity to internal race. Furthermore, the clock-inverting buffer, CB, is designed to have a high P/N ratio to reduce its rising-edge delay. A variation of the master-slave flip-flop called the C<sup>2</sup>MOS register has been proposed [3] that is insensitive to internal race but suffers from poor latency.

As depicted in Fig. 11.3, all timing parameters of the flip-flop are referenced to the assertion edge of CLK. The data must be stable for a minimum time,  $t_{SU}$ , before and must be held for at least  $t_H$ , after CLK transitions. The aperture window of the flip-flop is hence  $t_{SU} + t_H$ . The latency of the flip-flop is the delay from input transitioning to Q becoming valid,  $t_{DQ}$ , and comprises the additive components of  $t_{DC}$  and  $t_{CQ}$ .

Figure 11.4(b) shows the flip-flop latency versus data-to-clock delay, or  $t_{DC}$ , including the clock uncertainty. In the case of the master-slave flip-flop, the  $t_{SU}$  is

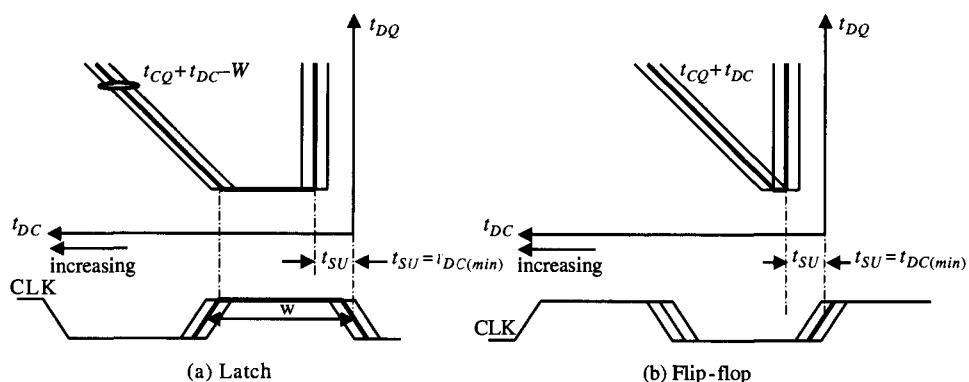


Figure 11.4 Latch and flip-flop latencies ( $t_{DQ}$ ) vs. data-to-clock set-up time ( $t_{DC}$ ).

solely determined by the master latch. As long as the minimum set-up time is not violated, on the rising edge of the clock, the data is transferred to the output with a delay equal to the  $t_{CQ}$  of the slave latch. Thus, with a sufficiently large  $t_{DC}$ ,  $t_{CQ}$  of the flip-flop (and of the slave) is fixed and independent of  $t_{DC}$ . The latency of the flip-flop,  $t_{DQ}$ , is then either undefined, i.e., when a minimum  $t_{DC}$ , or  $t_{SU}$ , is not met or monotonically increases with  $t_{DC}$ . Unlike level-sensitive latches which can operate in a timing regime unaffected by clock uncertainty [see Fig. 11.4(a)], the latency of most edge-triggered storage elements is linearly impacted by this uncertainty.

## 11.5 RULES FOR ROBUST DESIGN—1

Design considerations for latching elements in the transparent state are similar to those of combinational logic in the same circuit topology. Additional guidelines are necessary to ensure that the latching element can hold state when it is opaque. To this end and based on the noise level that can be sustained by the storage element, a *noise margin* is quantified. Noise sources are then allotted a percentage of this margin that they cannot exceed.

The set of rules presented here is intended to minimize the storage-element sensitivity to a number of noise sources as described in the following.

1. **Noise on data input.** Referring to the basic latch of Fig. 11.2(a), it can be seen that the data,  $D$ , is driven into the source of the transmission gate T1. Consider that the clock is de-asserted and the state node holds a logical ‘1’. Furthermore assume that the input is at a logical ‘0’ generated by a distant driver as in Fig. 11.5. In addition to the ground level differences between the driving buffer and the latch, a *low-down* coupling event to the data signal can be sufficient to activate the NMOS device in the transmission gate and discharge the state node, S.

- (1) Noise on input
- (2) Leakage
- (3)  $\alpha$ -Particle and cosmic rays
- (4) Unrelated signal coupling
- (5) Power supply ripple

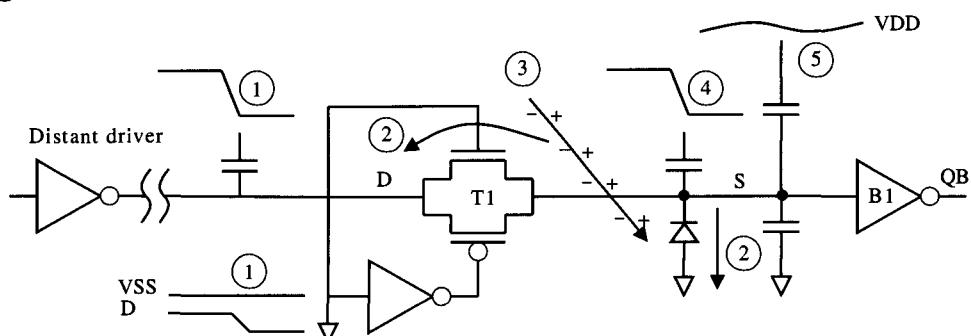


Figure 11.5 Sources of noise affecting latch state node when opaque.

Depending on the availability of advanced CAD tools, some designs implement the following checks to protect the storage node against noise on data input:

- An *absolute distance check*. To minimize spatial differences of ground and supply levels, an absolute limit is enforced on the driver distance from the receiving storage element.
- A *relative distance check*. An upper limit is placed on the driver distance to the latch *normalized to its drive strength*, thus guaranteeing a maximum coupling limit.

The problem of input noise can be avoided if the latch input is gate-isolated, i.e., buffered (Fig. 11.6), alas, with an increase in latch latency. In the highly complex microprocessor designs, it is best to use buffered latching constructs generally and then selectively employ an unbuffered version in the particularly difficult timing paths. As a result, *a robust design requires that the input of a latching element be gate-isolated*.

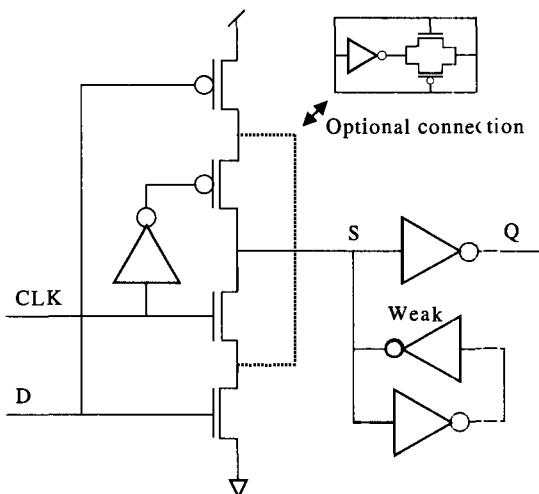


Figure 11.6 Gate-isolated pseudo-static THL.

2. **Leakage.** The basic latch-pair and flip-flop described earlier hold their state dynamically, i.e., the data is stored on the gate capacitance of the buffering drivers, B1 and B2 of Figs. 11.2 and 11.3. Subthreshold device and junction leakages gradually rob the stored charge off the state node. Since most microprocessors allow for modes in which clock can be slowed down or completely stopped, the state node must replenish its charge regeneratively by employing a weak feedback inverter. Latching elements constructed in this manner are called *pseudo-static* (Fig. 11.6). Again, the weak feedback has a cost: a minor degradation in the latency of the storage element. Despite this, *a robust design requires that the state of the storage element be held pseudo-statically*.
3. **Cosmic ray and  $\alpha$ -particle perturbations.**  $\alpha$ -Particles emanating from the decay of radioactive isotopes found in packaging materials and cosmic ray particles abundant at high altitudes act as external noise sources that may cause a storage node to lose its state.

Once a particle strikes the silicon, it generates electron-hole pairs along its track, of which some are collected by a nearby junction. For a dynamically held

storage node, the collected charge causes a voltage drop, which is inversely proportional to the capacitance of the node. For a given technology, the amount of charge collected by a junction depends on its area, its distance from, and the energy and the angle of incidence of the particle.

Due to the poor transient response of the feedback inverter, in the event of a particle strike, initially, the regeneratively held storage node does not fare much better than its dynamic counterpart. However, as long as the noise margin allotment is not violated, over time, the feedback inverter will completely restore the node to its original level. Thus, *immunity to  $\alpha$ -particle and cosmic ray perturbations places a lower bound on the storage node capacitance*.

4. **Unrelated signal coupling.** Another source of noise is the coupling of unrelated signals to the storage node. Since the placement and usage of latching elements should not impose restrictive limitations, the storage node must be layed out compactly, thereby minimizing its exposure to adverse coupling. Further, as in item 3, a minimum fixed capacitance rule for the storage element should be observed. To recap: *for robust design, ensure minimum exposure of the storage node to adverse coupling and guarantee a lower limit for the fixed capacitance present at the node*.
5. **Power supply ripple.** The microprocessor power supply network comprises a lossy tank circuit that exhibits a mid-frequency ripple in the 50 to 100 MHz range for processors operating above 500 MHz. The peak-to-peak variation of the supply rail in reference to the ground rail can be as large as 10% of the nominal value of VDD.

A dynamically held storage node tracks the supply ripple in *phase* but at an attenuated level determined by the capacitive arrangement of the storage node. In the pseudo-static case, the storage node follows the supply voltage with a *phase shift* but with less attenuation. The inability of the storage node to fully track the supply rail variations reduces its noise margin.

The ratio of the storage node capacitance to the supply rails and the strength of the feedback inverter determine how well the node tracks the supply rails. *For robust design, transistors in the feedback inverter should be made large enough to ensure adequate tracking of the storage node with either of the supply rails when referenced to the other rail*.

The pseudo-static storage element provides yet another very important advantage over its dynamic counterpart. It tends to decouple nontemporal noise events as the feedback inverter restores the storage node voltage to its full level in time. In the opaque state, noise events affecting a dynamically held storage node are, in contrast, *additive*. Consequently, the dynamic noise margin limits are far more stringent than those for the pseudo-static storage element.

Figure 11.6 shows a gate-isolated pseudo-static THL. As described in this section, it provides a far more robust operation when compared to the basic THL of Fig. 11.2(a). In the same manner, the TLL and the master–slave flip-flop can be modified for enhanced robustness.

## 11.6 TIMING PROPERTIES OF SEQUENTIAL LOGIC

Pipeline stages of sequential logic are formed by clocked storage elements interspersed with combinational circuits. In latch-based designs, for correct operation of logical pipelines, latches with opposite clock assertion levels must partition the combinational

gates. In contrast, a single flip-flop along with logic forms a pipeline stage. And thus, a *flip-flop* is logically equivalent to a *pair of latches* with opposite assertion levels.

This section describes the timing properties of sequential logic: the maximum frequency at which the circuit can operate, and the minimum latch-to-latch delay required to avoid hold time violation. Due to its conceptual simplicity, edge-triggered clocking for flip-flop based designs is covered first. The discussion of level-sensitive clocking for latch-based designs then follows. Both clocking schemes are presented with single-wire clocks. The attributes of multiwire clocking for latch-based designs are probed at the end of this section.

### 11.6.1 Edge-Triggered Clocking

Figure 11.7 shows consecutive pipeline stages formed by two flip-flops, logic comprising a long path and logic comprising a short path. The flip-flops are clocked by CLK1 and CLK2, which have the same period,  $t_{CY}$ . CLK1 lags CLK2 by  $t_{SK}$ . Accounting for jitter, the time difference between the successive edges of CLK1 and CLK2, or the effective cycle time is

$$t_{CY\ eff} = t_{CY} - (t_{SK} + t_{JS}) \quad (11.1)$$

Triggered by CLK1 and after a  $t_{CQ}$  delay of the flip-flop, the evaluation of the long path begins and must complete such that the desired data sets up properly at the input of the receiving flip-flop before the rising edge of CLK2. Consequently, the maximum time available for evaluation of combinational logic in one clock period,  $t_{CL\ max}$  is expressed as

$$t_{CL\ max} = t_{CY} - (t_{SK} + t_{JS}) - (t_{SU} + t_{CQ}) \quad (11.2)$$

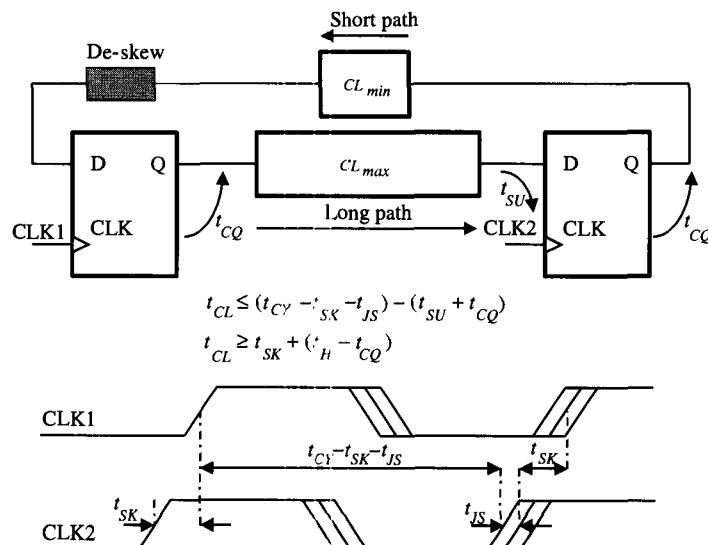


Figure 11.7 Minimum and maximum timing restrictions for edge-triggered clocking.

It can be seen from Eq. (11.2) that both clock skew and jitter adversely affect the maximum amount of logic that can be computed in a clock period. Failing the maximum delay restriction, which is also called the *set-up time violation*, can be moderated by running the clock at a lower frequency.

Similarly, on the rising edge of CLK2 and after a  $t_{CQ}$  delay of the flip-flop, the short path evaluates and begins to set up at the input of the receiving flip-flop. If the time to complete this operation is less than the flip-flop hold time in addition to the clock skew, the next state of the flip-flop will sneak in and corrupt the current state of the pipe stage. Hence, for correct operation, a minimum delay restriction is placed on combinational logic in a pipeline stage:

$$t_{CL\min} = t_{SK} + (t_H - t_{CQ}) \quad (11.3)$$

Failing the minimum delay restriction of Eq. (11.3), i.e., violating the *hold time of the flip-flop* relates to the *uncertainty present with temporally equivalent edges* of CLK1 and CLK2, and thus jitter does not enter in Eq. (11.3). Likewise, increasing the clock period cannot correct the violation of hold time.

### 11.6.2 Level-Sensitive Clocking

As was discussed in Section 11.3, the latency of the level-sensitive latch,  $t_{DQ}$ , is unaffected by clock edge uncertainty provided that the input data is set up well within the assertion period of the latch. This section further elaborates on this theme and shows that when certain timing requirements are observed, the effective cycle time of level-sensitive clocking is not impacted by clock edge uncertainty. Consider Fig. 11.8 where logic along with a THL/TLL pair, clocked by CLK1 and CLK2, forms a pipeline stage. A third latch, clocked by CLK3, receives the final output of this stage.

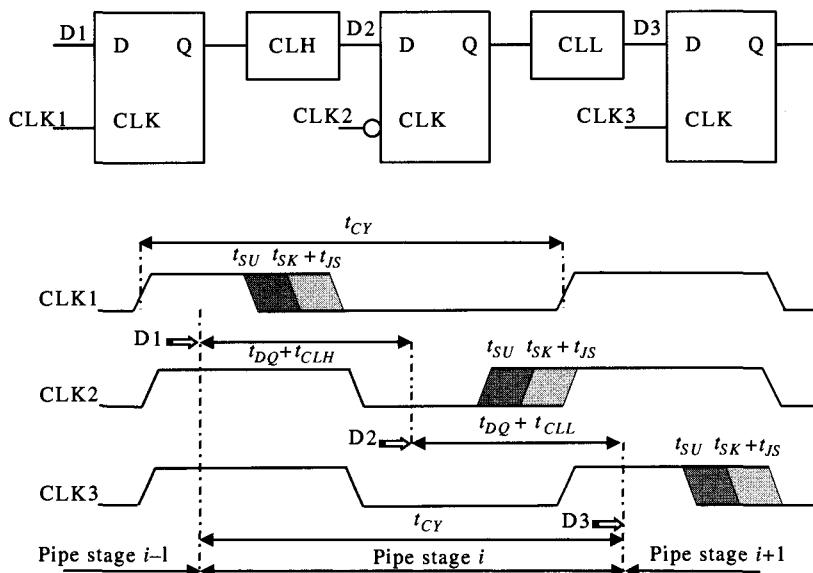


Figure 11.8 Skew-tolerant level-sensitive clocking.

As shown in Fig. 11.8, the relevant edges of the clock signals can be displaced from their intended position by  $t_{SK} + t_{JS}$ . Thus if the latch input transitions *within the assertion period* but *no closer* to the de-assertion edge by  $t_{SU} + (t_{SK} + t_{JS})$ , then the delay through the latch,  $t_{DQ}$ , is completely independent of the position of the clock edges.

Assuming a balanced design where the combinational delays of pipeline stages,  $t_{CL} = t_{CLH} + t_{CLL}$ , are all the same, and that the latch inputs become available well within their transparency period, then the cycle time can be expressed as

$$t_{CY} = t_{CL} + 2 \times t_{DQ} \quad (11.4)$$

or equivalently,

$$t_{CL} = t_{CY} - 2 \times t_{DQ} \quad (11.5)$$

Figure 11.9 shows the timing waveforms leading to the violation of hold time for THL. Clocked by CLK2, the THL is preceded by a TLL with intervening logic CLL. CLK1, which clocks the TLL, leads CLK2. On the falling edge of CLK1, the TLL becomes transparent and passes D1 to Q1 after a  $t_{CQ}$  delay. Responding to Q1 and after the delay of the combinational logic CLL, D2 transitions. As it is apparent from the figure, the latch hold time is violated *unless* the time delay of the above operation is greater than the sum of clock skew and latch hold time. Thus,

$$t_{CLL\ min} = t_{SK} + (t_H - t_{CQ}) \quad (11.6)$$

The minimum delay restriction for *singe-wire* level-sensitive clocking, while similar to that of the edge-triggered clocking, must be observed on both phase boundaries of the clock, and thus,

$$t_{CLH\ min} = t_{SK} + (t_H - t_{CQ}) \quad (11.7)$$

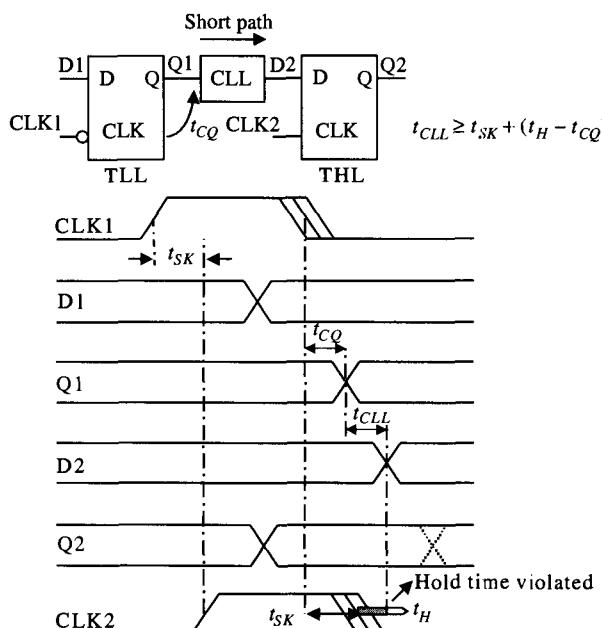


Figure 11.9 Minimum timing restriction for level-sensitive clocking.

### 11.6.3 Slack-Passing and Time-Borrowing

Slack-passing and time-borrowing, two powerful but somewhat esoteric properties of level-sensitive clocking, are presented in this section. The ability to pass the unused time of a pipeline stage forward is called slack-passing. Likewise, the ability to take away time necessary to complete an operation from a succeeding stage is called time-borrowing.

Equation (11.4) is derived based on the following assumptions: (a) pipeline stage design is balanced, i.e., the stage delay for all pipeline stages are the same, and that (b) latch inputs transition well within the transparency period of their corresponding latch.

Slack-passing and time-borrowing relate to the *unbalanced pipeline designs*. Figure 11.10 shows the timing waveforms for a pipeline stage  $i$  which has a large delay. The combinational delay of the preceding stage,  $(i - 1)$ , is short and hence its output,  $D_1$ , sets up early and in the limiting case, before the assertion of  $CLK_1$ . Comparing this with  $D_1$  of Fig. 11.8, it becomes clear that stage  $(i - 1)$  makes available (or slack passes) a portion of its unused time to be used on demand by the following stage. The maximum time made available is of course *limited by the assertion edge of  $CLK_1$* . As is the case in Fig. 11.10, on the rising edge of  $CLK_1$  and immediately after a  $t_{CQ}$  delay of  $THL_1$ , the evaluation of pipe stage  $i$  begins.

Since the delay of stage  $i$  is large, not only will it take advantage of the *voluntarily* passed slack by stage  $(i - 1)$ , it will *forcibly* take the time it needs by encroaching in the time allotted to stage  $(i + 1)$ . The maximum encroachment is *limited only by the falling edge of  $CLK_3$*  and its uncertainty. Thus, following the progression of data,  $D_1 \rightarrow D_2 \rightarrow D_3$ , in Fig. 11.10, a pipeline stage can, theoretically, have a maximum delay given by

$$t_{CL\ max} = t_{CY} - (t_{CQ} + t_{DQ} + t_{SU}) + (t_{CY}/2 - t_{SK} - t_{JS})$$

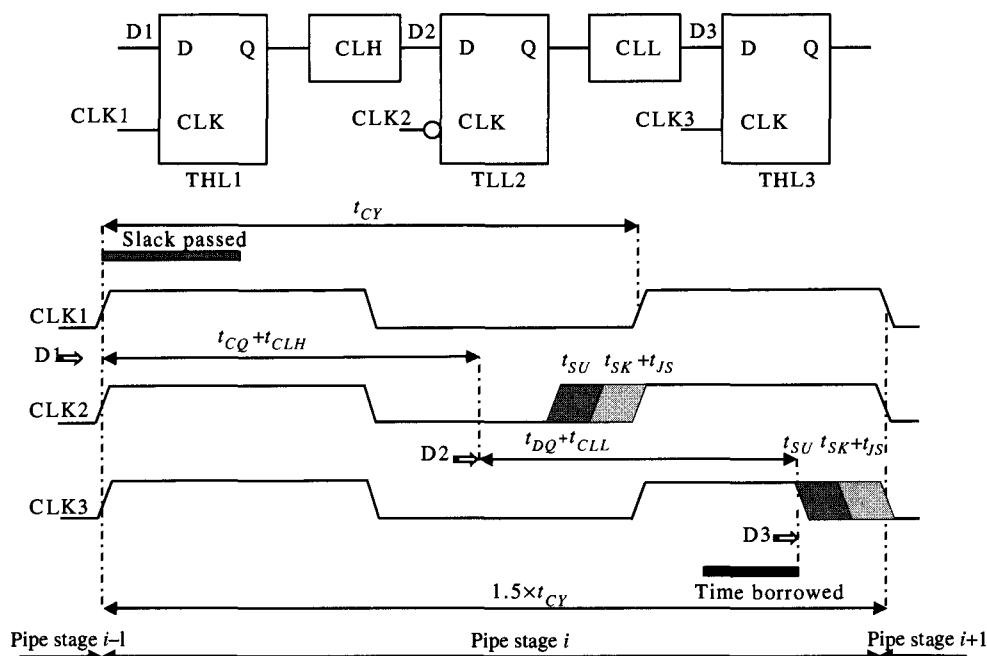


Figure 11.10 Maximum slack-passing and cycle-borrowing.

or

$$t_{CL\ max} \cong t_{CY} - 2 \times t_{DQ} + (t_{CY}/2 - t_{SK} - t_{JS}) \quad (11.8)$$

Regarding Eq. (11.8), the following observations can be made:

- Unlike a balanced design where the time available per stage is not affected by clock-edge uncertainty, i.e.,  $t_{CL} = t_{CY} - 2 \times t_{DQ}$ , Eq. (11.8) shows dependence on skew, jitter, and duty cycle variation.
- The theoretical maximum time available for a single-pipe stage is defined by the *assertion edge* of the latch starting a stage, THL1 and the *de-assertion edge* of the latch starting the next stage, THL3.
- The unused time made available by a stage with a short delay may be used on demand by any succeeding stage. It may be taken all by a single stage or the slack may be shared by many.
- If a stage forcibly borrows time from *a stage with no time to give*, the effective cycle time is influenced by the uncertainty of the clock. Under these conditions, the latch data-input sets up as its clock is being de-asserted and much like the master-slave flip-flop incurs the penalty of jitter and skew.

Well-designed micro-architectures attempt to have relatively balanced pipeline stage delays. Level-sensitive clocking is capable of absorbing variations in stage latencies as seen by the clock edge as well as the uncertainties of the clock edge when referenced to pipeline outputs.

#### 11.6.4 Two-Wire Non-Overlapping Clocks

As discussed in Section 11.6.2, the minimum delay restriction for *single-wire* level-sensitive clocking must be considered twice per cycle on each phase boundary of the clock. Using two-wire non-overlapping clocks with adequate tolerances can in principle eliminate the minimum delay restriction altogether. It must be realized, however, that the *dead-time* between the assertion edges of non-overlapped clocks reduces the period during which latches are transparent. In turn, this limits the degree of slack-passing and time-borrowing, and, moreover, pipeline stage latency is far more prone to being affected by clock edge uncertainty.

At current operating frequencies and processor integration levels, the effort to generate and distribute two non-overlapping clocks globally *or even locally* appears futile. Consequently, further discussions on level-sensitive latch-pairs will be limited to those controlled by a single-wire clock.

### 11.7 COMPARING LATCH-PAIRS AND FLIP-FLOPS

Having described the basic latch-pair, the basic flip-flop and their timing characteristics in the context of level-sensitive and edge-triggered clocking schemes, we are now in a position to compare the relative pros and cons of their principal differences.

1. The useful time afforded by latch-based level-sensitive clocking to a pipeline stage can be made insensitive to clock edge uncertainty. Edge-triggered clocking presents

a *hard edge* at cycle boundaries and hence the available time per period is reduced by skew and jitter of the clock.

2. Unbalanced stage delays of latch-based sequential networks can be equalized by slack-passing and time-borrowing. These features are absent in (most) edge-triggered clocking schemes where the pipeline stage with the longest delay determines the cycle time.
3. The latency of a flip-flop is generally comparable to or smaller than a latch-pair of the same family.
4. In contrast to level-sensitive clocking where the relative position of timing arcs with respect to the clock cannot be accurately determined, flip-flops provide a conceptual simplicity where all timing events are referenced to a single transition of the clock.
5. In comparison to edge-triggered clocking where the minimum delay restriction has to be met on cycle boundaries, this restriction for *single-wire* level-sensitive clocking must be considered twice per cycle, on the phase boundaries of the clock.

In addition, for a sequential network implemented at the same frequency by both clocking methods, there is nearly half as much logic between consecutive latches as there is between flip-flops. As a result, the relative difference between minimum and maximum delays in a phase is small, making it particularly difficult to fix hold-time violations. Thus, it is far more difficult to meet the minimum delay restriction for the *single-wire* latch-based designs as compared to its edge-triggered counterpart.

6. The flow of data is retarded every time combinational logic encounters a clocked storage element. Imagine that a pipeline stage must perform some logic and then drive a large load. With edge-triggered clocking, the clocked storage element, i.e., the flip-flop, is followed by levels of logic fanned-up properly to drive a load. In contrast, to form a pipeline stage, the level-sensitive design places a latch within the logic, as it is being fanned up. Thus, for similar latency to the edge-triggered scheme, the latch must be sized up, presenting a larger load to the clock. In general, for similar operating frequencies, the level-sensitive design must support larger clock loads when compared to its edge-triggered counterpart.

An inspection of this list indicates that the merits of edge-triggered clocking outweigh those for level-sensitive clocking. This assertion is subject to much controversy and disagreement. Making the opposite case would be equally as controversial. The purpose of this discussion, however, is that the reader be familiar with the differences between the two clocking schemes so that based on the available technology, tools and specification, make the appropriate choice of clocking for the design at hand.

## 11.8 HIGH-PERFORMANCE CLOCKED STORAGE ELEMENTS

This section covers the clocked storage elements used in three generations of the Alpha processor designs as well as those used in two generations of AMD's windows-compatible processors. The motivation in the above selections is many-fold. First, a review of literature reveals that virtually all latching elements used in other processor designs fall within categories identifiable by the above designs. Further, while providing a

somewhat historical perspective beginning in the late eighties, the review follows the design efforts of two groups of designers. The first, the Alpha group at DEC, changed the latching element in every design. Starting with a modified Svensson latch in 21064, it switched to a more basic transmission-gate level-sensitive latch on 21164. Finally, DEC designers abandoned level-sensitive clocking and chose a sense-amplifier-based edge-triggered clocking methodology. On the other hand, the K6 and K7 designs provide more continuity, adding improvements and complexity to the same family of pulsed flip-flops over two generations.

### 11.8.1 The Modified Svensson Latch

In contrast to its immediate predecessor, the VAX 6000 [4], which used a four-phase clocking scheme, the Alpha 21064 adopted the single-wire level-sensitive clocking methodology. In order to relax the latch-to-latch minimum delay restriction, the basic storage element was chosen to be a modified version of the Svensson latch-pair [5] as shown in Fig. 11.11.

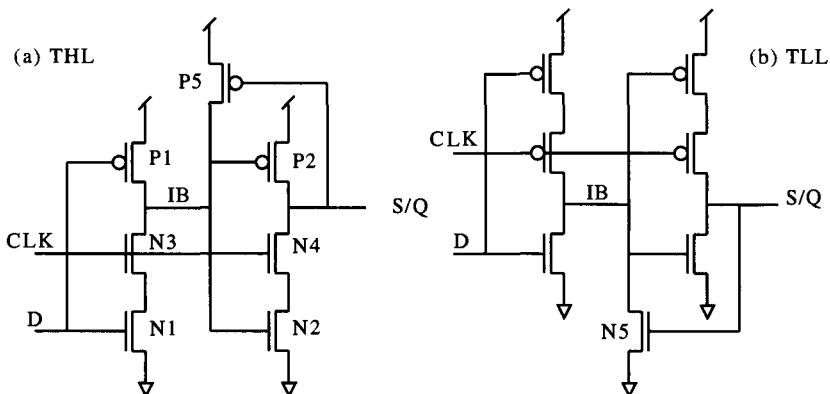


Figure 11.11 The modified Svensson latch-pair of 21064.

The operation of the Svensson THL is described as follows. When the clock is asserted, the THL is transparent and behaves as a pair of cascaded inverters, transferring the input data to the output, S/Q. Once the clock is de-asserted, the stacked pull-downs N1/N3 and N2/N4 are disabled. In the case that a '0' is stored at the output of the latch, S/Q, transistor P5 holds the intermediate node, IB at a logical '1'. The inclusion of P5 (which is the modification to the basic Svensson latch) is due to the small noise margin of the second stage of the latch with IB at a logical '1'. If IB drops below the supply rail by a threshold, P2 is activated, destroying the state of the latch. On the contrary, if S/Q stores a '1', and because the N2/N4 stack is inactive, the node IB is allowed to float at a logical '0'.

The motivation behind using the Svensson latch-pair, i.e., relaxing the minimum delay restriction, becomes apparent when one compares it to the basic latch-pair of Fig. 11.2. The Svensson latch-pair does not require the inversion of the clock signal as does the basic latch-pair. In the case of the basic latch-pair, the inversion of the clock, which can be thought of as a component of clock skew, must be accounted for in the minimum delay allowable between latching elements.

Unfortunately, this advantage of the Svensson latch-pair is to a large extent offset by the very short  $t_{CQ}$  of the  $1 \rightarrow 0$  transition. Consider the Svensson THL again. Assume that D is at a logical ‘0’ before the assertion of the clock. Consequently, IB is at a logical ‘1’ and P2 is off. Once the clock rises, the N2/N4 stack, without resistance from P2, quickly drives S/Q to ground. Recalling,  $t_{CLH\ min} = t_{SK} + (t_H - t_{CQ})$ , it can be seen that the short  $t_{CQ}$  adversely affects the minimum delay restriction and offsets the smaller  $t_{SK}$  afforded by the Svensson latches. In addition, the hold time of this latching structure is notoriously susceptible to lower clock edge rates.

Worse still, the Svensson latch-pair presents an unduly large load to the clock tree. In contrast to the basic latch-pair of Fig. 11.2 where the clock is buffered to drive one transistor of a transmission gate, the clock directly drives a pair of transistors in successive gain stages of the Svensson latch, almost doubling the clock load in comparison.

Looking at Fig. 11.11, it is seen that the latch-state is ‘held’ at the output node. In other words, it can be said that *the state node of the latch is exposed*. Moreover, one might wonder as to how the state is held when the latch is opaque: there is no feedback and no *local buffer* to hold the state pseudo-statically or dynamically. In the absence of the above, one must believe that the state is held on the gate of a potentially *distant receiver*.

While introducing a multitude of problems, *exposing the state node of a clocked storage element is an attempt to reduce its latency*. Later in this chapter, we will address issues that must be considered when the state node is exposed. Although not recommended for general use, exposing the state node must be considered in the particularly difficult timing paths of a design.

### 11.8.2 The Transmission-Gate Level-Sensitive Latch

In addressing the problems described above, the 21164 design reverted to basics, selecting a transmission-gate, level-sensitive, single-wire, dynamic latching methodology (Fig. 11.2). To lower the overhead associated with the storage element in difficult timing paths, simple logic functions were incorporated in the input and the output gates of the latch [6]. This approach effectively reduces the latching element latency to that of a single transmission gate. Including logic in the *output* gate introduces a challenging problem. Data must set up well within the transparency period of the latch such that Q transitions when the state nodes S1 and S2 are driven. If data arrive late as the latch is becoming opaque, switching of the output node and its back-coupling to the state nodes could possibly disturb the state of the latch.

### 11.8.3 The Amplifier-Based Flip-Flop

Used sparingly in earlier DEC processors as a de-skewing circuit element, variations of the amplifier-based flip-flop [7], [8] were selected as the primary storage element of the third-generation Alpha processor as well as that for DEC’s strong arm processor [9].

Figure 11.12 depicts a possible construct of the amplifier-based flip-flop. The circuit operates as follows. When the clock is low, internal nodes I and IB are pre-charged, and nodes DMR and DBMR are held low. As a result, NMOS push-pull pairs, N8/N9 and N10/N11, are both off and the state of the flip-flop is held pseudo-statically by cross-coupled inverters. On the rising edge of the clock and based on D, either IB or I transitions low, driving DMR or DBMR high. In response,

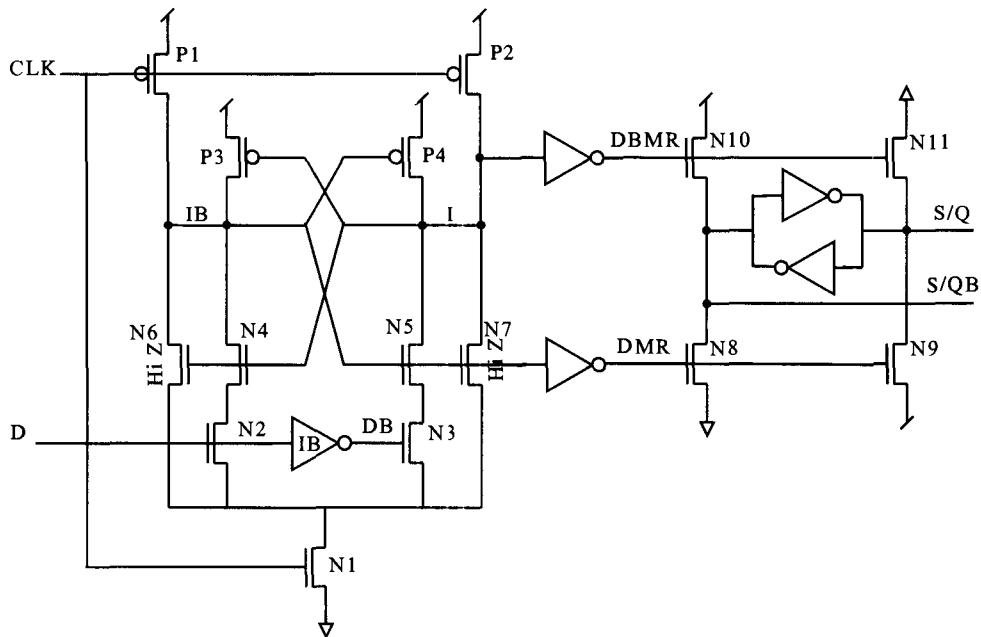


Figure 11.12 The amplifier-based flip-flop.

one of the push-pull pairs is activated. Take the case where DMR transitions; as N8 drives S/QB low, S/Q is driven high by N9. The cross-coupled inverter pair is sized to favor a logical ‘1’ in order to aid the body-affected NMOS pull-up devices, N9 and N10. On the falling edge of the clock, the circuit enters precharge.

The clocked-strobed amplifier comprising the first stage of the flip-flop is discussed here in further detail. Input data, D, and its complement, DB, gate the *source-coupled* transistors, N2 and N3, which, in turn *source-modulate* the cross-coupled transistors, N4 and N5. On the rising edge of the clock, and depending on the voltage difference between D and DB, nodes I and IB discharge at different rates. The rate difference increases in time as one path at a higher discharge rate impedes the discharge of the other path. Finally, I or IB will completely discharge while its complement fully recovers to VDD by either P3 or P4.

Once one of the source-modulated devices, N4 or N5, is turned off, further changes in D will not affect the state of the flip-flop; thus the hold time of the amplifier-based flip-flop depends on the speed at which the amplifier responds to a voltage difference at its inputs. The hold time of the flip-flop can be arbitrarily reduced by overdriving the amplifier with a large strobe transistor N1. Shorter  $t_H$ , however, implies a longer  $t_{SU}$  as the inversion delay of IB becomes more pronounced because of the reduced resolution time of the amplifier. The high-impedance transistors, N6 and N7, ensure a path to ground and hence a pseudo-static operation once the data transitions after the hold-time requirement has been met.

Due to the NMOS push-pull construction of its final stage, this latching element provides nearly identical clock-to-q latencies for S/Q and S/QB for both high and low transitions. This property finds important applications in source-synchronization and high-speed data-transfer circuits.

The amplifier-based flip-flop exhibits excellent properties of low latency and arbitrarily small hold time. However, it suffers from relatively high power dissipation due to its complementary dual-rail operation. Regardless of the previous state of the flip-flop, one or the other of complementary nodes of its first two stages will always switch twice per cycle. Moreover, the amplifier-based flip-flop needs both the data and its complement as inputs. Consequently, it is difficult to incorporate logic within the flip-flop to reduce its latching overhead.

#### 11.8.4 The Latch and Flip-Flop Hybrid Element

As explained in Section 11.6, level-sensitive designs must partition combinational gates by *latches with opposite clock assertion levels*. Without this arrangement, the sequential operation will fail once the latency of intervening logic between *like* latches falls short of the duration of the asserted clock.

Latches with the same assertion levels *can* be used to form pipeline stages provided that the transparency period of the latching element is smaller than the shortest delay present in the sequential logic. This goal is achieved if the ‘clock signal’ controlling the storage element is *a pulse derived from the assertion edge of the clock* whose duration is made to be independent of the clock period.

In this manner, a latch-pair forming a pipeline stage can be replaced by a single *pulsed* latch. Consequently, the storage-element overhead, as well as the clock load, is reduced. Further, since a single pulsed latch is required to define the boundary of a pipeline stage, the pulsed latch logically operates as a flip-flop. Last but not least, the pulsed latch does not impose a hard edge at the clock boundary as is the case with other edge-triggered storage elements, and thus, like level-sensitive latches is capable of absorbing the clock skew and jitter and can pass slack or borrow time across clock boundaries. Having the properties of both latches and flip-flops, this latching construct is known as the hybrid latch and flip-flop (HLFF) element [10].

The K6 microprocessor used the HLFF, shown in Fig. 11.13, as its primary latching element [11]. The circuit operates based on a clock-derived integrated one-shot generated by an odd number of cascaded inverters. When the clock is low, N1 and N3 are off; P1 is on and holds IB at VDD. As a result, the flip-flop is opaque and its state is held pseudo-statically. On the rising edge of the clock, transistors N1 and N3 are activated,

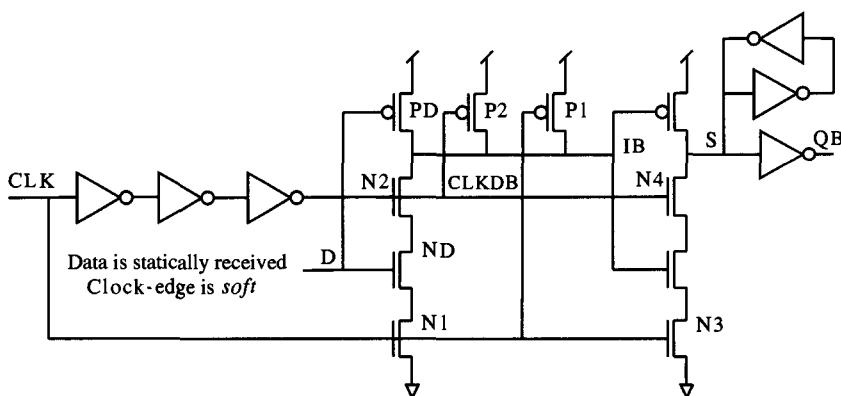
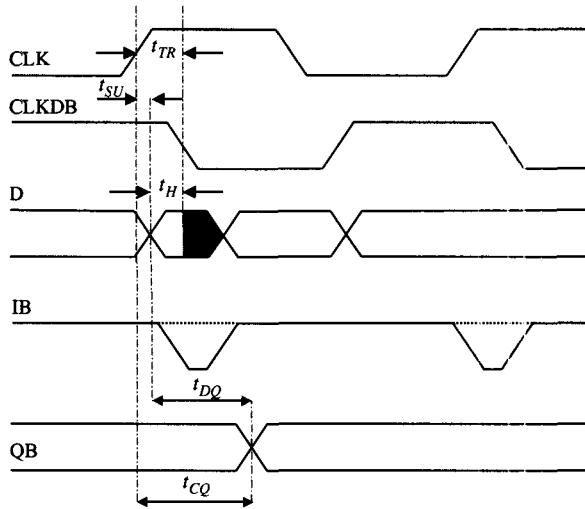


Figure 11.13 The hybrid latch flip-flop element of K6.



**Figure 11.14** Timing waveforms for the hybrid latch flip-flop element of K6.

rendering the latch transparent. In response to the assertion of CLK, some time later, CLKDB transitions low. As illustrated in Fig. 11.14, HLFF is transparent only for a brief period when *both* the CLK and CLKDB are high. It is during this time,  $t_{TR}$  (determined by the delay of the cascaded inverter chain), that the flip-flop input is sampled.

The first stage of HLFF can be used as a dynamic structure because its output IB is forced high in the opaque state, and prior to the assertion of the clock. This attribute, absent in the prior art, can be exploited to incorporate very complex logic in the flip-flop and will be explored in further detail later. The HLFF variation of Fig. 11.13, in contrast, receives the data *statically*, i.e., D gates the complementary device-pair ND/PD. This allows for both high and low transitions of data *to be captured after* the assertion of the clock during the transparency period of the latch. As a result, as depicted in Fig. 11.14, the data can have a negative set-up time,  $t_{SU}$ , relative to the assertion edge of the clock.

The ability to pass the data to the output of HLFF after the assertion of the clock makes the clock *edge look soft*. Providing a soft edge at the clock boundaries allows for the absorption of the uncertainties of the clock; thus skew and jitter are eliminated from the budget in meeting maximum timing restrictions. Consequently, Eq. (11.2) is modified for HLFF to reflect its insensitivity to the clock edge uncertainties.

$$t_{CL\ max} = t_{CY} - (t_{SU} + t_{CQ}) \quad (11.9)$$

The transparency period of HLFF also dictates its hold time. Thus, the minimum delay of a pipeline stage must adhere to the following restriction:

$$t_{CL\ min} = t_{SK} + (t_{TR} - t_{CQ}) \quad (11.10)$$

Like other flip-flops (Sect. 11.4), HLFF must be designed such that it is not sensitive to *internal race*. The falling edge of CLKDB activates P2 that drives IB toward  $V_{DD}$ . Concurrently, gated by CLKDB, N4 is driven off. A poor edge rate on CLKDB together with a strong P2 device can cause a *high-down transition* on S, which if large, results in the loss of the flip-flop state. In general a sufficiently small P2 device easily addresses this design consideration.

The K7 microprocessor uses a negative edge-triggered derivative of the pulsed flip-flop [12], [13]. In contrast to the basic HLFF, a pulse-generator circuit that is decoupled from the body of the flip-flop explicitly generates the one-shot. The motivation behind this design change was twofold: (1) to increase power efficiency, pulse-generators were shared by groups of adjacent flip-flops; and, (2) a decoupled pulse generator decreases the NMOS stack height in the flip-flop. In addition to reducing charge sharing, this facilitates the incorporation of complex logic in the first stage of the flip-flop.

Figure 11.15 shows a *dynamically received*, enabled two-way mux flip-flop used in the K7 processor. The absence of the complementary PMOS devices of the mux structure makes the first stage of the flip-flop dynamic. As is the case with dynamic logic, data cannot change once the clock is asserted. In this sense, the flip-flop poses a *hard edge* at the cycle boundary to its inputs. The ability to include complex logic in the flip-flop in many cases outweighs the benefits provided by the soft clock edge.

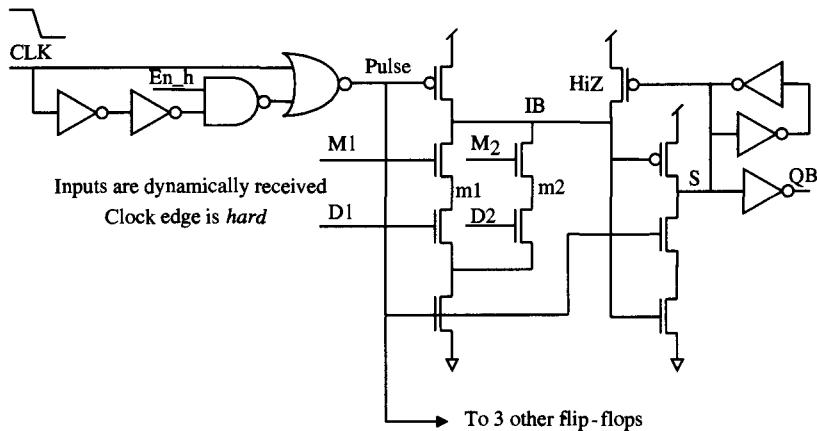


Figure 11.15 The enabled two-way mux pulsed flip-flop of K7.

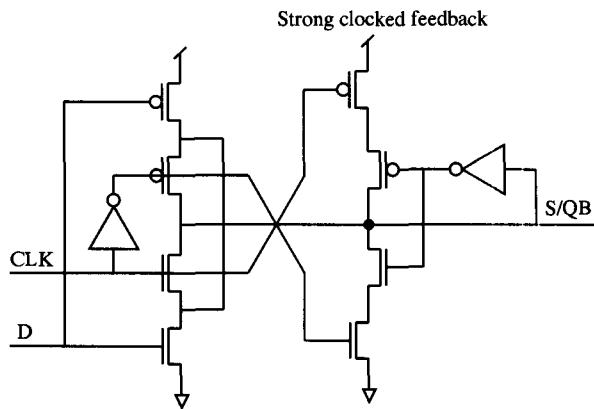
Furthermore, as can be seen from the figure, the flip-flop also incorporates an enable function: if on the falling edge of the clock  $\text{En}_h$  is low, the flip-flop recirculates its old state; if high, new data is passed to its output. The equivalent function is achieved at a higher cost in latency and area with other flip-flop topologies. The flip-flop, in such cases, must be preceded by a two-way mux controlled by  $\text{En}_h$  and having as its inputs the flip-flop output and the data input.

Because of its many advantages, this versatile clocked storage element has gained popularity in the last few years and been used in other designs [14]–[16]. The major disadvantage of the pulsed flip-flop, however, is its long hold time. Usually the effort to meet minimum delay restriction in a processor design with upwards of 50,000 latching elements requires advanced CAD tools but can still be prohibitive. Due to its negligible hold-time, the K7 processor used the classic master–slave flip-flop for over 70% of its latching elements. The master–slave flip-flop was then selectively replaced by its pulse-based counterpart when low latency and inclusion of logic were absolute requirements. The *selective* use of the pulsed flip-flop simplifies the design and has improved the operating frequency of the K7 processor by as much as 12% [13]. Interestingly, a similar methodology, i.e., the general use of the master–slave and the selective use of the pulsed flip-flops, has been adopted by IBM's S/390 and G5 processor designs [14], [15].

## 11.9 RULES FOR ROBUST DESIGN—2

1. **Exposing the state node of a latching element.** As was described in Section 11.8.1, the state node of a latching element may be exposed to reduce its latency. However, as the term ‘exposed’ implies, the element is now prone to a multitude of disturbances that can destroy its state when opaque. To reduce the sensitivity of a pseudo-static, exposed state node to such perturbations, the coupling to the node must be controlled. In addition, the state node cannot drive a transmission-gate structure.

A less restrictive approach involves the use of clocked feedback topology to closely mimic a static operation. Figure 11.16 illustrates such a circuit. Referring to the figure, it can be seen that the node S/QB is driven at all times. When transparent, data is passed to the output S/QB. When the clock is de-asserted, the clocked feedback circuit is activated, strongly holding the state captured at the falling edge of the clock. It is important to note that exposing the state node in this manner, while reducing the latency of the storage element, considerably increases the load presented to the clock network. Thus, *if the state node of a latching element is exposed, it must only drive a gate or gates, and its routing must be carefully controlled. For added immunity to noise, the state node should be statically driven at all times by employing a regenerative feedback structure that is clock-gated.*



Stack order of the feedback is to take advantage of *good* charge - sharing

**Figure 11.16** Gate-isolated THL with clocked feedback.

2. **Management of race.** The violation of minimum delay restriction is usually exhibited on silicon as a *frequency-independent* functional failure at elevated supply voltage levels. One measure of the robustness of a design is the voltage margin above the specified supply range beyond which this failure occurs. Indeed, barring reliability and power issues, robust design should allow for operating voltages well above (and below) the specified range. A design that is operational over a wide supply range can easily be extended from high-performance to low-power applications and indicates that it scales well with technology migration.

In addition to a tight control over clock skew, the management of race requires an accurate characterization of latching elements as well as the combinational logic used in the design. In order to meet the minimum delay restriction for

edge-triggered [Eq. (11.3)], or level-sensitive [Eqs. (11.6), (11.7)] clocking, the *combinational gates* between storage elements and their associated parasitics should be characterized for minimum latency as in the following:

- Devices should be skewed properly for lowered delay to account for intra-die device variations.
- Gates should be characterized for simultaneously switching inputs.
- Wire capacitance should be downscaled to account for *favorable coupling* of unrelated signals.

In addition to scaled parasitics and intra-die device variations, the characterization of the *clocked storage elements* must include the full possible range of the data and the clock edge rates. Possible values for the above parameters must be used to minimize  $t_{CQ}$  (also called the *contamination delay*) and to maximize  $t_H$ .

The above strategy is indeed conservative. It is certainly possible to relax some of these requirements if, for example, the exact coupling could be determined on a path-to-path basis; but such a task might be cumbersome and prohibitive. In order to meet the minimum delay restriction, it is, however, feasible to relax the timing budget allotted to the global clock skew by observing the following:

- There is negligible clock skew for latching elements in the same clock domain that are spatially close.
  - Skew need not be considered in the minimum delay restriction if (1) the data flows in the opposite direction of the clock, or if (2) the data flows in the same direction but cannot overtake the clock.
3. **Clock domains and the de-skewing element.** There are often times when it is difficult to accurately assess the precise uncertainty of the clocks controlling storage elements on the transmitting and receiving ends of signals that cross the clock domains. In particular, signals produced in the SYSCLK and the GCLK clock domains inherit the long-term jitter,  $t_{JL}$ , as an additional parameter to be considered in the minimum delay calculations. Under such conditions, the robust method to guard against hold-time violation is to partition a signal path by inserting a flip-flop, called a de-skewing element, which is triggered off the opposite edge of the *transmitting* clock.
  4. **Scan—observability and controllability.** The ability to observe a snapshot of the processor operation at a particular cycle or cycles and/or to control the state of a processor at a desired cycle, are invaluable tools for debugging a processor. As an example, it is possible to completely isolate a path violating the minimum delay restriction by the ability to ‘scan dump’ the state of the processor at cycles leading to and including a functional failure. The implementation of scan is achieved at a relatively small cost to latency and die area, and its benefits far outweigh the added logical complexity.

## 11.10 PERFORMANCE METRICS FOR CLOCKED STORAGE ELEMENTS

Since the processor operating frequency is more of a marketable commodity than is its architectural performance, super-pipelined designs are particularly popular. The maximum number of gates occupying a *clock tick* for such designs can be as few as

six. As a result, minimizing the latency overhead of the latching element is crucial. Furthermore, because in such designs the relative difference between the minimum and maximum delay restrictions is small, large hold time plays as a major liability.

The continuing integration of peripheral functions into the core processing unit, in addition to the popularity of mobile applications, renders the *power efficiency* of a design increasingly more important. Further, the desire to have a single design be used for both high performance and low power requires a wide operating power-supply voltage range and thus a very robust design.

The above outline suggests the somewhat orthogonal properties of low latency, tolerance to race, power efficiency, and design robustness as desirable for clocked storage elements. Let us now discuss in detail the primary metrics of performance for assessing the merits of clocked storage elements.

**Functional latency.** We define the functional latency of a clocked storage element as the data-to-output delay associated with the necessary minimum number of gates to achieve *just* the function of latching data. In this context, it is assumed that all other resources are abundant: no restrictions are placed on power consumption, clock load, or hold time, and that the state node may be exposed. Furthermore, with the same line of reasoning, the functional latency of an element is measured independent of its load, i.e., in the self-loaded configuration, and dependent only on its internal workings.

**Inclusion of logic.** The ability to include logic with a latching element serves to reduce its latency or the overhead associated with the latching function itself. The incorporation of complex logic in general leads to larger hold times that limit the application of such latching elements to the few paths with particularly difficult timing constraints.

**Power efficiency.** As the power-delay product of a circuit is load-dependent, the efficiency of a latching element is defined as its *power-delay product normalized to the output load* that it is optimally designed to drive ( $PDP/C_L$ ). One must remember that the load presented by a storage element to the clock network can be considered as yet another viable performance metric. In this text, however, the clock load is measured in the context of the power efficiency of a latching element: the power for switching clock-driven transistors of an element is included in power calculations.

**Race tolerance.** With the increasing operating frequencies and integration levels, the importance of race management cannot be overstated. Recalling Eqs. (11.3), (11.6) and (11.7), one observes that hold time,  $t_H$ , and the contamination delay,  $t_{CQ\ min}$ , of a latching element both affect the minimum delay restriction. As a result, we define the *race tolerance metric* to be  $(t_{CQ\ min} - t_H)$ .

**Design robustness.** Storage element topologies cannot be accurately compared unless their design adheres to similar degrees of robustness and testability as outlined in Sections 11.5 and 11.9. For example, latching elements must be designed for comparable tolerance to data input and output perturbations. In particular, an element with a protected state node cannot be fairly compared with one whose state node is exposed.

Table 11.1 compares the relative merits of the master-slave, the amplifier-based, and the pulse-based flip-flops discussed in Section 11.8. All elements were designed with similar design robustness and simulated under identical parametric conditions. As can be seen, the pulse-based flip-flop provides superior latency and power efficiency. In contrast, it suffers from poor race tolerance. The master-slave counterpart has superior race tolerance but poor latency and power efficiency. The amplifier-based flip-flop ranks in the middle in these categories. Unlike the master-slave and the amplifier-based elements, the pulse-based topology is capable of incorporating complex logic.

**Table 11.1** Comparison of Clocked Storage Elements

Element Type	$t_{DQ}$ (ps)	$t_{CQ} - t_H$ (ps)	$PDP/C_L$ (mV) <sup>2</sup>	Complex Logic
Master-slave	169	82	0.19	No
Amplifier-based	110	32	0.18	No
Pulse-based	94	2	0.15	Yes

## 11.11 LATCHING ELEMENTS FOR DYNAMIC CIRCUITS

The discussion of clocked storage elements, thus far, has been limited to those used with static combinational logic. In this section, storage elements conducive to dynamic logic will be covered. Although latching elements used in static logic can also be employed by dynamic logic, there are latching elements that in particular facilitate dynamic circuit operations and allow for seamless transitions between the dynamic and the static circuit domains. A dynamic structure is itself considered as a form of a storage element; in this chapter, however, the conversion circuits between the dynamic and static domains are specifically coined as storage elements for dynamic logic.

If the inputs of a dynamic structure are de-asserted when it is being precharged (or is opaque), and are *allowed to transition only in one direction* once the structure becomes transparent, then it is said that they are *monotonic*. In contrast to the positive set-up time required for static signals forming the inputs of dynamic logic, monotonic signaling allows for the inputs to set up before, or *anytime after* a dynamic structure has become transparent. This property, in addition to the fact that the outputs of dynamic circuits are also of the monotonic form, lead to a class of dynamic circuits tolerant to clock skew [17]. In general dynamic circuits require the availability of both the *true and complement of a signal in their monotonic form*. Thus, dynamic structures that produce *dual-rail* outputs can be *cascaded in a domino fashion* to form complete pipeline stages.

A static-to-dynamic converter (SDC) circuit receives a static input D and generates an output which is the dual-rail monotonic version of its input. The output can then gate a cascade of dynamic circuits. Figure 11.17 shows the pulse-based dual-rail SDC circuit. The circuit operates as follows. When the clock is low, nodes I and IB are precharged and output signals DMR and DBMR are de-asserted. On the rising edge of the clock and based on the input data, either DMR or DBMR will monotonically rise. The state is held for as long as the clock is high. Once the clock falls, the circuit enters precharge and its outputs are de-asserted.

Since monotonic signals are only valid for a time less than a clock period (usually a phase), a dynamic circuit output must be converted to its static form so that it can be captured and transferred to a pipeline stage with static logic. A dynamic-to-static converter (DSC) circuit processes its monotonic inputs to generate static outputs to be received by a flip-flop.

*Dual-rail* monotonic signals are readily converted to their static form using RS flip-flops. There are, however, circuit types in dynamic logic where it is difficult to generate the dual-rail version of a signal. A CAM structure comprising wide dynamic OR circuits serves as an example. A *single-rail* monotonic signal can be converted to its static form by using the dynamic-to-static converter circuit of Fig. 11.18. Also known as

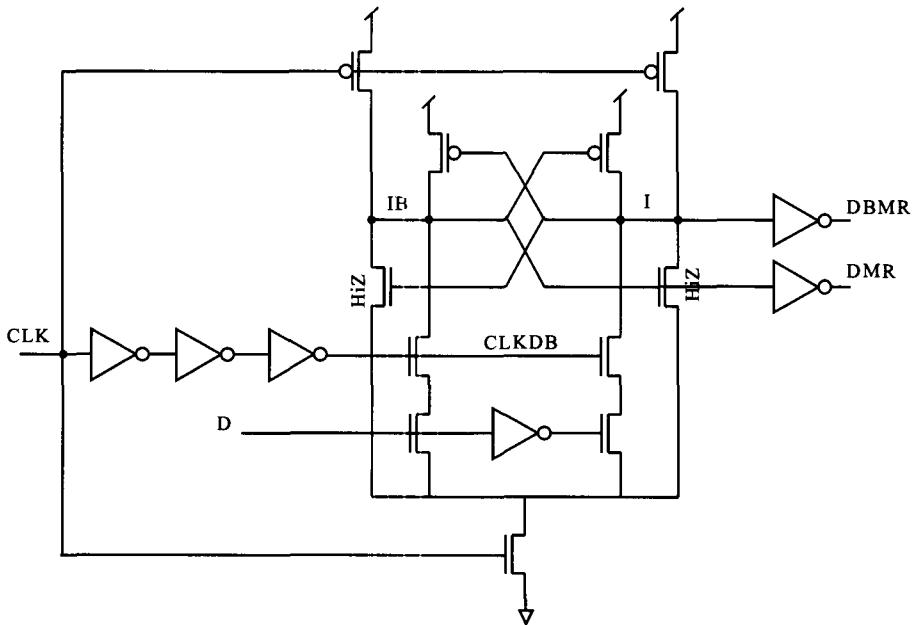


Figure 11.17 Pulse-based static to dual-rail dynamic converter.

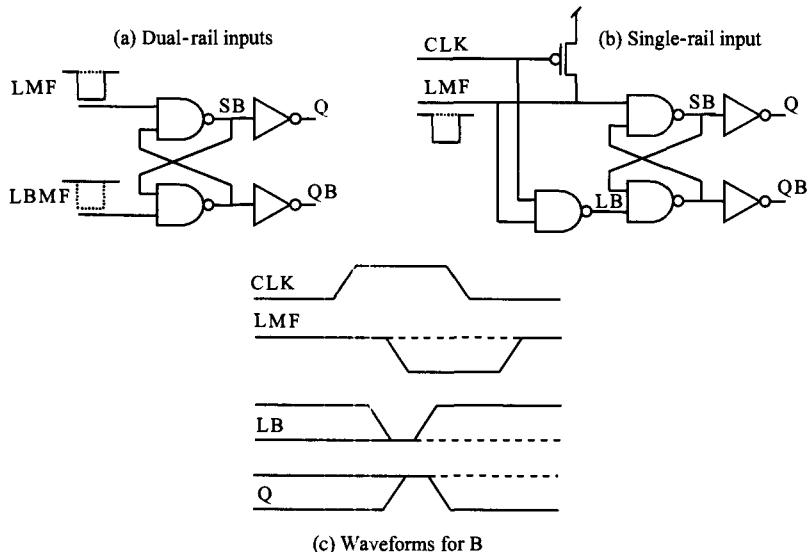


Figure 11.18 DSC Circuits with dual- and single-rail monotonic inputs.

a *glitch latch*, the DSC responds to a pulse or a glitch at its input. The circuit operates as follows. When the clock is low, LMF and LB are both precharged to a logical '1', and SB holds state statically. On the rising edge of the clock, the output of the NAND gate, LB, is driven toward ground, SB falls and Q rises. Based on LMF, which may remain high or monotonically fall during the assertion of the clock, SB will remain low or will

transition (back) to  $V_{DD}$ . When the clock falls, the structure becomes opaque and will hold its state for as long as the clock remains low.

## 11.12 RECOMMENDATIONS AND CONCLUSION

To recapitulate the rather exhaustive treatment of clocked storage elements, the chapter concludes by proposing the storage element strategy that the author finds most suitable for the processor designs in the 1 GHz frequency range.

A single-wire clock is distributed globally; it is locally received, buffered, and qualified. The clocking strategy is edge-triggered. A general-purpose flip-flop structure with high power efficiency and/or solid race tolerance is selected to be used in the majority of pipeline stages which easily meet the maximum delay restriction. A special-purpose flip-flop with low functional latency and the ability to incorporate complex logic is designed to replace the general-purpose flip-flops where it is not possible to meet circuit timing.

Scientific enquiry necessitates an impartial study of a subject. Although desirable, it is difficult to remain unbiased in subjects such as the latching elements that are as much based on knowledge, as they are art forms. The author hopes that his treatment of the basic concepts in this chapter is impartial. Also, in drawing comparisons between edge-triggered and level-sensitive clocking as well as those among storage elements, merits, and not groundless pontifications have formed the basis for the assignment of value. However, critical analysis of any topology based on the special requirements of a design is encouraged in making design selections.

## REFERENCES

- [1] B. Razavi, Ed. *Monolithic Phase Locked Loops and Clock Recovery Circuits*. IEEE Press, New York, 1996, pp. 3–4.
- [2] V. R. von Kaenel, “A High-speed, Low-power Clock Generator for a Microprocessor Application,” *IEEE Journal of Solid State Circuits*, vol. 33, no. 11, Nov. 1998, pp. 1634–1639.
- [3] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, NJ, 1996, pp. 351–353.
- [4] R. W. Badeau, R. I. Bahar, et al., “A 100-MHz Macropipelined VAX Microprocessor,” *IEEE Journal of Solid State Circuits*, vol. 27, no. 11, Nov. 1992, pp. 1585–1598.
- [5] D. Dobberpuhl, R. Witek, et al., “A 200-MHz 64-b Dual-issue CMOS Microprocessor,” *IEEE Journal of Solid State Circuits*, vol. 27, no. 11, Nov. 1992, pp. 1555–1565.
- [6] B. J. Benschneider, A. J. Black, et al., “A 300-MHz, 64-b, Quad-issue CMOS RISC Microprocessor,” *IEEE Journal of Solid State Circuits*, vol. 30, no. 11, Nov. 1995, pp. 1203–1214.
- [7] W. C. Madden and W. J. Bowhill, “High Input Impedance, Strobed CMOS Differential Sense-amplifier,” United States Patent 4,910,713.
- [8] B. A. Gieseke, R. A. Conrad, et al., “Push-pull Cascode Logic,” United States Patent 5,023,480.
- [9] J. Montanaro, R. Witek, et al., “A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor,” *IEEE Journal of Solid State Circuits*, vol. 31, no. 11, Nov. 1996, pp. 1703–1714.

- [10] H. Partovi, R. Burd, et al., "Flow-through Latch and Edge-triggered Flip-flop Hybrid Elements," *1996 IEEE International Solid State Circuits Conference Digest of Technical Papers*, 1996, pp. 138–139.
- [11] D. Draper, M. Crowley, et al., "Circuit Techniques in a 266-MHz MMX-Enabled Processor," *IEEE Journal of Solid State Circuits*, vol. 32, no. 11, Nov. 1997, pp. 1650–1664.
- [12] S. Hesley, V. Andrade, et al., "A 7th-generation X86 Microprocessor," *1999 IEEE International Solid State Circuits Conference Digest of Technical Papers*, 1999, pp. 92–93.
- [13] A. Scherer, M. Golden, et al., "An Out-of-order Three-way Superscalar Multimedia Floating Point Unit," *1999 IEEE International Solid State Circuits Conference Digest of Technical Papers*, 1999, pp. 94–95.
- [14] C. F. Webb, C. J. Anderson, et al., "A 400-MHz S/390 Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 32, no. 11, Nov. 1997, pp. 1665–1675.
- [15] G. Northrop, R. Averill, et al., "600 MHz G5 S/390 Microprocessor," *1999 IEEE International Solid State Circuits Conference Digest of Technical Papers*, 1999, pp. 88–89.
- [16] F. Klass, C. Amir, et al., "A New Family of Semidynamic and Dynamic Flip-flops with Embedded Logic for High Performance Processors," *IEEE Journal of Solid State Circuits*, vol. 34, no. 5, May 1999, pp. 712–716.
- [17] D. Harris and M. A. Horowitz, "Skew-tolerant Domino Circuits," *IEEE Journal of Solid State Circuits*, vol. 32, no. 11, Nov. 1997, pp. 1702–1711.

John George Maneatis  
*JGM Enterprises*

## 12.1 INTRODUCTION

Phase-locked loops, a set of circuits that include delay-locked loops, have found many applications within the realm of microprocessors and digital chips in the past 15 years. These applications include clock frequency synthesis, clock de-skewing, and high bandwidth chip interfaces. A typical chip interface application is shown in Fig. 12.1 in which two chips synchronously send data to one another. To achieve high bandwidth, the data rate must be maximized with minimum data latency. Achieving this objective requires careful control over system timing in order to guaranteeing that setup and hold times are always satisfied.

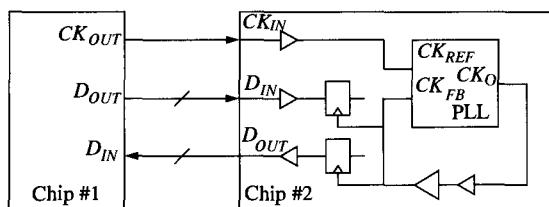


Figure 12.1 Typical chip interface.

Let's consider the requirements for receiving data by Chip #2. Chip #1 transmits this data synchronously along with a clock signal. Chip #2 would need to buffer this clock signal to drive all of the input latches and use it to sample the data. Buffering the clock signal will introduce a delay that will vary with process and environmental factors. The setup and hold time window for the input latches will then be shifted from the input clock edge by this varying delay amount. Such a delay can make it very difficult to ensure that setup and hold times are always satisfied as the data rate is increased and this delay becomes a larger fraction of the clock cycle.

To alleviate the situation, it is desirable to eliminate this clock distribution delay and center the setup and hold time window on the input clock edge which would remove any uncertainty in the window position relative to the clock signal. Such an approach also has the added benefit of avoiding the necessity for delay padding on the data wires to compensate for the clock distribution delay which would increase the latency. It is also desirable to be able to multiply the frequency of the clock signal for use in the chip core so that the core logic can run with a higher clock frequency than available from the interface. These objectives can all be accomplished with a phase-locked loop (PLL) [1], [2].

The PLL generates an on-chip clock from the input clock to drive the clock distribution network and ultimately all of the latches and registers on the chip. By sensing the clock at the input of the receiving latches and adjusting its output phase until this latch clock aligns with the input clock, the PLL is able to subtract out the clock distribution delay and make it appear as though the input clock directly connects to all of the latches. The result is that the setup and hold time window is centered on the input clock edge with no process or environmental dependencies. The amount of setup and hold time can also be controlled relative to the clock cycle by centering the setup and hold time window relative to a different part of the clock cycle.

While PLLs may seem to be the universal cure to all clock generation and interface problems, they do not come without problems of their own. PLLs can introduce time-varying offsets in the phase of the output clock from its ideal value as a result of internal and environmental factors. These time-varying offsets in the output clock phase are commonly referred to as jitter. Jitter can have disastrous effects on the timing of an interface by causing setup and hold time violations that lead to data transmission errors.

Jitter was not a significant issue when PLLs were first introduced into digital IC interfaces. The techniques employed were fairly effective in addressing the jitter issue. However, designers often reapply those same PLL design techniques even though the nature of the problem has changed. IC technologies have improved, leading to decreasing cycle times. The number of I/O pins and I/O data rates have increased leading to an increasing on-chip noise environment. An increasing aggressiveness in I/O system design has lead to a decreasing tolerance for jitter. The result is that PLL output jitter has increased while jitter tolerances have decreased, leading to significant jitter problems.

This chapter focuses on the analysis and design of PLLs for interface applications in digital ICs with particular emphasis on achieving low output jitter. It begins by considering two basic phase-locked loop architectures in Section 12.2. Sections 12.3 and 12.4 perform a stability analysis for each architecture in order to gain insight into the various design trade-offs and then present a comprehensive design strategy to establish the various loop parameters for each architecture. More advanced PLL architectures are briefly discussed in Section 12.5. Section 12.6 shifts gears to review the causes of output jitter in PLLs and examines circuit level techniques for reducing its magnitude. Circuits issues related to the implementation of the various PLL loop components are presented in Section 12.7. This chapter concludes with a brief discussion of self-biased techniques in Section 12.8 that can be used to eliminate the process and environmental dependencies within the PLL designs.

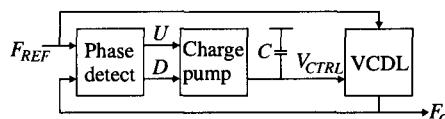
## 12.2 PLL ARCHITECTURES

The basic operation of the PLLs considered in this chapter is the adjustment of the phase of the output so that no phase error is detected between the reference and feedback inputs. There are a number of ways PLLs can be structured to accomplish this objective. Their structure can be classified based on how they react to phase errors and how they control the phase of the output. This chapter focuses only on PLLs that integrate the phase error in the loop filter using charge pumps [3]. Charge pump PLLs have the property that in the locked state the detected phase error is theoretically zero.

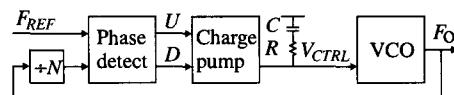
In general, PLLs can control their output phase directly by delaying the reference signal or indirectly by changing the output frequency. The first is commonly referred to

as a *delay-locked loop* (DLL) since it actually locks the delay between the reference input and the feedback input to some fraction of the reference input period. The second is referred to as a VCO-based PLL or simply as a PLL since it controls the frequency of a voltage-controlled oscillator (VCO) generating the output such that the feedback input is in phase with the reference input.

Figure 12.2 shows the general structure of a DLL. It is composed of a phase detector, charge pump, loop filter, and voltage-controlled delay line (VCDL). The negative feedback in the loop adjusts the delay through the VCDL by integrating the phase error that results between the periodic reference and delay line output. When in lock, the VCDL delays the reference input by a fixed amount to form the output such that the phase detector detects no phase error between the reference and feedback inputs. The clock distribution network, although not shown in the figure, is between the DLL output and the feedback input. Functionally, it can be considered as part of the VCDL.



**Figure 12.2** Typical DLL block diagram (clock distribution omitted).



**Figure 12.3** Typical PLL block diagram (clock distribution omitted).

Figure 12.3 shows the general structure of a PLL. It is composed of a phase detector, charge pump, loop filter, bias generator, and voltage-controlled oscillator (VCO). Two key differences from the DLL are that the PLL contains a VCO instead of a VCDL and, as will be discussed below, requires a resistor in the loop for stability. The negative feedback in the loop adjusts the VCO output frequency by integrating the phase error that results between the periodic reference input and the divided VCO output. When in lock, the VCO generates an output frequency and phase such that the phase detector detects no phase error between the reference and feedback inputs. With no phase error between the reference and feedback inputs, the inputs must also be at the same frequency. If a frequency divider, which divides by  $N$ , is inserted between the PLL output and feedback input, the PLL output will be  $N$  times higher in frequency than the reference and feedback inputs, thus allowing the PLL to perform frequency multiplication.

The difference in loop structure between a DLL and a PLL gives rise to different properties and operating characteristics. DLLs tend to have short locking times and relatively low tracking jitter, but generally do not support frequency multiplication or duty cycle correction, have limited delay ranges, and require special lock reset functions. PLLs have unlimited phase ranges, support frequency multiplication and duty cycle correction, do not require special lock reset functions, but usually have longer lock times and higher tracking jitter. DLLs are less complex than PLLs from a loop architecture perspective, but are generally more complex from a design and system integration perspective.

### 12.2.1 Loop Components

PLLs and DLLs share many common building blocks. These building blocks are the phase detector, charge pump, loop filter, voltage-controlled delay line, and voltage-controlled oscillator.

A phase detector, also known as a phase comparator, compares two input signals and generates “UP” and “DN” output pulses that represent the direction and magnitude of the input phase error. There are many types of phase detectors; they differ in how they sense the input signals, what target input phase difference would cause them to detect no phase error, and how the phase error is represented in the output pulses.

For simplicity, we will initially only consider phase-frequency detectors. These detectors have the property that they are only rising or falling edge sensitive and, for each pair of input reference and feedback edges, produce a single pulse at either the UP or DN output, depending on which edge arrives first, with a duration equal to the time difference between the two edges or, equivalently, the input phase difference. When the reference and feedback edges arrive at the same time for zero input phase difference, the phase detector will effectively generate no UP or DN pulses. However, in actual implementation, the input phase difference may be represented by the phase detector as the difference between the pulse widths of the UP and DN outputs, where both are always asserted for some minimum duration in order to guarantee that no error information is lost due to incompletely rising pulses as the input phase difference approaches zero.

A charge pump, connected to the phase detector, sources or sinks current for the duration of the UP and DN pulses from the phase detector. The net output charge is proportional to the difference between the pulse widths of the UP and DN outputs. The charge pump drives the loop filter which integrates and filters the charge current to produce the control voltage. The control voltage drives a voltage-controlled delay line (VCDL) in a DLL which generates a delay proportional to the control voltage, or drives a voltage-controlled oscillator (VCO) in a PLL which generates a frequency proportional to the control voltage.

## 12.3 DELAY-LOCKED LOOPS

Before we consider a detailed analysis of the loop dynamics of a DLL, it is instructive to consider the control dynamics from a qualitative perspective as the loop approaches lock. Figure 12.4 illustrates the waveforms of signals and quantities inside a DLL during this locking process. Initially, the DLL is out of lock as the reference and output edges are not aligned.

Since the first output edge arrives before the first corresponding reference edge, the phase detector outputs a pulse at the UP output equal in duration to this phase error. A pulse at the UP output indicates that the delay needs to be increased. The charge pump generates an output charge proportional to the phase error, which increases the control voltage and thus the delay of the VCDL. After several cycles, the phase error is corrected.

Since the error is sampled only once per cycle, the DLL is a sampled system as represented by the phase error impulses. However, if we limit the response time of the system to be a decade below the operating frequency, we can make a continuous time approximation. This approximation assumes that the phase errors are determined continuously as represented by the dashed line. Such a bandwidth limit will be required anyway to guarantee stability.

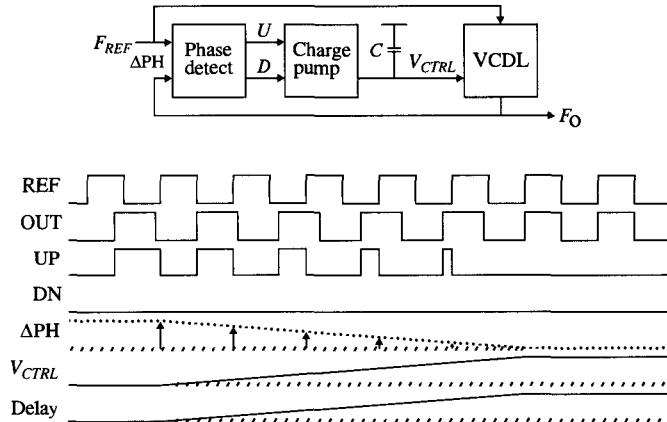


Figure 12.4 DLL locking waveforms.

Since the magnitude of the delay correction per cycle is proportional to the detected phase error, the phase error, control voltage, and delay should change with an exponential decay toward their final values, rather than linearly as shown, for simplicity, in the figure. Also, it should be noted that there are different ways of configuring the charge pump in the DLL. Some DLLs, for example, output a fixed charge independent of the size of the phase error. This type of charge pump converts the DLL into a nonlinear system and as such will not be considered in the following DLL analysis.

### 12.3.1 DLL Frequency Response

More insight into DLL design issues can be gained by determining the frequency response of the DLL. This frequency response can be derived with a continuous time approximation, where the sampling behavior of the phase detector is ignored. This approximation holds for response bandwidths that are a decade or more below the operating frequency. This bandwidth constraint is also required for stability due to the reduced phase margin near the higher-order poles that result from the delay around the sampled feedback loop.

Because the loop filter integrates the phase error, the DLL has a first-order closed-loop response. The response could be formulated in terms of input phase and output phase. However, this set of variables is incompatible with the continuous time analysis since the sampled nature of the system must be considered. A better set of variables is input delay and output delay. The output delay is the delay between the reference input and the DLL output or, equivalently, the delay established by the VCDL. The input delay is some fraction of the input clock period as determined by the phase detector. It is typically one, one half, or one quarter of the input clock period.

The output delay, \$D\_O(s)\$, is related to the input delay, \$D\_I(s)\$, by

$$D_O(s) = (D_I(s) - D_O(s)) \cdot F_{REF} \cdot I_{CH} / (s \cdot C) \cdot K_{DL}$$

where \$F\_{REF}\$ is the reference frequency (Hz), \$I\_{CH}\$ is the charge pump current (A), \$C\$ is the loop filter capacitance (F), and \$K\_{DL}\$ is the VCDL gain (s/V). The product of the delay difference and the reference frequency is equal to the fraction of the reference period in which the charge pump is activated. The average charge pump output current is equal

to this fraction times the peak charge pump current. The output delay is then equal to the product of the average charge pump current, the loop filter transfer function, and the delay line gain.

The closed-loop response is then given by

$$D_O(s)/D_I(s) = 1/(1 + s/\omega_N)$$

where  $\omega_N$ , defined as the loop bandwidth (rad/s), is given by

$$\omega_N = I_{CH} \cdot K_{DL} \cdot F_{REF}/C$$

This response is of first order with a pole at  $\omega_N$ . Thus, the DLL acts as a single-pole low-pass filter to changes in the input reference period with cutoff frequency  $\omega_N$ . The delay between the reference and feedback signal will be a filtered version of a set fraction of the reference period. It is unconditionally stable as long as the continuous time approximation holds or, equivalently, as long as  $\omega_N$  is a decade below  $\omega_{REF}$ . As  $\omega_N$  increases above  $\omega_{REF}/10$ , the delay in sampling the phase error will become more significant and will begin to undermine the stability of the loop.

### 12.3.2 DLL Design Strategy

With an understanding of the DLL frequency response, we can consider how to structure the loop parameters to obtain desirable loop dynamics. Using the bandwidth results from the DLL frequency response and limiting it to a decade below the reference frequency, we can determine the constraints on the charge pump current, VCDL gain, and loop filter capacitance as

$$\omega_N/F_{REF} := I_{CH} \cdot K_{DL}/C \leq \pi/5$$

The VCDL also must be structured so that it spans adequate delay range to guarantee lock for all operating, environmental, and process conditions. Also, special measures may be required to guarantee that the DLL reaches lock after being reset. These measures depend on the specific structure of the DLL. Typically, the VCDL delay is set to its minimum delay and the state of the phase detector is reset. However, for some DLLs, more complicated approaches may be required.

### 12.3.3 Alternative DLL Structures

The complexity of designing a DLL is not so much in the control dynamics as it is in the underlying structure. While the DLLs discussed in this chapter are analog based using VCDLs with analog control, many other approaches are possible that utilize different amounts of analog and digital control. These approaches can circumvent the problems associated with limited delay ranges and reaching lock. One possible structure is a rotating phase DLL that digitally selects and optionally interpolates with analog or digital control between intermediate output phases from a VCDL or VCO phase-locked to the clock period [4]. A related structure interpolates with analog control between quadrature phases generated directly from the clock signal [5]. Another even simpler structure with reduced jitter performance digitally selects intermediate outputs from an inverter chain based delay line [6]. While digital control provides more flexibility, analog control requires less power and area.

## 12.4 PHASE-LOCKED LOOPS

Like a DLL, a PLL aligns the phase of the output to match the input. The DLL accomplishes this task by appropriately delaying the input signal. The PLL accomplishes this task by controlling an oscillator to match the phases of the input signal. The control for the PLL is more indirect, which requires it to have the resistor in the loop filter for stability.

Consider the typical PLL shown in Fig. 12.3 as it starts out from an unlocked state with a VCO frequency that is relatively close to but slightly higher than the reference frequency. To help understand the function of the resistor in the loop filter, let's first assume that it is zero valued making the loop filter equivalent to that of the DLL. Initially, the PLL is out of lock as the reference and feedback edges are not aligned. With the first feedback edge arriving before the first corresponding reference edge, the phase detector outputs a pulse at the DN output equal in duration to this phase error. A pulse at the DN output indicates that the VCO frequency needs to be reduced. The charge pump generates an output charge proportional to the phase error, which reduces the control voltage and thus the VCO frequency.

In order to reduce the phase error, the feedback edges need to arrive later and later with respect to the reference edges or, equivalently, the VCO frequency must be reduced below the reference frequency. After several cycles, the phase error is reduced to zero, but the VCO frequency is now lower than the reference frequency. This causes the feedback edges to begin to arrive later than the corresponding reference edges leading to the opposite error condition from which the loop started. The loop then begins to increase the VCO frequency above the reference frequency to reduce the phase error, but at the point when the phase error is zero, the VCO frequency is now higher than the reference frequency. Thus, in the PLL with a zero-valued resistor, the phase error will oscillate freely around zero, which represents unstable behavior.

This unstable behavior can be circumvented by adding an extra frequency adjustment that is proportional to the phase error and is therefore applied only for the duration of the phase error. This proportional control allows the loop to adjust the VCO frequency past the reference frequency in order to reduce the phase error without the frequency difference persisting when the phase error is eliminated. When the phase error reaches zero and the extra adjustment is reduced to zero, the VCO frequency should match the reference frequency leading to a stable result. This proportional control can be implemented by adding a resistor in series with the loop filter capacitor. This resistor converts the charge pump current, which is proportional to the phase error, into an extra control voltage component which is added to the control voltage already integrated on the loop filter capacitor.

From another perspective this resistor dampens out potential phase and frequency overshooting. The amount of damping depends on the value of the resistor. Clearly, with zero resistance there will be no damping and the loop will be unstable as the output phase will oscillate forever around zero phase difference. As the resistor value is increased, the loop will become increasingly less underdamped as the oscillations will decay to zero at an increasing rate. For some resistor value, the loop will become critically damped as the oscillations will go away entirely and the phase will approach zero without overshooting. As the resistor value is increased further, the loop becomes overdamped as the phase initially approaches zero rapidly then slows down, taking a long time to reach zero.

The overdamped behavior results when the damping is so high that it creates a large frequency difference between the VCO and reference that initially drives the phase error rapidly toward zero. However, this added frequency difference goes away when the phase error approaches zero. The VCO frequency that results from the voltage across the loop filter capacitor may still be different from the reference frequency. Unfortunately, the phase error has been reduced substantially so that there is little charge pump current to change the voltage on the loop filter capacitor very quickly. The phase will change rapidly to the point where the resultant phase error generates a proportional frequency correction that makes the VCO frequency match the reference frequency. As the proportionality constant, or, equivalently, the resistance, is increased, the rate at which the phase changes will also increase and the phase error after the initial phase change will decrease. However, as the initial phase error is reduced, the amount of time required to eliminate the phase error will increase as the charge pump current will also decrease.

### 12.4.1 PLL Frequency Response

The different types of damping behavior can be quantified more carefully by deriving the frequency response of the PLL. As with the DLL, the frequency response of the PLL can be analyzed with a continuous time approximation for bandwidths a decade or more below the operating frequency. This bandwidth constraint is also required for stability due to the reduced phase margin near the higher-order poles that result from the delay around the sampled feedback loop. Because the loop filter integrates the charge representing the phase error and the VCO integrates the output frequency to form the output phase, the PLL has a second-order closed loop response.

Considerable insight can be gained into the design of the PLL by first considering its open loop response. This response can be derived by breaking the loop at the feedback input of the phase detector. The output phase,  $P_O(s)$ , is related to the input phase,  $P_I(s)$ , by

$$P_O(s) = P_I(s) \cdot I_{CH} \cdot (R + 1/(s \cdot C)) \cdot K_V / s$$

where  $I_{CH}$  is the charge pump current (A),  $R$  is the loop filter resistor (ohms),  $C$  is the loop filter capacitance (F), and  $K_V$  is the VCO gain (Hz/V). The open loop response,  $H(s)$ , then given by

$$H(s) = P_O(s) / P_I(s) = I_{CH} \cdot K_V \cdot (1 + s \cdot R \cdot C) / (s^2 \cdot C)$$

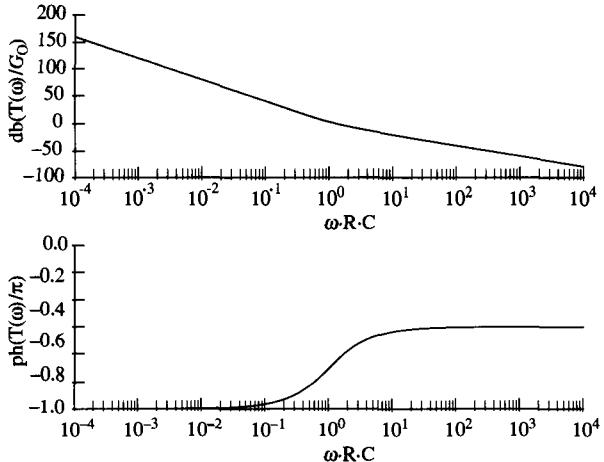
The loop gain,  $T(s)$ , which is the product of the gain through the forward path,  $H(s)$ , and the gain through the feedback path,  $1/N$ , is given by

$$T(s) = H(s)/N$$

The normalized loop gain magnitude and phase plots for the PLL are shown in Fig. 12.5. At low frequencies, the loop gain drops at 40 dB per decade where the phase is at  $-180$  degrees since there are two poles at zero frequency. The zero caused by the resistor in the loop filter is at frequency  $1/(R \cdot C)$  and causes the loop gain at higher frequencies to only drop at 20 dB per decade and the loop phase to “decrease” to  $-90$  degrees which makes it possible to stabilize the loop.

The plotted loop gain magnitude is normalized with the gain normalization factor,  $G_O$ , given by

$$G_O = R^2 \cdot C \cdot I_{CH} \cdot K_V / N$$



**Figure 12.5** PLL loop gain magnitude and phase (without  $C_2$ ).

The value of this factor will set the frequency at which the loop gain is unity. This frequency is significant because it determines the phase margin, which is a measure of the stability and the amount of damping for the PLL system. The phase margin is measured as 180 degrees or  $\pi$  radians plus the loop gain phase at the unity gain frequency or, equivalently, the frequency where the loop gain magnitude is unity. The unity gain level on the plot is the inverse of the gain normalization factor. There is no phase margin at unity gain frequencies below  $0.1/(R \cdot C)$  because the loop gain phase is about  $-180$  degrees. The phase margin gradually increases with increasing unity gain frequency as a result of the zero at frequency  $1/(R \cdot C)$ .

The loop is critically damped with a phase margin of 76 degrees, corresponding to a normalized loop gain magnitude of 0.25, a gain normalization factor of 4, and a unity gain frequency of  $4.12/(R \cdot C)$  (rad/sec). The loop will be underdamped for smaller phase margins and overdamped for greater phase margins.

The closed loop response can be derived from the open loop response by considering the feedback signal. In the closed loop system, the output phase,  $P_O(s)$ , is related to the input phase,  $P_I(s)$ , by

$$P_O(s) = (P_I(s) - P_O(s)/N) \cdot H(s)$$

where  $N$  is the feedback clock divider value. The closed loop response is then given by

$$\begin{aligned} P_O(s)/P_I(s) &= 1/(1/N + s/H(s)) \\ &= N \cdot \frac{1 + s \cdot C \cdot R}{1 + s \cdot C \cdot R + s^2/(I_{CH}/C \cdot K_V/N)} \end{aligned}$$

or, equivalently, by

$$P_O(s)/P_I(s) = N \cdot \frac{1 + 2 \cdot \zeta \cdot (s/\omega_N)}{1 + 2 \cdot \zeta \cdot (s/\omega_N) + (s/\omega_N)^2}$$

where  $\zeta$ , defined as the damping factor, is given by

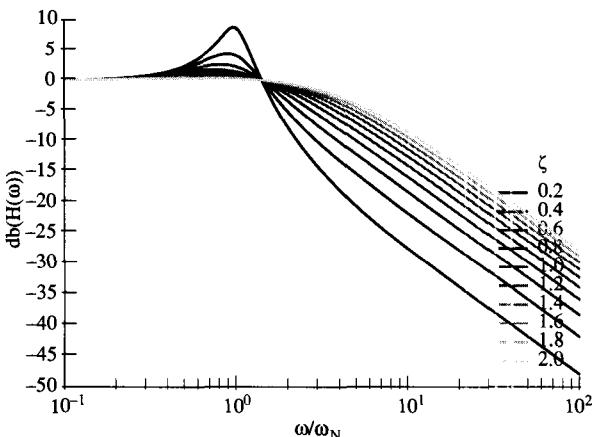
$$\zeta = 1/2 \cdot \sqrt{1/N \cdot I_{CH} \cdot K_V \cdot R^2 \cdot C}$$

and  $\omega_N$ , defined as the loop bandwidth (rad/s), is given by

$$\omega_N = 2 \cdot \zeta / (R \cdot C)$$

The loop bandwidth and damping factor completely characterize the closed loop response. The PLL is critically damped with a damping factor of one and overdamped with damping factors greater than one.

The closed loop frequency response of the PLL for different values of  $\zeta$  and for frequencies normalized to  $\omega_N$  is shown in Fig. 12.6. This plot shows that the PLL is a low-pass filter to phase noise at frequencies below  $\omega_N$ . Phase noise at frequencies below  $\omega_N$  passes through the PLL unattenuated. Phase noise at frequencies above  $\omega_N$  is filtered with slope of  $-20$  dB/decade. For small values of  $\zeta$ , the filter cutoff at  $\omega_N$  is sharper with initial slopes as high as  $-40$  dB/decade. However, for these values of  $\zeta$ , the phase noise is amplified at frequencies near  $\omega_N$ . This phase noise amplification or peaking increases, along with the initial cutoff slope, for decreasing values of  $\zeta$ . This phase noise amplification can have adverse affects on the output jitter of the PLL. It is important to notice that because of the zero in the closed loop response, there is a small amount of phase noise amplification at phase noise frequencies of  $\omega_N$  for all values of  $\zeta$ . However, for values of  $\zeta$  less than 0.7, the amplification gain starts to become significant.



**Figure 12.6** PLL closed loop frequency response.

The closed loop transient step response of the PLL for different values of  $\zeta$  and for times normalized to  $1/\omega_N$  is shown in Fig. 12.7. The step response is generated by instantaneously advancing the phase of the reference input by one radian and observing the output for different damping levels in the time domain. For damping factors below one, the system is underdamped as the PLL output overshoots the final phase and rings at the frequency  $\omega_N$ . The amplitude of the overshoot increases and the rate of decay for the ringing decreases as the damping factor is decreased below one. The fastest settling response is generated with a damping factor of one where the system is critically damped. For damping factors greater than one, the system is overdamped as the PLL output initially responds rapidly but then takes a long time to reach the final phase. The rate of the initial response increases and the rate of the final response decreases as the damping factor is increased above one.

### 12.4.2 PLL with Higher-Order Roll-Off

It is very common for an actual PLL implementation to contain an extra capacitor,  $C_2$ , in shunt with the loop filter as shown in Fig. 12.8. This capacitor may have been

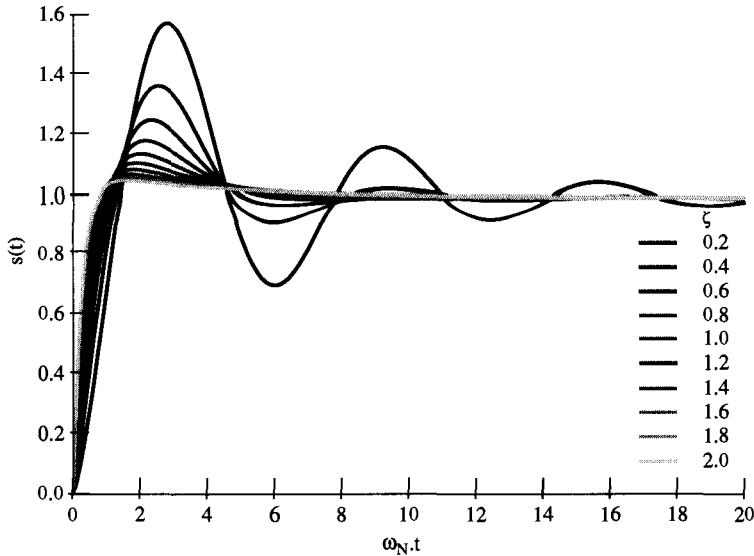


Figure 12.7 PLL closed loop transient step response.

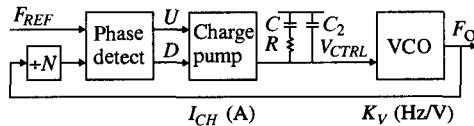


Figure 12.8 Typical PLL block diagram with  $C_2$  (clock distribution omitted).

introduced intentionally for filtering or may result from parasitic capacitances within the resistor or at the input of the VCO.

Because the charge pump and phase detector are activated once every reference frequency cycle, they can cause a periodic disturbance on the control voltage node. This disturbance is usually not an issue for loops with  $N$  equal to one because the disturbance will occur in every VCO cycle. However, the disturbance can cause a constant shift in the duty cycle of the VCO output. When  $N$  is greater than one the disturbance will occur once every  $N$  VCO cycles, which could cause the first one or two of the  $N$  cycles to be different from the others, leading to jitter in the PLL output period. In the frequency domain, this periodic disturbance will cause sidebands on the fundamental peak of the VCO frequency spaced at intervals of the reference frequency.

Capacitor  $C_2$  will help to filter out this reference frequency noise by introducing a pole at  $\omega_C$ . It will decrease the magnitude of the reference frequency sidebands by the ratio of  $\omega_{REF}/\omega_C$ . However, the introduction of  $C_2$  can also cause stability problems for the PLL since it converts the PLL into a third-order system. In addition,  $C_2$  makes the analysis of the PLL much more difficult.

The PLL is now characterized by the four loop parameters  $\omega_N$ ,  $\omega_C$ ,  $\zeta$ , and  $N$ . The damping factor,  $\zeta$ , is changed by  $C_2$  as follows:

$$\zeta = 1/2 \cdot \sqrt{1/N \cdot I_{CH} \cdot K_V \cdot R^2 \cdot C^2 / (C + C_2)}$$

The loop bandwidth,  $\omega_N$ , is changed by  $C_2$  through its dependency on  $\zeta$ . The added pole in the open loop response is at frequency  $\omega_C$  given by

$$\omega_C = (C + C_2)/(R \cdot C \cdot C_2)$$

This pole can reduce the stability of the loop if it is too close to the loop bandwidth frequency. Typically, it should be set at least a factor of ten above the loop bandwidth so as not to compromise the stability loop.

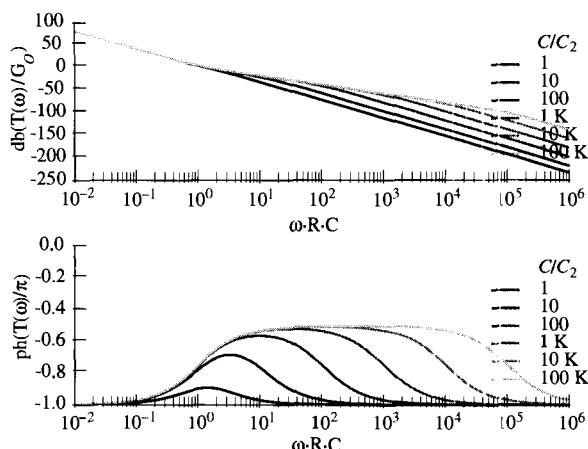
Because the stability of the loop is now established by both  $\zeta$  and  $\omega_C/\omega_N$ , a figure of merit can be defined that represents the stability of the loop as

$$\zeta \cdot \omega_C/\omega_N = (C/C_2 + 1)/2$$

This definition is useful because it actually defines the maximum possible phase margin given an optimal choice for the loop gain magnitude.

Consider the normalized loop gain magnitude and phase plots for the PLL with different ratios of  $C$  to  $C_2$  shown in Fig. 12.9. From these plots it is clear that the added pole at  $\omega_C$  causes the loop gain magnitude slope to increase to  $-40$  dB per decade and the loop gain phase to “increase” to  $-180$  degrees above the frequency of the pole. Between the zero at  $1/(R \cdot C)$  and the pole at  $\omega_C$  there is a region where the loop gain magnitude slope is  $-20$  dB per decade and the loop gain phase approaches  $-90$  degrees. It is in this region where a unity gain crossing would provide the maximum possible phase margin. As the ratio of  $C$  to  $C_2$  increases, this region becomes wider and the maximum phase becomes closer to  $-90$  degrees. Thus the ratio of  $C$  to  $C_2$ , and therefore the figure of merit for stability, defines the maximum possible phase margin.

Based on the frequency response results for the PLL we can make a number of observations about its behavior. First, the continuous time analysis used assumes that the reference frequency is about a decade above all significant frequencies in the response. Second, both the second-order and third-order responses are independent of operating frequency as long as  $K_V$  remains constant. Third, the resistor  $R$  introduces a zero in the open loop response which is needed for stability. Finally, capacitor  $C_2$  can decrease the phase margin if larger than  $C/20$  and can reduce the reference frequency sidebands by  $\omega_{REF}/\omega_C$ .



**Figure 12.9** PLL loop gain magnitude and phase with  $C_2$ .

### 12.4.3 PLL Design Issues

With a good understanding of the PLL frequency response, we can consider issues related to the design of the PLL. The design of the PLL involves first establishing the loop parameters that lead to desirable control dynamics and then establishing device parameters for the circuits that realize those loop parameters.

The loop parameters  $\omega_N$ ,  $\omega_C$ , and  $\zeta$  are often set by the application. The desired value for  $\zeta$  is typically near unity for the fastest overdamped response and about 76 degrees of phase margin, or at least 0.707 for minimal ringing and about 65 degrees of phase margin.  $\omega_N$  must be about one decade below the reference frequency for stability. For frequency synthesis or clock recovery applications, where input jitter filtering is desirable,  $\omega_N$  is typically set relatively low. For input tracking applications, such as clock de-skewing,  $\omega_N$  is typically set as high as possible to minimize jitter accumulation, as discussed in Section 12.6.4. When reference sideband filtering is important,  $\omega_C$  is typically set as low as possible at about a decade above  $\omega_N$  to maximize the amount of filtering.

The values of the loop parameters must somehow be mapped into acceptable values for the device parameters  $R$ ,  $C$ ,  $C_2$ ,  $I_{CH}$ , and  $K_V$ . The values of these parameters are typically constrained by the implementation. The value for capacitor  $C_2$  is determined by all capacitances on the control voltage node if the zero is implemented directly with a resistor. If capacitor  $C$  is implemented on chip, which is desirable to minimize jitter, its size is constrained to less than about 1 nF. The charge pump current  $I_{CH}$  is constrained to be greater than about 10  $\mu$ A depending on the level of charge pump charge injection offsets.

The problem of selecting device parameters is made more difficult by a number of constraining factors. First,  $\omega_N$  and  $\zeta$  both depend on all of the device parameters. Second, the maximum limit for  $C$  and minimum limit for  $I_{CH}$  will impose a minimum limit on  $\omega_N$ , which already has a maximum limit due to  $\omega_{REF}$  and other possible limits due to jitter and reference sideband issues. Third and most important, all worst-case combinations of device parameters due to process, voltage, and temperature variability must lead to acceptable loop dynamics.

Handling the interdependence between the loop parameters and device parameters is simplified by observing some scaling properties that directly result from the equations that relate the loop and device parameters. These rules are shown in Table 12.1. The

**TABLE 12.1** PLL Loop and Device Parameter Scaling Rules

Constant frequency scaling: Given  $x$ , suppose that

$$I_{CH} * x \rightarrow I_{CH}$$

$$C_i * x \rightarrow C_i$$

$$R/x \rightarrow R$$

Then all parameters,  $K_O$ ,  $\omega_i$ , and  $\zeta$  remain constant

Proportional frequency scaling: Given  $x$ , suppose that

$$I_{CH} * x \rightarrow I_{CH}$$

$$I_{CH} * x^2 \rightarrow I_{CH}$$

$$I_{CH} \rightarrow I_{CH}$$

$$C_i/x \rightarrow C_i$$

$$C_i \rightarrow C_i$$

$$C_i/x^2 \rightarrow C_i$$

$$R \rightarrow R$$

$$R/x \rightarrow R$$

$$R * x \rightarrow R$$

Then

$$G_O \rightarrow G_O$$

$$\omega_i * x \rightarrow \omega_i$$

$$\omega_C/\omega_N \rightarrow \omega_C/\omega_N$$

$$\zeta \rightarrow \zeta$$

where  $C_i$  represents all capacitors and  $\omega_i$  represents all frequencies.

constant frequency scaling rules can transform one set of device parameters to another without changing any of the loop parameters. The proportional frequency scaling rules can transform one set of device parameters, with the resistance, capacitances, or charge pump current held constant, to another set with scaled loop frequencies and the same damping factor. These rules make it easy to make adjustments to the possible device parameters with controlled changes to the loop parameters.

With the many constraints on the loop and device parameters established by both the system environment and the circuit implementation, the design of a PLL typically turns into a compromise between conflicting design requirements. It is the job of the designer to properly balance these conflicting requirements and determine the best overall solution.

#### 12.4.4 PLL Design Strategy

There are two general approaches that can be used to determine the device parameters for the PLL design. The first approach is based on an open loop analysis. This approach makes it easier to visualize the stability of the design from a frequency domain perspective. The open loop analysis also easily accommodates more complicated loop filters. The second approach is based on a closed loop analysis. This approach involves the loop parameters  $\omega_N$  and  $\zeta$  which are commonly specified by higher-level system requirements. The complexity of these approaches depends on whether  $C_2$  exists and its level of significance.

If  $C_2$  does not need to be considered, a simplified version of the open loop analysis or second-order analysis can be used. For an open loop analysis without  $C_2$ , we need to consider the open loop response of the PLL in Fig. 12.5. The loop gain normalization constant,  $G_O$ , for the normalized loop gain magnitude plot is directly related to the damping factor  $\zeta$  by

$$G_O = R^2 \cdot C \cdot I_{CH} \cdot K_V / N = 4 \cdot \zeta^2$$

This normalization constant is also the loop gain magnitude at the asymptotic break point for the zero at  $1/(R \cdot C)$ . An increase in the loop gain normalization constant will lead to a higher frequency unity gain crossing, and therefore more phase margin. A plot of phase margin as a function of the damping factor  $\zeta$  is shown in Fig. 12.10. In order to adequately

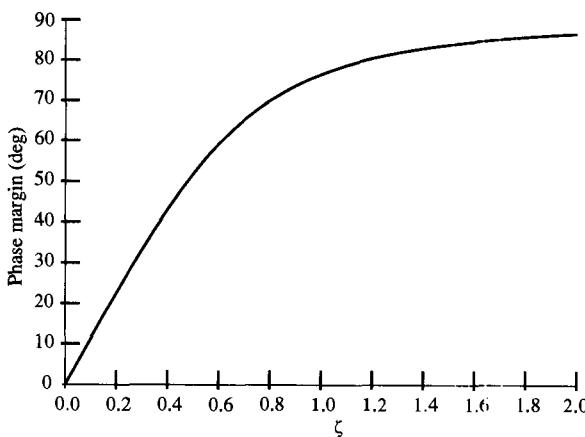


Figure 12.10 PLL phase margin as a function of damping factor.

stabilize the design, the phase margin should be set to 65 degrees or more and the unity gain bandwidth should be set no higher than  $\omega_{REF}/5$ . It is easiest to first adjust the loop gain magnitude level to set the phase margin, then to use the frequency scaling rules to adjust the unity gain bandwidth to the desired frequency. Without  $C_2$ , the second-order analysis simply depends on the loop parameters  $\omega_N$  and  $\zeta$ . To adequately stabilize the design,  $\omega_N$  should be set no higher than  $\omega_{REF}/10$  and  $\zeta$  should be set to 0.707 or greater.

If  $C_2$  exists but is not too large, an extension of the above approaches can be used.  $C$  should be set greater than  $C_2 \cdot 20$  to provide a minimum of 65 degrees of phase margin at the unity gain bandwidth with the maximum phase margin. For any  $C/C_2$  ratio, the maximum phase margin is given by

$$PM_{max} = 2 \cdot \text{atan}(\sqrt{C/C_2 + 1}) - \pi/2$$

With the open loop analysis, as before, the phase margin should be set to at least 65 degrees or at the maximum and the unity gain bandwidth should be set no higher than  $\omega_{REF}/5$ . With the second-order analysis,  $\omega_N$  should be set no higher than  $\omega_{REF}/10$ ,  $\zeta$  should be set to 0.707 or greater, and  $\omega_C$  should be at least a decade above  $\omega_N$ .

If  $C_2$  exists and is large enough to make it difficult to guarantee adequate phase margin, then a third-order analysis must be used. This situation may have been caused by physical constraints on the capacitor sizes, or by attempts to minimize  $\omega_C$  in order to maximize the amount of reference frequency sideband filtering. In this case, it is desirable to determine the optimal values for the other device parameters that maximize the phase margin. The phase margin,  $PM$ , and unity gain bandwidth,  $\omega_O$ , where the phase margin is maximized can be determined from the open loop analysis as

$$PM = 2 \cdot \text{atan}(\sqrt{C/C_2 + 1}) - \pi/2$$

$$\omega_O = \sqrt{\frac{C/C_2 + 1}{R \cdot C}}$$

In order to realize the optimal value for  $\omega_O$ , the loop gain magnitude level must be appropriately set. This can be accomplished by determining  $I_{CH}$  given  $R$  or  $R$  given  $I_{CH}$  using the equations

$$I_{CH} = N/K_V \cdot C_2 / (R \cdot C)^2 \cdot (C/C_2 + 1)^{3/2}$$

$$R = \sqrt{N/(K_V \cdot I_{CH}) \cdot C_2/C^2 \cdot (C/C_2 + 1)^{3/2}}$$

It is important to remember that all worst-case combinations of device parameters due to process, voltage, and temperature variability must be considered since they must lead to acceptable loop dynamics for the PLL to operate correctly under all conditions.

## 12.5 ADVANCED PLL ARCHITECTURES

PLL and DLL architectures each have their own advantages and disadvantages. PLLs are easier to use in systems than DLLs. DLLs typically cannot perform frequency multiplication and have a limited delay range. PLLs, however, are more difficult to design due to conflicting design constraints. It is difficult to assure stability while designing for a high bandwidth.

By using variations on the basic architectures many of these problems can be avoided. DLLs can be designed to perform frequency multiplication by recirculating intermediate edges around the delay line [7]. DLLs can also be designed to have an

unlimited phase shift range by employing a delay line that can produce edges that completely span the clock cycle [4]. In addition, both DLLs and PLLs can be designed to have very wide bandwidths that track the clock frequency by using self-biased techniques [8], as discussed in Section 12.8.

## 12.6 DLL/PLL PERFORMANCE ISSUES

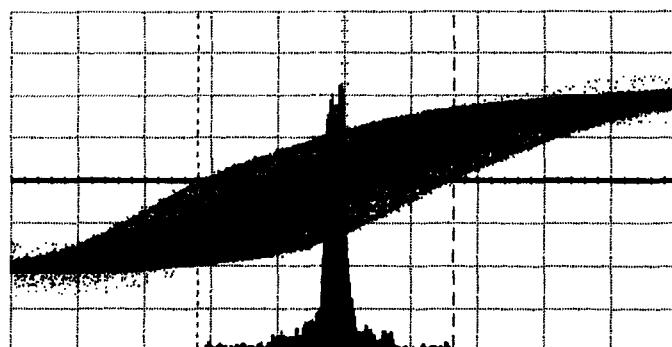
To this point, the chapter presents basic issues concerning the structure and design of DLLs and PLLs. While these issues are important, a good understanding of the performance issues is essential to successfully design a DLL or PLL. There are many performance parameters that can be specified for a DLL or PLL design. They include frequency range, loop bandwidth, loop damping factor (PLL only), input offset, output jitter, both cycle-to-cycle (period) jitter and tracking (input-to-output) jitter, lock time, and power dissipation. However, the biggest performance problems all relate to input offset and output jitter.

Input offset refers to the average offset in the phase of the output clock from its ideal value. It typically results from asymmetries between the circuits for the reference and feedback paths of the phase detector or from charge injection or charge offsets in the charge pump. In contrast, output jitter refers to the time-varying offsets in the phase of the output clock from its ideal value or from some reference signal caused by disturbances from internal and external sources.

### 12.6.1 Output Jitter

Output jitter can create significant problems for an interface by causing set-up and hold time violations which lead to data transmission errors. Consider, for example, the measured jitter histogram in Fig. 12.11. It shows the traces of many PLL output transitions triggered from transitions on the reference input and a histogram with the number of output transitions as a function of their center voltage crossing time. Most of the transition samples occur very close to the reference, while a few outlying transitions occur far to either side of the peak. These outlying transitions must be within the jitter tolerance of the interface. These few edges are typically caused by data-dependent low-frequency noise events with fast rise times.

Output jitter can be measured in a number of ways. It can be measured relative to absolute time, to another signal, or to the output clock itself. The first measurement of



**Figure 12.11** Measured PLL jitter histogram.

jitter is commonly referred to as absolute jitter or long-term jitter. The second is commonly referred to as tracking jitter or input-to-output jitter when the other signal is the reference signal. If the reference signal is perfectly periodic such that it has no jitter, absolute jitter and tracking jitter for the output signal are equivalent. The third is commonly referred to as period jitter or cycle-to-cycle jitter. Cycle-to-cycle jitter can be measured as the time-varying deviations of a single clock period or of several clock periods referred to as cycle-to-Nth-cycle jitter.

Output jitter can also be reported as RMS or peak-to-peak jitter. RMS jitter is interesting only to applications that can tolerate a small number of edges with large time displacements that are well beyond the RMS specification with gracefully degrading results. Such applications can include video and audio signal generation. Peak-to-peak jitter is interesting to applications that cannot tolerate any edges with time displacements beyond some absolute level. The peak-to-peak jitter specification is typically the only useful specification for jitter related to clock generation since most set-up or hold time failures are catastrophic to the operation of a chip.

The relative magnitude for each of these measurements of jitter depends on the type of loop and on how the phase disturbances are correlated in time. For a PLL design, the tracking jitter can be ten or more times larger than the period jitter depending on the noise frequency and the loop bandwidth. For a DLL design, the tracking jitter can be equal to or a factor of two times larger than the period jitter. However, in the particular case when the noise occurs at half the output frequency, the period jitter can be twice the tracking jitter for either the PLL or DLL due to the correlation of output edges times.

### 12.6.2 Causes of Jitter

Tracking jitter for DLLs and PLLs can be caused by both jitter in the reference signal and by noise sources. The noise sources include thermal noise and supply and substrate noise. Thermal noise is generated by the devices within the DLL or PLL and can be significant at low bias currents. Supply and substrate noise is generated by on-chip sources external to the DLL or PLL, including chip output drivers and functional blocks such as adders and multipliers, and by off-chip sources. This noise can be very significant in digital ICs.

The supply and substrate noise generated by the on-chip and off-chip sources is highly data dependent and can have a wide range of frequency components that include low frequencies. Substrate noise tends not to have as large low-frequency components as possible for supply noise since no significant “DC” drops develop between the substrate and the supply voltages. Under worst-case conditions, DLLs and PLLs may experience as much as 500 mV of supply noise and 250 mV of substrate noise with a nominal 2.5 V supply. The actual level of substrate noise depends on the nature of the substrate used by the IC process. To reduce the risk of latch-up, many IC processes use lightly doped epitaxy on the same type heavily doped substrate. These substrates tend to transmit substrate noise across large distances on the chip which make it difficult to eliminate through guard rings and frequent substrate taps.

Supply and substrate noise affect DLLs and PLLs differently. They affect a DLL by causing delay shifts in the delay line output which lead to fixed phase shifts that persist until the noise pulses subside or the DLL can correct the delay error, at a rate limited by its bandwidth (proportional to  $\omega_{REF}/\omega_N$  cycles). They affect a PLL by causing

frequency shifts in the oscillator output which lead to phase shifts that accumulate for many cycles until the noise pulses subside or the PLL can correct the frequency error, at a rate limited by its bandwidth (proportional to  $\omega_{REF}/\omega_N$  cycles). Because the phase error caused by period shifts in PLLs accumulate over many cycles, unlike the delay shifts in DLLs, the tracking jitter for PLLs as a result of supply and substrate noise can be several times larger than the tracking jitter for DLLs. However, due to the added jitter from on-chip clock distribution networks, which typically have poor supply and substrate noise rejection, the observable difference is typically less than a factor of two for well-designed DLLs and PLLs.

### 12.6.3 Supply/Substrate Noise Response

More insight can be gained into the noise response of DLLs and PLLs by considering how much jitter is produced as a function of frequency for supply and substrate noise. For DLLs, the output jitter sensitivity to supply and substrate noise is independent of the loop bandwidth and the reference frequency for the worst case of square wave noise.

For PLLs, several observations can be made about the tracking jitter sensitivity to supply and substrate noise. First, the jitter magnitude decreases inversely proportional to increases in the loop bandwidth for the worst case of square wave noise at frequencies near or below the loop bandwidth. However, the loop bandwidth must be about a decade below the reference frequency, which imposes a lower limit on the jitter magnitude. Second, the jitter magnitude decreases inversely proportional to the reference frequency for a fixed Hz/V frequency sensitivity, since the phase disturbance measured in radians is constant, but the reference period decreases inversely proportional to the reference frequency. Third, the jitter magnitude is independent of reference frequency for fixed %/V frequency sensitivity, since the phase disturbance measured in radians changes inversely proportional to the reference period. Finally, the jitter magnitude increases directly proportional to the square root of  $N$ , the feedback divider value, with a constant oscillator frequency and if the loop is overdamped, since the loop bandwidth is inversely proportional to the square root of  $N$ .

### 12.6.4 Observations on Jitter

The optimal loop bandwidth depends on the application for the PLL. For frequency synthesis or clock recovery applications, where the goal is to filter out jitter on the input signal, the loop bandwidth should be as low as possible. For this application, the phase relationship between the output of the PLL and other clock domains is typically not an issue. As a result, the only jitter of significance is period jitter and possibly jitter spanning a few clock periods. This form of jitter does not increase with reductions in the loop bandwidth. However, if the phase relationship between the PLL output and other clock domains is important or if the jitter of the PLL output over a large number of cycles is significant, then the loop bandwidth should be maximized. Maximizing the loop bandwidth will minimize this form of jitter since it decreases proportional to increases in loop bandwidth.

Because of the hostile noise environments of digital chips, the peak value of the measured tracking jitter from DLLs and PLLs will likely be caused by square wave supply and substrate noise. For PLLs, this noise is particularly significant when the noise frequencies are at or below the loop bandwidth. If a PLL is underdamped, noise frequencies near the loop bandwidth can be even more significant. In addition, a PLL

can amplify input jitter at frequencies near the loop bandwidth, especially if it is under-damped. However, as previously discussed, jitter in a PLL or DLL can also be caused by a dead-band region in phase detector and charge pump characteristics.

In order to minimize jitter it is necessary to minimize supply and substrate noise sensitivity of the VCDL or VCO. The supply and substrate noise sensitivity can be separated into both static and dynamic components. The static components relate to the sensitivity to the DC value of the supply or substrate voltage. The static noise sensitivity can predict the noise response for all but the high-frequency components of the supply and substrate noise. The dynamic components relate to the extra sensitivity to a sudden change in the supply or substrate voltage that the static components do not predict. The effect of the dynamic components increases with increasing noise edge rate. For PLLs, the dynamic noise sensitivity typically has a much smaller overall impact on the supply and substrate noise response than the static noise sensitivity. However, for DLLs, the dynamic noise sensitivity can be more significant than static noise sensitivity. Only static supply and substrate noise sensitivity are considered in this chapter.

### 12.6.5 Minimizing Supply Noise Sensitivity

All VCDL and VCO circuits will have some inherent sensitivity to supply noise. In general, supply noise sensitivity can be minimized by isolating the delay elements used within the VCDL or VCO from one of the supply terminals. This goal can be accomplished by using a buffered version of the control voltage as one of the supply terminals. However, this technique can require too much supply voltage headroom. The preferred and most common approach is to use the control voltage to generate a supply independent bias current so that current sources with this bias current can be used to isolate the delay elements from the opposite supply.

Supply voltage sensitivity is directly proportional to current source output conductance. Simple current sources provide a delay sensitivity per fraction of the total supply voltage change ( $(dt/t)/(dV_{DD}/V_{DD})$ ), of about 10%, such that if the supply voltage changed by 10% the delay would change by 1%. This level of delay sensitivity is too large for good jitter performance in PLLs. Cascode current sources provide an equivalent delay sensitivity of about 1%, such that if the supply voltage changed by 10% the delay would change by 0.1%, which is at the level needed for good jitter performance. However, cascode current sources can require too much supply voltage headroom. Another technique that can also offer an equivalent delay sensitivity of about 1% is replica current source biasing [9]. In this approach, the bias voltage for simple current sources is actively adjusted by an amplifier in a feedback configuration to keep some property of the delay element, such as voltage swing, constant and possibly equal to the control voltage.

Once adequate measures are taken to minimize the current source output conductance, other supply voltage dependencies may begin to dominate the overall supply voltage sensitivity of the delay elements. These effects include the dependencies of threshold voltage and diffusion capacitance for switching devices on the source or drain voltages, which can be modulated by the supply voltage. With any supply terminal isolation technique, all internal switching nodes will have voltages that track the supply terminal opposite to the one isolated. Thus, these effects can be manifested by devices with bulk terminals connected to the isolated supply terminal. These effects are always a problem for substrate devices with an isolated substrate-tap voltage supply

terminal, such as for NMOS devices in an N-well process with an isolated negative supply terminal. Isolating the well-tap voltage supply terminal avoids this problem since the bulk terminals of the well devices can be connected to their source terminals, such as with PMOS devices in an N-well process with an isolated positive supply terminal. However, such an approach leads to more significant substrate noise problems. The only real solution is to minimize their occurrence and to minimize their switching diffusion capacitance. Typically these effects will establish a minimum delay sensitivity per fraction of the total supply voltage change of about 1%.

### 12.6.6 Supply Noise Filters

Another technique to minimize supply noise is to employ supply filters. Supply filters can be passive, or active, or a combination of the two. Passive supply filters are basically low-pass filters. Off-chip passive filters work very well in filtering out most off-chip noise but do little to filter out on-chip noise. Unfortunately, on-chip filters can have difficulty in filtering out low-frequency on-chip noise. Off-chip capacitors can easily be made large enough to filter out low-frequency noise, but on-chip capacitors are much more limited in size. In order for the filter to be effective in reducing jitter for both DLLs and PLLs, the filter cut-off frequency must be below the loop bandwidth.

Active supply filters employ amplifiers in a feedback configuration to buffer a desired reference supply voltage and act as high-pass filters. The reference supply voltage is typically established by a band gap or control voltage reference. The resultant supply isolation will decrease with increasing supply filter bandwidth due to basic amplifier feedback trade-offs. In order for the active filter to be effective, the bandwidth must exceed the inverse VCDL delay of a DLL or the loop bandwidth of a PLL. The DLL bandwidth limit originates because the VCDL delay will begin to be less affected by a noise event if it subsides before a signal transition propagates through the complete VCDL. The PLL bandwidth limit exists because, as higher-frequency noise is filtered out above the loop bandwidth, the VCO will integrate the resultant change in frequency for fewer cycles. While the PLL bandwidth limit is achievable in a supply filter with some level of isolation, the DLL bandwidth limit is not. Thus, while active supply filters can help PLLs, they are typically ineffective for DLLs. However, the combination of passive and active filters can be an effective supply noise-filtering solution for both PLLs and DLLs by avoiding the PLL and DLL bandwidth constraints. When the low-pass filter cutoff frequency is below the high-pass filter cutoff frequency, filtering can be achieved at both low and high frequencies so that tracking bandwidths and inverse VCDL delays are not an issue.

Other common isolation approaches include using separate supply pins for a DLL or PLL. This approach should be used whenever possible. However, the isolated supplies will still experience noise from coupling to other supplies through off-chip paths and coupling to the substrate through well contacts and diffusion capacitance, requiring that supply noise issues be addressed. Also, having separate supply pins at the well tap potential can lead to increased substrate noise depending on the overall conductivity of the substrate.

### 12.6.7 Minimizing Substrate Noise Sensitivity

Substrate noise sensitivity like supply noise sensitivity can create jitter problems for a PLL or DLL. Substrate noise can couple into the delay elements by modulating device threshold voltages. Substrate noise can be minimized by only using well-type devices for

fixed-biased current sources, only using well-type devices for the loop filter capacitor, only connecting the control voltage to well-type devices, and only using the well-tap voltage as the control voltage reference. These constraints will insure that substrate noise does not modulate fixed-bias current source outputs or the conductance of devices connected to the control voltage both through threshold modulation. In addition, they will prevent supply noise from directly summing into the control voltage through a control voltage reference different from the loop filter capacitor reference. Even with these constraints, substrate noise can couple into switching devices, as with supply noise, through the threshold voltage and diffusion capacitance dependencies on the substrate potential.

Substrate noise can be converted to supply noise by connecting the substrate potential supply terminals of the delay elements only to the substrate [10]. This technique insures that the substrate and the substrate potential supply terminals are at the same potential. However, it only works with low operating currents since otherwise voltage drops will be generated in the substrate and excessive minority carriers will be dumped into the substrate.

## 12.7 DLL/PLL CIRCUITS

Prior sections discussed design issues related to DLL and PLL loop architectures and low-output jitter. With these issues in mind, this section discusses the circuit-level implementation issues of the loop components. These components include the VCDL and VCO, phase detector, charge pump, and loop filter.

### 12.7.1 VCDLs and VCOs

The VCDL and VCO are the most critical parts of DLL and PLL designs for achieving low-output jitter and good overall performance. There are two general types of VCDLs with analog control. First, a VDCL can interpolate between two delays through an analog-weighted sum circuit. This approach only leads to linear control over delay if the two interpolated delays are relatively close, which restricts the overall range of the VCDL. Second, a VCDL can be based on an analog delay line composed of identical cascaded delay elements, each with a delay that is controlled by an analog signal. This approach usually leads to a wide delay range with nonlinear delay control. A wide delay range is often desired in order to handle a range of operating frequencies and process and environmental variability. However, nonlinear delay control can restrict the usable delay range due to undesirable loop dynamics.

There are several types of VCOs. First, a VCO can be based on an LC tank circuit. This type of oscillator has very high supply noise rejection and low phase noise output characteristics. However, it usually also has a restricted tuning range which makes it impractical for digital ICs. Second, a VCO can be based on a relaxation oscillator. The frequency in this circuit is typically established by the rate a capacitor can be charged and discharged over some established voltage range with an adjustable current. This approach typically requires too much supply headroom to achieve good supply noise rejection and can be extra sensitive to sudden changes in the supply voltage. Third and most popular for digital ICs, a VCO can be based on a phase shift oscillator, also known as a ring oscillator. A ring oscillator is a ring of identical cascaded delay elements with inverting feedback between the two elements that close the ring. A ring oscillator can typically generate frequencies over a wide range with linear control over frequency.

The delay elements, also known as buffer stages, used in a delay line or ring oscillator can be *single-ended*, that is, they have only one input and one output and invert

the signal, or *differential*, such that they have two complementary inputs and outputs. Single-ended delay elements typically lead to reduced area and power, but provide no complementary outputs. Complementary outputs provide twice as many output signals with phases that span the output period compared to single-ended outputs, and allow a 50% duty cycle signal to be cleanly generated without dividing the output frequency by two. Differential delay elements typically have reduced dynamic noise coupling to their outputs and provide complementary outputs.

A number of factors must be considered in the design of the delay elements. The delay of the delay elements should have a linear dependence on control voltage when used in a VCDL and an inverse linear dependence on control voltage when used in a VCO. These control relationships will make the VCDL and VCO control gains constant and independent of the operating frequency which will lead to operating frequency independent loop dynamics. The static supply and substrate noise sensitivity should be as small as possible, ideally less than 1% delay sensitivity per fraction of the total supply voltage change. As previously discussed, this reduced level of supply sensitivity can be established with current source isolation.

Figure 12.12 shows a single-ended delay element circuit for an N-well CMOS process. This circuit contains a PMOS common-source device with a PMOS diode clamp and a simple NMOS current source. The diode clamp restricts the buffer output swing in order to keep the NMOS current source device in saturation. In order to achieve high static supply and substrate noise rejection, the bias voltage for the simple NMOS current source is dynamically adjusted with changes in the supply or substrate voltage to compensate for its finite output impedance.

Figure 12.13 shows a differential delay element circuit for an N-well CMOS process [9]. This circuit contains a source coupled pair with resistive load elements called *symmetric loads*. Symmetric loads consist of a diode-connected PMOS device in shunt with an equally sized biased PMOS device. The PMOS bias voltage  $V_{BP}$  is nominally equal to  $V_{CTRL}$ , the control input to the bias generator.  $V_{BP}$  defines the lower voltage swing limit of the buffer outputs. The buffer delay changes with  $V_{BP}$  because the effective resistance of the load elements also changes with  $V_{BP}$ . It has been shown that these load elements lead to good control over delay and high dynamic supply noise rejection. The simple NMOS current source is dynamically biased with  $V_{BN}$  to compensate for drain and substrate voltage variations, achieving the effective static supply noise rejection performance of a cascode current source without the extra supply

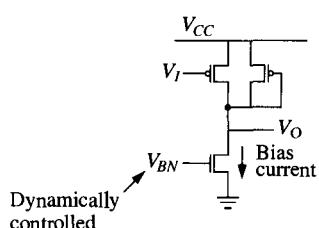


Figure 12.12 Single-ended delay element for an N-well CMOS process.

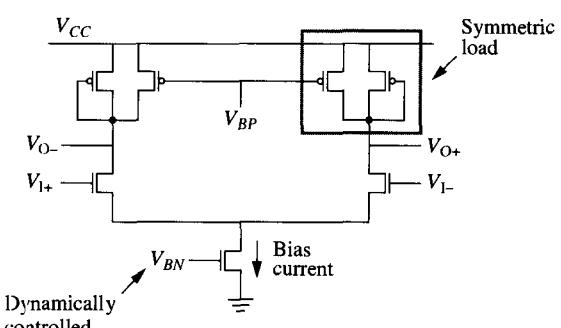


Figure 12.13 Differential delay element with symmetric loads for an N-well CMOS process.

voltage required by cascode current sources. Buffer stages with low supply and substrate noise sensitivity are essential for low-jitter DLL and PLL operation.

### 12.7.2 Phase Detectors

The phase detector detects the phase difference between the reference input and the feedback signal of a DLL or PLL. There are several types of phase detectors that can be used, each of which will allow the loop to achieve a different phase relationship once in lock. An XOR or mixer can be used as a phase detector to achieve a quadrature lock on input signals with a 50% duty cycle. The UP and DN outputs are complementary, and, once in lock, each will generate a 50% duty cycle signal at twice the reference frequency. The 50% duty cycle will cause the UP and DN currents to cancel out leaving the control voltage unchanged. An edge-triggered SR latch can be used as the phase detector for an inverted lock. The UP and DN outputs are also complementary, and, once in lock, each will generate a 50% duty cycle signal at the reference frequency. If differential inputs are available, an inverted lock can be easily interchanged with an in-phase lock. A sampling flip-flop can be used to sample the reference clock as the phase detector in a digital feedback loop, where the flip-flop is used to match the input delay for digital inputs also sampled by identical flip-flops. The output state of the flip-flop will indicate if the feedback clock is early or late. Finally, a phase-frequency detector (PFD) can be used as a phase detector to achieve an in-phase lock. PFDs are commonly based on two SR latches or two D flip-flops. They have the property that only UP pulses are generated if the frequency is too low; only DN pulses are generated if the frequency is too high; and to first order, no UP or DN pulses are generated once in lock. Because of this property, PLLs using PFDs will slew their control voltage with, on average, half of the charge pump current until the correct frequency is reached, and will never falsely lock at some harmonic of the reference frequency. PFDs are the most common phase detectors used in DLLs and PLLs.

Phase detectors can have several potential problems. The phase detector can have an input offset caused by different edge rates between the reference and feedback signals or caused by asymmetric circuits or device layouts between the reference and feedback signal paths. In addition, the phase detector can exhibit nonlinearity near the locking point. This nonlinearity can include a dead-band, caused by an input delay difference where the phase detector output remains zero or unchanged, or a high gain region, caused by an accelerated sensitivity to transitions on both the reference and feedback inputs. In order to properly diagnose potential phase detector problems, the phase detector must be simulated or tested in combination with the charge pump.

### 12.7.3 Charge Pumps

Charge pumps can be structured in a number of ways. The key issues for the structure are input offset and linearity. An input offset can be caused by a mismatch in charge-up or charge-down currents or by charge injection. The nonlinearity near the lock point can be caused by edge-rate dependencies and current source switching.

A push-pull charge pump is shown in Fig. 12.14. This charge pump tends to have low output ripple because small but equal UP and DN pulses, produced by a PFD once in lock, generate equal current pulses at exactly the same time that cancel out with an insignificant disturbance to the control voltage. The switches for this charge pump are best placed away from the output toward the supply rails in order to minimize charge injection from the supply rails to the control voltage. The opposite configuration can

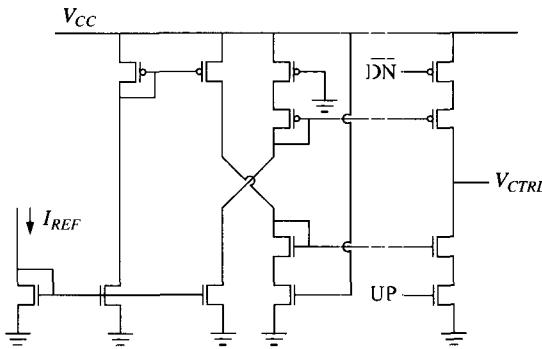


Figure 12.14 Push-pull charge pump.

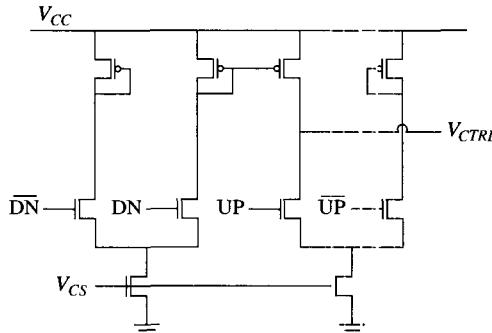


Figure 12.15 Current mirror charge pump.

inject charge from the supply rails through the capacitance at the shared node between the series devices.

A current mirror charge pump is shown in Fig. 12.15. This charge pump tends to have the lowest input offset due to balanced charge injection. In the limit that a current mirror has infinite output impedance, it will mirror exact charge quantities. However, because the DN current pulse is mirrored to the output, it will occur later and have a longer tail than the UP current pulse which is switched directly to the output. This difference in current pulse shape will lead to some disturbance to the control voltage.

Another combined approach for the charge pump and loop filter involves using an amplifier-based voltage integrator. This approach is difficult to implement in most IC processes as it requires floating capacitors. Any of the above approaches can be modified to work in a “bang-bang” mode, where the output charge magnitude is fixed independent of the phase error. This mode of operation is sometimes used with digital feedback loops when it is necessary to cancel the aperture offset of a high-speed interface receiver [9]. However, it makes the loop control very nonlinear and commonly produces dither jitter, where the output phase, once in lock, alternates between positive and negative errors.

#### 12.7.4 Loop Filters

The loop filter directly connects to the charge pump to integrate and filter the detected phase error. The most important detail for the loop filter is which supply terminal is to be used as the control voltage reference. As discussed in Section 12.6.7, substrate noise can couple into delay elements through threshold modulation of the active devices. The substrate noise sensitivity can be minimized by using well-type devices for the loop filter capacitor and for fixed-biased devices. Also, care must be

taken to insure that the voltage reference used by the circuitry that receives the control voltage is the same as the supply terminal to which the loop filter capacitor connects. Otherwise, any supply noise will be directly summed with the control voltage.

Some designs employ level shifting between the loop filter voltage and the control voltage input to the VCDL or VCO. Such level shifting is often the cause of added supply noise sensitivity and should be avoided whenever possible. Also, some designs employ differential loop filters. A differential loop filter is useful only if the control input to the VCDL or VCO is differential, as is often the case with a delay interpolating VCDL. If the VCDL or VCO has a single-ended control input, a differential loop filter adds no value because its output must be converted back to a single-ended signal. Also, the differential loop filter needs some type of common-mode biasing to establish the common-mode voltage. The common-mode bias circuit will add some differential mode resistance that will cause the loop filter to leak charge and will lead to an input offset for the DLL or PLL.

For PLLs, the loop filter must implement a zero in order to provide phase margin for stability. The zero can be implemented directly with a resistor in series with the loop filter capacitor. In this case, the charge pump current is converted to a voltage through the resistor which is added to the voltage across the loop filter capacitor to form the control voltage. Alternatively, this zero can be formed by summing an extra copy of the charge pump current directly with a bias current used to control the VCO, possibly inside a bias generator for the VCO. This latter approach avoids using an actual resistor and lends itself to self-biased schemes [6].

### 12.7.5 Circuit Summary

In general, all DLL and PLL circuits must be designed from the outset with supply and substrate noise rejection in mind. Obtaining low noise sensitivity requires careful orchestration among all circuits and cannot be added as an after thought. Supply noise rejection requires isolation from one supply terminal, typically with current source isolation. Substrate noise rejection requires all fixed-biased devices to be well-type devices to minimize threshold modulation. However, the best circuits to use depend on both the loop architecture and the IC technology.

## 12.8 SELF-BIASED TECHNIQUES

Achieving low tracking jitter and a wide operating frequency range in PLL and DLL designs can be difficult due to a number of design trade-offs. To minimize the amount of tracking jitter produced by a PLL, the loop bandwidth should be set as high as possible. However, the loop bandwidth must be set at least a decade below the lowest desired operating frequency for stability with enough margin to account for bandwidth changes due to the worst-case process and environmental conditions. Achieving a wide operating frequency range in a DLL requires that the VCDL work over a wide range of delays. However, as the delay range is increased, the control becomes increasingly nonlinear, which can undermine the stability of the loop and lead to increased jitter. These different trade-offs can cause both PLLs and DLLs to have narrow operating frequency ranges and poor jitter performance.

Self-biasing techniques can be applied to both PLLs and DLLs as a solution to these design trade-off problems [6]. Self-biasing can remove virtually all of the process technology and environmental variability that affect PLL and DLL designs, and provide a loop bandwidth that tracks the operating frequency. This tracking bandwidth sets

no limit on the operating frequency range and makes wide operating frequency ranges spanning several decades possible. This tracking bandwidth also allows the bandwidth to be set aggressively close to the operating frequency to minimize tracking jitter. Other benefits of self-biasing include a fixed damping factor for PLLs and input phase offset cancellation. Both the damping factor and the bandwidth to operating frequency ratio are determined completely by a ratio of capacitances giving effective process technology independence. In general, self-biasing can produce very robust designs.

The key idea behind self-biasing is that it allows circuits to choose the operating bias levels in which they function best. By referencing all bias voltages and currents to other generated bias voltages and currents, the operating bias levels are essentially established by the operating frequency. The need for external biasing, which can require special band-gap bias circuits, is completely avoided. Self-biasing typically involves using the bias currents in the VCO or VCDL as the charge pump current. Special accommodations are also added for the feed-forward resistor needed in a PLL design.

## 12.9 CONCLUSION

In conclusion, DLLs and PLLs can be used to relax system timing constraints. The best loop architecture strongly depends on the system application and the system environment. DLLs experience less jitter than PLLs due to their inherently reduced noise sensitivity. PLLs provide more flexibility by supporting frequency multiplication and an unlimited phase range. Independent of the chosen loop architecture, supply and substrate noise will likely be the most significant cause of output jitter. As such, all circuits must be designed from the outset with supply and substrate noise rejection in mind.

## REFERENCES

- [1] M. Johnson and E. Hudson, "A Variable Delay Line PLL for CPU-Coprocessor Synchronization," *IEEE J. Solid-State Circuits*, vol. SC-23, no. 5, pp. 1218–1223, Oct. 1988.
- [2] I. Young, et al., "A PLL Clock Generator with 5 to 110 MHz of Lock Range for Microprocessors," *IEEE J. Solid-State Circuits*, vol. 27, no. 11, pp. 1599–1607, Nov. 1992.
- [3] F. Gardner, "Charge-Pump Phase-Lock Loops," *IEEE Trans. Communications*, vol. COM-28, no. 11, pp. 1849–1858, Nov. 1980.
- [4] S. Sidiropoulos and M. Horowitz, "A Semidigital Dual Delay-locked Loop," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1683–1692, Nov. 1997.
- [5] T. Lee, et al., "A 2.5 V CMOS Delay-Locked Loop for an 18Mbit, 500Megabyte/s DRAM," *IEEE J. Solid-State Circuits*, vol. 29, no. 12, pp. 1491–1496, Dec. 1994.
- [6] D. Chengson, et al., "A Dynamically Tracking Clock Distribution Chip With Skew Control," *CICC 1990 Dig. Tech. Papers*, pp. 13–16, May 1990.
- [7] A. Waizman, "A Delay Line Loop for Frequency Synthesis of De-Skewed Clock," *ISSCC 1994 Dig. Tech. Papers*, pp. 298–299, Feb. 1994.
- [8] J. Maneatis, "Low-Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1723–1732, Nov. 1996.
- [9] J. Maneatis and M. Horowitz, "Precise Delay Generation Using Coupled Oscillators," *IEEE J. Solid-State Circuits*, vol. 28, no. 12, pp. 1273–1282, Dec. 1993.
- [10] V. von Kaenel, et al., "A 600 MHz CMOS PLL Microprocessor Clock Generator with a 1.2GHz VCO," *ISSCC 1998 Dig. Tech. Papers*, pp. 396–397, Feb. 1998.
- [11] M. Horowitz, et al., "PLL Design for a 500MB/s Interface," *ISSCC 1993 Dig. Tech. Papers*, pp. 160–161, Feb. 1993.

Daniel W. Bailey  
*Compaq Computer Corporation*

### **13.1 INTRODUCTION**

Microprocessor clock design is a demanding challenge for several reasons. First and foremost, any clock performance shortcoming systemically affects the microprocessor performance. That is, if the floating-point multiplier takes too many cycles, only certain benchmarks, such as SPECfp95, will suffer. If a clock does not meet performance specifications, then *all* CPU benchmarks will suffer. Second, microprocessor clock design stretches the data-handling capacity and throughput of computer aided design (CAD) tools. This is because, in aggregate, the clock usually has the largest fan-out of any node on the die, and consumes more power than any other node [1]. Third, unlike most CMOS digital circuits that migrate well into new processes, clock designs get harder with each generation. Data path design of an integer-execution unit, for example, can usually be based on the schematics from a previous generation. The clock on a new microprocessor, on the other hand, might have to drive more than twice the area in design units with proportionately higher loads than the previous generation [2],[3], and do so with an equivalent or higher performance specification. Simply scaling the clock driver sizes and expanding the clock distribution network does not work in practice, first because floorplans change, and second because techniques that improve performance, decrease power, and reduce sensitivity to variations tend to conflict with one another. The bottom line is that innovative, CAD-tool friendly, and carefully verified approaches must be used with each new microprocessor clock design.

To successfully meet the challenges and objectives of a new clock design, implementation fundamentals and trade-offs must be clearly understood. This is true even when relying heavily on sophisticated CAD tools. This chapter focuses on tasks that are essential to microprocessor clock design and are needed to complement CAD-tool analysis. These basic implementation tasks can be grouped into the following categories:

1. defining objectives,
2. choosing a clock distribution network implementation,
3. specifying a clock driver layout, and
4. accounting for sources of variation.

Taken in tandem with a conservative simulation and verification strategy, the above tasks form the foundation for a robust, successful clock design.

### 13.1.1 Definitions

First of all, *clock* is a signal used to synchronize storage elements in a digital state machine [4], where the state machine referenced in this chapter is generally a microprocessor. *Clock* can also refer to the circuitry that generates the clock signal. Rising and falling edges of the clock must be carefully controlled in the face of process and environmental variations, and not be subject to signal integrity problems, otherwise the electrical and timing integrity of the state elements are jeopardized. If the electrical or timing integrity is compromised, the processor performance and functionality are threatened.

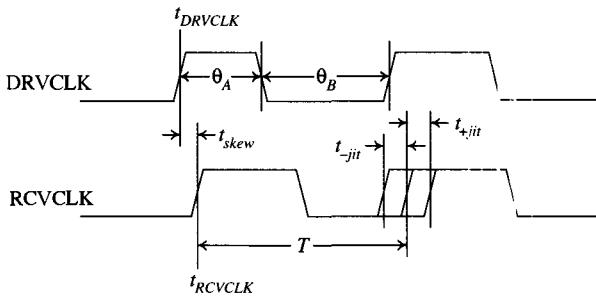


Figure 13.1 Definitions of skew, jitter, and duty cycle.

In practice, clock nonideality is manifested as variation of the delay, or as variation of rise and fall times. Figure 13.1 shows how delay variations can be characterized. *Skew* can be defined as the absolute delay between two clock edges measured at  $V_{DD}/2$  [5],[6]. This definition of skew  $t_{skew}$  in equation form is given by

$$t_{skew} = |t_{DRVCLK} - t_{RCVCLK}| \quad (13.1)$$

where  $t_{DRVCLK}$  and  $t_{RCVCLK}$  are the delays to the initial and final clocking points, respectively. Skew as defined in Eq. (13.1) can exist for a single node at two different locations, or for two different nodes. It is caused by an imbalance between clock drivers, interconnect delays, or loads for the two clock paths. As an alternative definition, skew can also be defined for both positive and negative values by assigning initial and final clock paths. In equation form, this second definition of skew is [4]

$$t_{skew} = t_{DRVCLK} - t_{RCVCLK} \quad (13.2)$$

Negative skew is shown in the example in Fig. 13.1. The definition in Eq. (13.2) is useful for discussing the benefits and disadvantages of skew as applied to particular timing paths. The counterpoint to adopting this definition exclusively is that it is path dependent, not location dependent. If there is a feedback path in a circuit, then initial and final nodes depend on which path is being considered. Thus, both definitions are useful.

*Long-term jitter* can be defined as the accumulated temporal variation of clock periods about the nominal period. If there is deterministic jitter, then the period distribution may exhibit multimodal characteristics, and thus may be asymmetric about the mean. *Cycle-to-cycle jitter* can be defined as the variation of consecutive clock periods. Jitter is strictly a temporal variation and is measured at a specific location for a single node, although it may apply globally. It can be attributed to data-dependent loads, coupling events, or variations of supply voltage or temperature. Figure 13.1 illustrates clock jitter as an uncertainty in the delay of a clock edge comprised of

compressing and expanding terms,  $t_{-jit}$  and  $t_{+jit}$ , respectively. Long-term jitter can negatively affect synchronization of external communications. Cycle-to-cycle jitter results in phase compression, which degrades microprocessor performance by forcing a lower operating frequency [7],[8].

*Duty cycle* is the percentage ratio of the phase to the period. In Fig. 13.1, the duty cycle of DRVCLK can be expressed as either  $100\% \times \theta_A/T$ , as  $100\% \times \theta_B/T$ , or as the combination of percentages. A poor or fluctuating duty cycle can originate in the clock synthesis, or be caused by asymmetry in the two paths through a clock driver chain. Similar to nonzero skew and jitter, a non-50% duty cycle is a form of phase compression.

Clock performance is not important in and of itself, but in how it affects circuit performance. Figure 13.2 shows an example of a *triplet*: a driving flip-flop clocked by DRVCLK, intervening circuits and interconnect, and a receiving flip-flop clocked by RCVCLK. A triplet is the basic circuit that provides context for evaluating clock performance [4], although it can exist in different forms and be complicated by intermediate latches. Figure 13.3(a) illustrates an example of appropriate timing for this circuit assuming both flip-flops are positive edge triggered, that is, triggered on the rising edge of clock. Input to the first flip-flop, DRVIN, is ready in advance of the DRVCLK edge. The signal proceeds forward at the DRVCLK edge with delay through the flip-flop given by  $t_{CLK-to-Q}$ , and delay through the circuits and interconnect denoted as  $t_{cir\&int}$ . Minimum and maximum delay of the signal path is shown in Fig. 13.3(a) as two possible edges for RCVIN. RCVIN must be set up and constant for the receiving flip-flop in advance of RCVCLK for correct functionality. If so, the signal will successfully be latched as shown on the output of the receiving flip-flop, OUT, after RCVCLK fires.

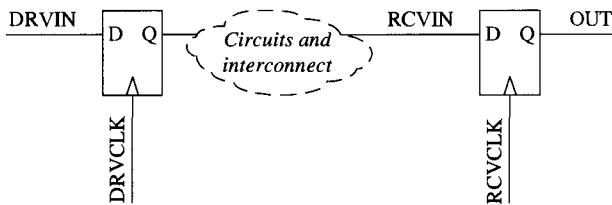
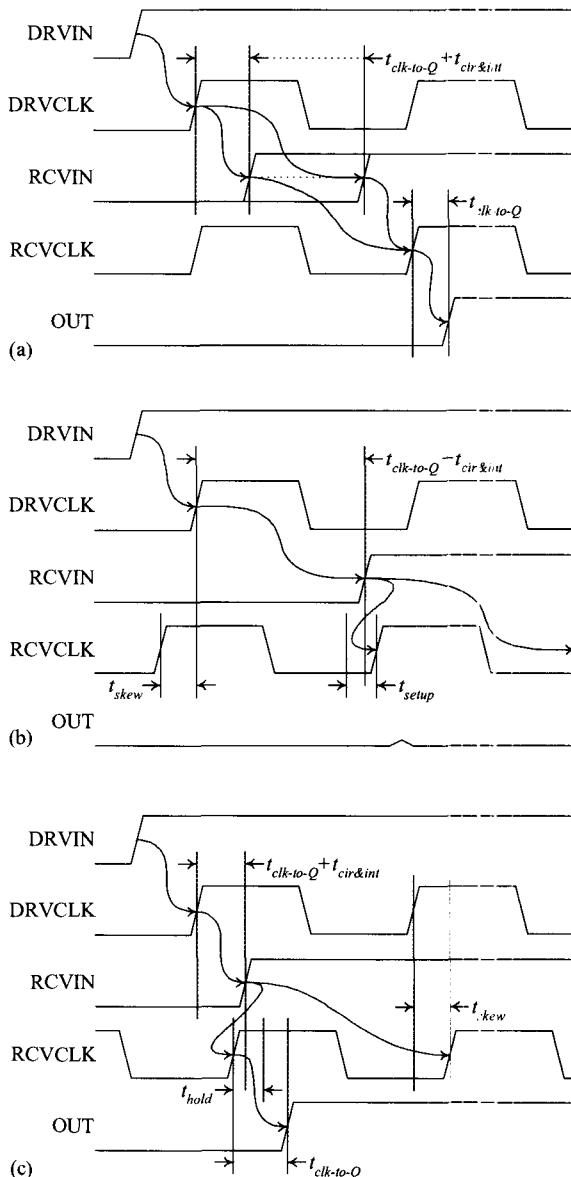


Figure 13.2 Circuit diagram of a triplet comprised of a driving flip-flop, intervening circuits and interconnect, and a receiving flip-flop.

Variation of the clock edge can degrade performance by decreasing the time allowed for the data path circuitry. A *set-up-time failure* is shown in Fig. 13.3(b). In this example, positive skew or jitter between RCVCLK and DRVCLK causes RCVIN to change during the set-up time of the receiving flip-flop. OUT does not change until the following RCVCLK rising edge, thus resulting in failure. This example shows that if clock variation is present either in the form of skew or jitter, then the clock period for this path must be extended. This can be done either by lowering the clock frequency, redesigning the clock distribution network by making RCVCLK fire later or DRVCLK earlier, using a driving flip-flop with lower latency, using a receiving flip-flop with lower set-up time, or by redesigning the data path to decrease  $t_{cir\&int}$ .

Variation of the clock edge can also cause a functional failure that is unresponsive to frequency changes. In this case, called a *race-through* or *hold-time failure*, data can proceed too quickly through the driving flip-flop, intervening circuits and interconnect



**Figure 13.3** Timing diagrams for the triplet in Fig. 13.2 showing (a) proper timing, (b) set-up failure because of (positive) skew between DRVCLK and RCVCLK, and (c) hold failure because of (negative) skew between DRVCLK and RCVCLK.

such that it gets latched by the receiving flip-flop on the same clock edge. Figure 13.3(c) shows a timing diagram of how negative clock skew can cause race-through failure. In this example, skew that delays the rising edge of RCVCLK compared to DRVCLK causes RCVIN to change during the hold time of the receiving flip-flop. Since this occurs on a single clock edge, extending the clock period for this path is ineffective. Changing the operating temperature or power supply voltage might allow the path to work successfully, but these changes are impractical for a commercialized product. Fabricating a new die in a different process corner may also work, but it is costly and impractical. The best solutions are redesigning the clock distribution by making RCVCLK fire earlier or

DRVCLK later, using a driving flip-flop with longer latency, a receiving flip-flop with lower hold time, or by redesigning the data path to increase  $t_{cir\&int}$ .

## 13.2 OBJECTIVES

### 13.2.1 Ideal Clock Objectives

An ideal microprocessor clock does not exist, but defining one is a useful exercise. It helps establish the implementation compromises that have to be made in designing a real clock, and clarifies which objectives are most important. First, an ideal clock has zero skew, zero jitter, and a 50% duty cycle. It follows that an actual high-performance microprocessor clock should have low skew, low jitter, and a duty cycle near 50%. A clock distribution network without these characteristics can jeopardize the commercial viability of a microprocessor because of poor performance or low yields.

Another primary characteristic of an ideal clock is that it uses no power  $P$  beyond that needed by the total clock gate load, which for an ungated clock is given by [5],[6]

$$P = C_L V_{DD}^2 f \quad (13.3)$$

where  $C_L$  is the total effective gate capacitance of the clock load, and  $f$  is the clock frequency. Note that Eq. (13.3) does not include an “activity factor” as typically seen in equations describing CMOS dynamic power. For an unconditional clock, the activity factor is unity because it switches every cycle. Equation (13.3) is incomplete, however, even for the concept of an ideal clock.

The total power dissipated by a clock also includes power consumed by the clock distribution network. Assuming a constant fan-out  $\lambda$  for each stage in the distribution network, the total clock power  $P_{tot}$  is given by

$$P_{tot} = C_L V_{DD}^2 f + C_L V_{DD}^2 f / \lambda + C_L V_{DD}^2 f / \lambda^2 + \dots = C_L V_{DD}^2 f \sum_{n=0}^{stages-1} \frac{1}{\lambda^n} \quad (13.4)$$

Equation (13.4) shows that most of the power is dissipated in the final stages. For a 10-stage clock distribution network with  $\lambda = 3$  at each stage, the minimum theoretical power used is  $1.50C_L V_{DD}^2 f$ , of which  $1.48C_L V_{DD}^2 f$  is dissipated in the final four stages. To neglect the clock distribution network, then, is to omit a major portion of the dissipated power.

Even Eq. (13.4) is insufficient for an actual clock distribution network, however, because of omitted parasitics. *Crossover* or *direct-path current*, which flows directly from  $V_{DD}$  through the PFET and the NFET to  $V_{SS}$ , has been neglected and can amount to 10–20% of the total current for each transition. Crossover current can be reduced by driving faster input edges for the same fanout [5],[6]. Also omitted from the equations above are diffusion and interconnect capacitance. Diffusion capacitance can be reduced by sharing diffusions between polysilicon fingers, and interconnect capacitance can be decreased by locating large final-stage clock drivers near their loads [9]. The last omitted nonideal term is power dissipated from leakage current [5],[6]. Accounting for all the parasitics that contribute to power dissipation clearly requires thorough bookkeeping.

In the quest for an ideal clock with perfect performance and minimum power, a couple of design dilemmas already appear. The clock on a microprocessor is typically one of the top consumers of power, typically in the range of 40% of the total microprocessor power [10]. If power consumption is clipping performance by constricting the maximum

operating frequency, then reducing clock power can directly benefit performance. A related problem is that power reduction methods often conflict directly with performance objectives. Tolerating slower edges allows driver size reduction that can help lower upstream power dissipation by decreasing predriver sizes, as long as crossover current does not dominate. On the other hand, skew can often be reduced if faster edges are used. Ultimately, compromises between performance and power must be made [9].

Reliability and immunity to variation are two other critical objectives of an ideal clock. An ideal clock distribution network achieves ideal performance and power characteristics in all process corners, and retains them for the life of the microprocessor in the face of all power supply, temperature, and data-dependent variations. Steps to ensure reliability are mostly specified by the foundry, but providing variation tolerance must be considered in the clock design. Indeed, of all the ideal clock characteristics listed, variation immunity is perhaps the most difficult objective to approach in practice [11].

Other traits can be added to the definition of an ideal clock, including constant rise and fall times, or minimum area. The list should also include product-specific objectives. Identifying all the characteristics that define an ideal clock is useful both for making design choices and for avoiding design oversights.

### 13.2.2 Evaluation of Objectives

After the desired clock attributes have been identified, the next step in making design decisions is ranking the importance of objectives. This evaluation requires determining the effect that the clock distribution network has on microprocessor performance, power, yield, and schedule. A complete assessment is difficult before the design is done, but a partial evaluation is possible. The performance cost of global long-term jitter is one example. If  $t_{-jitter}$  is jitter that subtracts from the period, then the effective or usable clock period  $T_{eff}$  is

$$T_{eff} = T - t_{-jitter} \quad (13.5)$$

The clock frequency including jitter is

$$f = \frac{1}{T} = \frac{1}{T_{eff} + t_{-jitter}} \quad (13.6)$$

The performance loss is given by the change in frequency with jitter,  $\Delta f$ , divided by the frequency with no jitter,  $f_{nojitter}$ . Assuming jitter is random and independent of frequency, the performance loss is then

$$\frac{\Delta f}{f_{nojitter}} = \frac{f_{nojitter} - f}{f_{nojitter}} = \frac{(1/T_{eff}) - (1/T_{eff} + t_{-jitter})}{1/T_{eff}} = \frac{t_{-jitter}}{T_{eff} + t_{-jitter}} = \frac{t_{-jitter}}{T} \quad (13.7)$$

That is, the percentage decrease in frequency caused by jitter,  $100\% \times \Delta f/f_{nojitter}$ , is equal to the percentage of jitter,  $100\% \times t_{-jitter}/T$ . For example, if jitter is  $\pm 15\%$ , then the frequency must be decreased by 15% compared to the ideal case with no jitter.

Skew is not as generally described as jitter because it is not globally unique. Different triplets have different skew. The standard pessimistic approach, however, is to analyze the global worst-case skew and apply that to every triplet for CAD-tool timing analysis. This is conservative because the worst-case skew may not exist across any triplet in the microprocessor, but this is also the safest approach [9],[12],[13].

Duty cycle, however, is a global issue, and a 50% duty cycle is the most commonly desired objective. A design methodology can allow and plan for a non-50% duty cycle, but in either case duty cycle has to be consistent to not penalize performance. The standard approach to maintaining a good duty cycle in the distribution network is to group the clock buffer chain into consecutive pairs. Within each pair, the beta ratio and fan-out are kept the same. The reason this helps avoid duty-cycle corruption is that a rising edge at the beginning of a clock chain always turns on the same alternating pairs of NFETs and PFETs. An initial falling edge turns on the opposite transistors in the chain. By keeping the beta ratio and fan-out constant for each pair of inverters, and ignoring variation effects, the NFET–PFET combination will have the same delay as the PFET–NFET combination. If the delays through these paths are different, duty cycle is corrupted and performance is unnecessarily lost [14],[15].

Determining the performance cost of a degraded duty cycle is similar to the analysis of jitter. The effective clock period  $T_{eff}$ , assuming equivalent critical paths exist in both clock phases, is

$$T_{eff} = 2\theta_{min} \quad (13.8)$$

where  $\theta_{min}$  is the minimum phase of the two phases shown in Fig. 13.1, that is,

$$\theta_{min} = \min(\theta_A, \theta_B) \quad (13.9)$$

The performance loss due to poor duty cycle is

$$\frac{\Delta f}{f_{50\%dc}} = \frac{f_{50\%dc} - f}{f_{50\%dc}} = \frac{(1/T_{eff}) - (1/T)}{1/T_{eff}} = 1 - \frac{T_{eff}}{T} = 1 - 2 \frac{\theta_{min}}{T} \quad (13.10)$$

where  $f_{50\%dc}$  is the clock frequency with a 50% duty cycle. Equation (13.10) shows that the clock frequency penalty is twice the duty-cycle degradation. For example, the clock frequency must be reduced 10% if the duty cycle is changed from 50% to 45%.

Assessing the performance cost of design objectives is only part of the analysis. Power trade-offs should be analyzed, too. Dissipating excessive clock power has two penalties: it can adversely sway package and power supply costs, and it can increase on-die power-supply variation. The cost of a clock that uses inordinate power is perhaps a more expensive package or power source, which may be unacceptable depending on the product specifications. Adding power-supply noise can introduce multiple modes of danger including timing variation, charge injection, and noise-margin reduction, all of which can result in functional failure [1]. Decreasing power dissipation is therefore usually a prime objective in the design of the clock distribution network.

The last item of evaluation discussed here is tolerance to process and environmental variation, which, as mentioned before, is critical. A common problem is that many clock designs start with zero or near-zero skew under ideal conditions, but with process, voltage, and temperature variations have large skew and jitter values that are difficult to predict and locate in debug [14]. A closely related objective is tolerance to load variation. A clock design that is sensitive to load variation cannot accommodate frequent or late design changes without delaying the schedule. If a clock design requires precise matching of widely dispersed clock drivers or loads, then it might be fatally flawed and not produce acceptable results in practice.

### 13.3 IMPLEMENTATION

Clocks can be divided into two structures: the predriver network, and the final stages. Predriver stages are controlled and managed internal to the clock design. Final stages must consider circuits external to the clock drivers. As a result, it is usually easiest to design the final stages first because they must be compatible with external-circuit constraints.

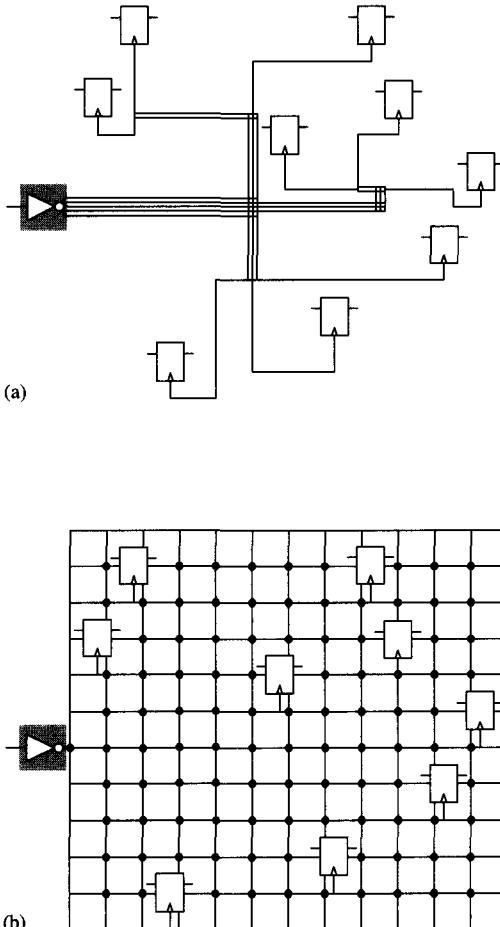
#### 13.3.1 Final Stage Drivers

The optimum clock driver configuration depends on the size and location of final loads, as well as on available locations open for clock driver placement. Large clock drivers clash with some circuit structures because of wire or device congestion. This sometimes conflicts with the principle that for low skew, fast edges, and low power, clock drivers should be located close to large loads [9]. The nonexcluded areas provide regions to be considered for final clock drivers, but compromises may be necessary to locate clock drivers near the largest loads.

Final-stage clock driver placement also depends on the method of distributing the clock signal. Figures 13.4(a) and (b) are diagrams of two common options for delivering clock from a final-stage driver to multiple loads. Figure 13.4(a) shows an *RC-matched tree* where the resistance–capacitance delay to each load is matched. Interconnect that branches in two directions can be twice as wide prior to the branch if the flip-flop loads are identical. Alternatively, Fig. 13.4(b) shows a clock distributed by a grid. Delay from the final driver to each load is not delay matched, but if the grid is dense and the distances are limited, skew can be acceptably small. This approach uses more power because of “excess” interconnect, but the grid is insensitive to load placement and thus is forgiving to late circuit changes [16]. An RC-matched distribution network for the final stage is the best choice when there are only a few loads or when the loads are arrayed. A grid should be considered when the load placement is dense or random, or when the variations are expected to be large.

Another principal consideration in clock driver placement is load balance. Low skew is easier to attain if the die is divided into regions with similar clock loads. How large the regions are depends on the number of subsequent driver stages [13].

Besides determining where the clock drivers are located, another design decision is determining what type of driver to use. *Gated clocking*, also called *conditional clocking*, is a common technique that can be used to reduce power in the final stages. Instead of an inverter, the clock drives a logic gate, such as a NAND or NOR gate. Other signals control whether downstream circuitry is clocked. Gated clocks are commonly used in RAM arrays and data paths so that only active sections are clocked, thus reducing both clock and data path power dissipation. There are drawbacks, however, to using gated clocks. First, a gated clock presents greater clock load than an unconditional clock with equivalent drive. Second, the effective gate capacitance of a gated clock is greater when selected than when not selected, which causes clock jitter in the form of data-dependent loading. Third, CAD timing and logic verification tools must be capable of handling the additional complexity of gated clocks. Nonetheless, if these factors are considered and exclusivity is exploited, gated clocks can be a tremendous power savings technique [10].

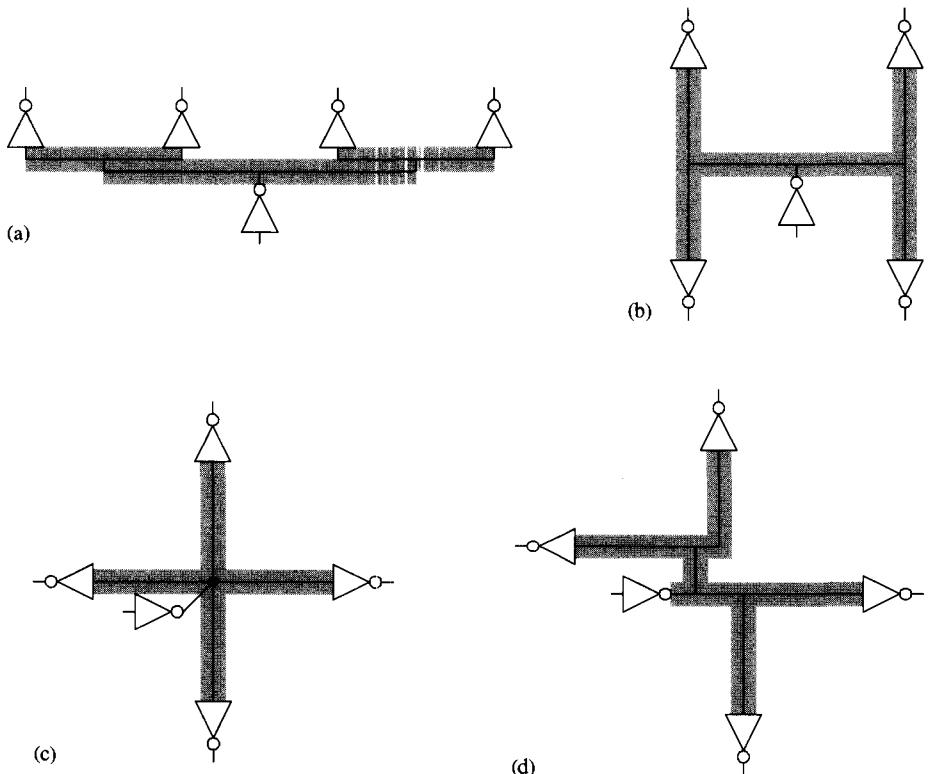


**Figure 13.4** Examples of final-stage clock distribution using (a) an RC-matched tree, and (b) a grid.

### 13.3.2 Predriver Network

Once the candidate locations for final drivers have been identified, predriver network design can begin. The simplest clock network is an undistributed clock network, that is, a cascade of inverters, also called a *buffer chain*. There are no internal synchronization requirements for a buffer chain because skew is meaningless for a single point. A buffer chain also offers the most power-efficient ramping possible. Unnecessary stages are not desirable, however, because a buffer chain is still subject to jitter, particularly from power-supply noise. Nonetheless, because skew is not an issue, a buffer chain is a useful structure for a predriver network [5].

When the clock must be distributed because the final stages are distributed, skew is a concern for the predriver stages. Figures 13.5(a)–(d) shows various common clock distribution networks that attain zero skew under ideal conditions. Figure 13.5(a) is often called a *binary tree* because at each branch point it splits into two equidistant interconnect segments. Figure 13.5(b) shows an *H tree*, named for the shape of the interconnect, as is the structure in Fig. 13.5(c), an *X tree*. Sometimes the term “H tree” is loosely applied to any distribution network with rotational or mirror symmetry.



**Figure 13.5** Diagrams of (a) a one-dimensional binary tree, (b) an H tree, (c) an X tree, and (d) an arbitrary RC-matched tree.

These are all forms, however, of the RC-matched tree, introduced in Fig. 13.4(a) for final-stage distribution and shown again in Fig. 13.5(d) for predriver stages. Each distribution network seeks to match delays to each receiving driver (or latch or flip-flop) by using equivalent receiving loads and interconnect segments [4].

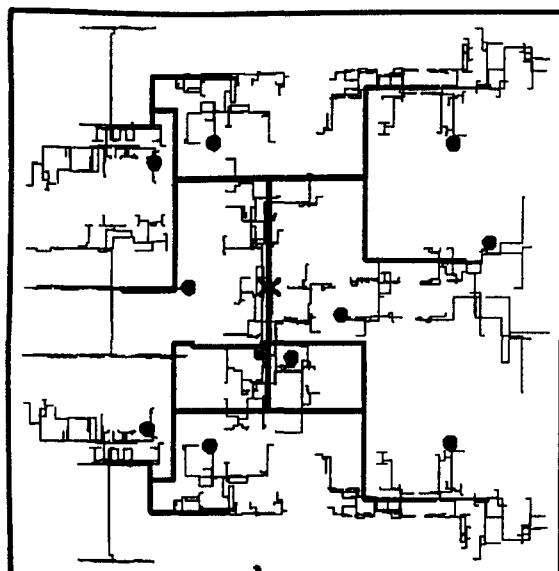
A more general method, which is not illustrated in Fig. 13.5, is to match the overall delay for the load and interconnect combination. That is, for unequal loads the interconnect resistance is varied so that under ideal conditions there is no skew between the receivers. Resistance can be controlled by modifying the width, length, or layer of the interconnect. The weakness to this generalized approach is that it does not rely on symmetry and is therefore subject to additional variations. Changes to the process, even speed improvements, can manifest skew because an imbalance can form between the different delay components [14],[15].

The choice of predriver tree depends on the possible locations for final-driver stages. If it is possible, there are advantages to arranging the final driver stages as a one-dimensional strip. This is often a convenient floorplan shape because it can be placed along an edge of the chip, or between rectangular execution units, and generally leaves other circuitry and wires undisturbed. Perhaps the main concern is layout allowances for signals that must pass through the driver cell in the transverse direction. Other advantages to a one-dimensional arrangement are that it allows the final output drivers to be connected, which integrates variation, and easily accommodates a binary tree predriver network.

If the final drivers are widely dispersed, a two-dimensional predriver network is the most power-efficient approach. The primary disadvantage of widely dispersed drivers is skew introduced by process and environmental variations between the drivers. There are benefits to widely scattered drivers, however, including uniform power and heat dissipation on the die. For a gridded distribution network with widely dispersed clock drivers, low skew is generally easier to attain because the maximum distance of any load to the nearest driver is reduced. Regarding the choice of driver locations, the preferred two-dimensional predriver distribution network will depend on the symmetries suggested by the final-stage driver positions.

### 13.3.3 Examples

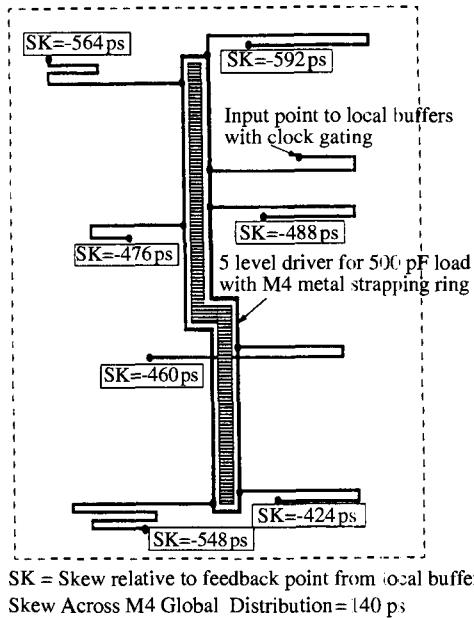
An example of RC-matched trees implemented in a microprocessor is shown in Fig. 13.6. In this clock distribution network, the die is divided into ten balanced-load “sectors.” A predriver network distributes the clock to sector drivers located at the dots in Fig. 13.6. Sector drivers then redistribute the clock to another 580 drivers using another RC-matched tree. This type of distribution network is common for microprocessors [4],[13],[17],[18].



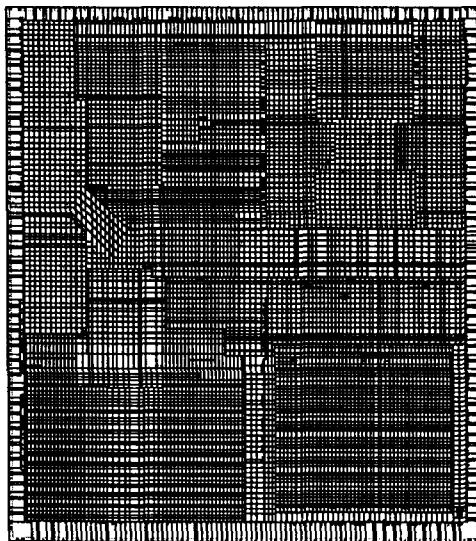
**Figure 13.6** An RC-matched distribution network for a microprocessor [13].

Figure 13.7 shows another implementation of an RC-matched tree, in this case for a test chip, but the idea is extendable to a microprocessor. In this approach, a one-dimensional distributed clock driver drives many nominally equidistant “serpentine” offshoots that terminate into equivalent loads. This approach has the advantages of using a one-dimensional predriver network and an RC-matched distribution network for the final stage, and accomplishing this for a two-dimensional distribution of loads [12].

An example of a gridded final-stage network is shown in Fig. 13.8. Shown is the global clock grid, a single node that spans the entire die. Not shown is the predriver distribution network comprised of X, binary, and RC-matched trees of increasingly larger



**Figure 13.7** Diagram of a one-dimensional clock predriver (with bend) and sample serpentine distribution wires for a test chip [12].



**Figure 13.8** Layout plot of gridded clock node for a microprocessor.

driver stages, the final stage of which drives the grid in parallel. Additional driver stages, including gated clocks, tap off this grid and are scattered across the microprocessor. The main disadvantage to a gridded final stage is higher power dissipation because the grid must be driven in addition to the gate load. The grid shown in Fig. 13.8 uses 3% of two upper-level metal layers and is responsible for two thirds of the total global clock load. The advantages to using a grid are that the skew is consistent and insensitive to both data-dependent and nonuniformly distributed gate load. In addition, driver variations for a grid are averaged because the outputs of the final-stage drivers are tied together [9],[19].

### 13.3.4 Trends

An emerging trend in clock distribution network design is *active skew management* of multiple clock domains [18],[21],[22]. Active skew management is the synchronization of separate clock domains, usually by delay-locked loops (DLLs). Not only is this trend expected to continue, the expected tendency is also that microprocessors will use more and smaller clock domains in the future to better control skew [23]. A disadvantage to this approach is that DLLs introduce additional jitter and offset latencies in the clock signals. These timing problems can be largely controlled, however, with careful circuit techniques.

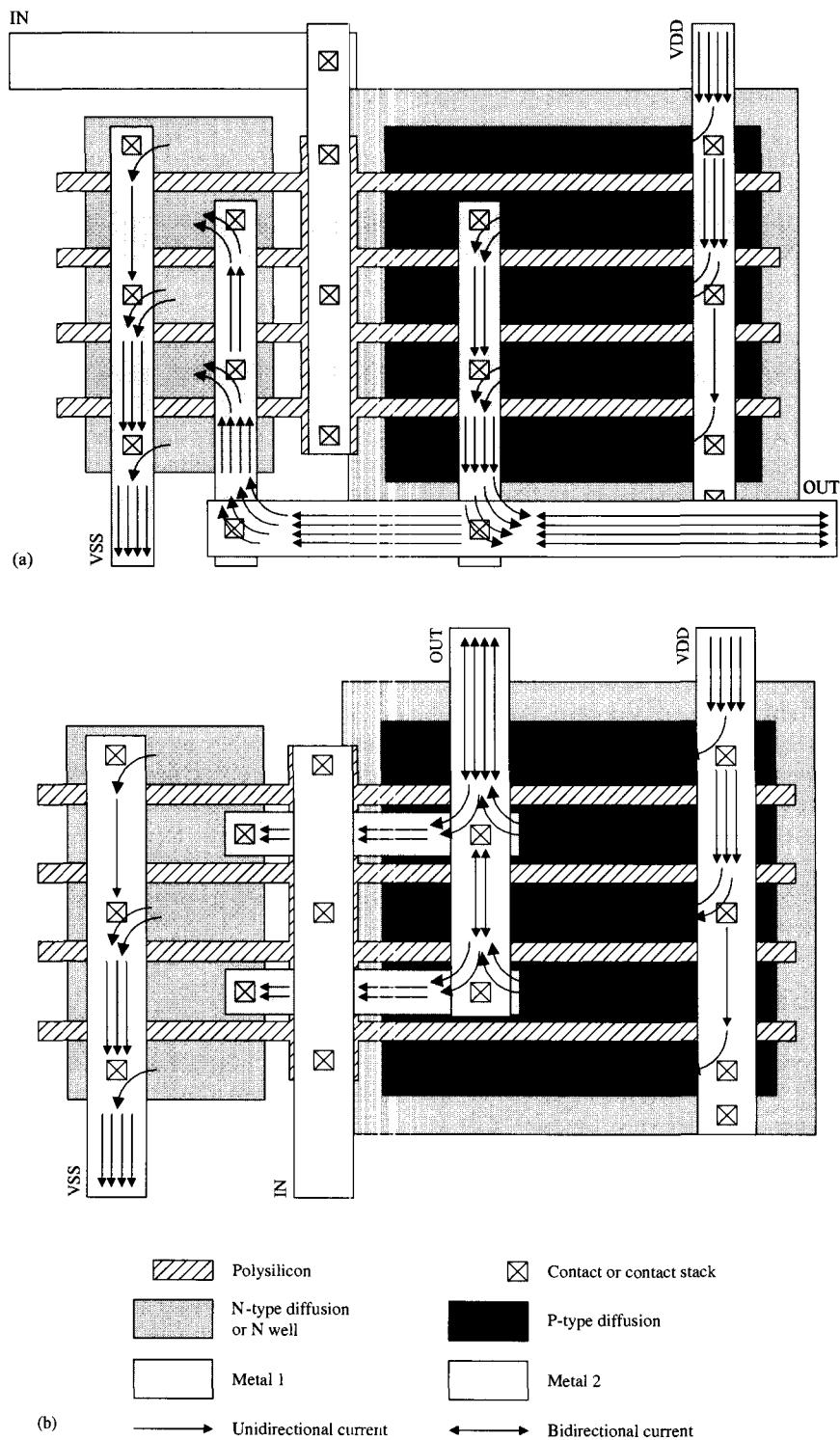
To get a good perspective of clock distribution network design, it is also useful to consider future directions as indicated in the literature, in addition to current commercial microprocessor trends and implementations. Research areas such as resonant clocking and energy recovery [20] present innovative approaches to clock design, and also reveal the most pressing upcoming problems. These techniques attack the issue of power dissipation because it threatens to constrict performance on future microprocessors. Thus, developing trends are worth investigating both for implementation ideas and for anticipating problems.

Another interesting research example is clock distribution in the package, which, like resonant clocking, has been successfully demonstrated in test implementations [24]. Package interconnect is 100 to 10000 times less resistive than on-chip interconnect, but has roughly the same or less capacitance per unit length. Clock distribution in the package exploits this parameter advantage to distribute a well-synchronized clock to distant regions of the chip, potentially solving a very difficult problem for future implementations. The final stages are distributed on the chip by conventional means. An alternative implementation uses a separate clock-generator chip to provide synchronized clocks to distant clock domains via the package [21]. Either way, wafer probe can be accomplished with a custom-designed probe card that distributes the clock, emulating the package. The remaining issues to successful microprocessor implementation are achieving low skew while providing electrostatic discharge protection and transmission line termination.

## 13.4 CLOCK DRIVER LAYOUT

Contemporary microprocessors rely on hundreds or thousands of clock drivers [1],[13], [18]. Oversight in the clock driver layout, therefore, are multiplied many times. Flawed layout can therefore unnecessarily jeopardize long-term performance, manufacturing yield, or can deteriorate layout productivity. Early analysis and forethought regarding the clock driver layout can benefit reliability, schedule, and yield.

Figures 13.9(a) and (b) show two versions of a clock driver. A brief orientation to the layout fill codes and transistor structure is helpful before discussing specific layout issues. Polysilicon is the gate terminal of a transistor any place that it overlays an N or P diffusion. The separated diffusions form the source and drain terminals of the transistor. A P<sup>-</sup> substrate is used in this example, so the PFET sits in an N well. The transistor back-gate terminal is the substrate (VSS) for NFETs, and is the N well (VDD) for PFETs. Diffusions, polysilicon, Metal 1, and Metal 2 are isolated from one another except by contacts between adjacent layers. To summarize, Figs. 13.9(a) and (b) show the layout to identically sized inverters with identical terminal connections, even though



**Figure 13.9** Layout plots of clock driver cell with (a) excessive electromigration risk, and (b) improved electromigration reliability.

the layer connections are different. These minor differences, however, can have a major effect on reliability.

First consider the further similarities in the clock driver layouts shown in Figs. 13.9(a) and (b). The input node, IN, is connected to four parallel polysilicon NFET fingers on the left, and four parallel polysilicon PFET fingers on the right. Polysilicon gates are driven from between the NFET and the PFET to keep polysilicon finger lengths and thus RC delay along the fingers acceptably short. The output node, OUT, is shared between two polysilicon fingers so that the diode capacitance between drain and back-gate terminals is minimized. The drivers in Figs. 13.9(a) and (b) should perform electrically the same.

### 13.4.1 Electromigration

At question is whether the two drivers in Figs. 13.9(a) and (b) will retain their fidelity over time. The primary reliability issue for clock drivers is *electromigration*. Electromigration is the movement of conductor material caused by the net flow of electrons. It can eventually cause both shorts and opens, although opens are more likely, and is a long-term hazard for interconnect that carry large currents. Electromigration is proportional to current density, so wider and thicker interconnect is more tolerant assuming the same dielectric is used. Also, electromigration is worse for unidirectional current than for bidirectional current because the bidirectional electron motion has a restoring effect on the metal atoms [25],[26]. Certain layout techniques can be used, however, to reduce the electromigration risk.

First compare the VDD and VSS nodes in Figs. 13.9(a) and (b). In Fig. 13.9(a), power and ground currents are delivered with Metal 1. In Fig. 13.9(b), VDD and VSS currents are delivered using Metal 2 and a contact stack in each diffusion. This has the positive effect of sending unidirectional current through wider and usually thicker Metal 2, instead of Metal 1. Next compare the output nodes in Figs. 13.9(a) and (b). In Fig. 13.9(a), all the current on OUT is unidirectional for some Metal 1 and Metal 2 interconnect segments, and for all the contacts shown. In Fig. 13.9(b), the NFET and PFET output nodes are connected with Metal 1, an important distinction. As a result, Metal 1 segments and Metal 1-to-Metal 2 contacts have to support only half the unidirectional current as for the layout in Fig. 13.9(a). In addition, Metal 2 and Metal 1-to-Metal 2 contacts on OUT in Fig. 13.9(b) have bidirectional currents. The summary to this comparison is that the layout in Fig. 13.9(b) is much more reliable. The driver in Fig. 13.9(a) can be protected from electromigration, but it will require wider interconnect and more contacts than the driver in Fig. 13.9(b).

### 13.4.2 Productivity

An undeniably important issue for layout is productivity. Characteristics that make a clock driver easier to place and connect are layout attributes that increase productivity. One of these characteristics is *porosity*. Porosity can be defined as the percentage of each metal layer that is available for signal interconnect unrelated to the clock. Clock drivers that have high porosity have fewer restrictions on driver placement in relation to other circuits. Therefore, porosity improves productivity, not because it eases the clock layout, but because it eases the circuit layout in the clock driver region.

Another clock-layout productivity characteristic is (size) *tunability*, a qualitative measure of how easy it is to increase or decrease the driver size. A fundamentally

difficult layout challenge is to increase a device size in existing layout where there is no additional room. Since on-chip *decoupling capacitors*, capacitors connected between VDD and VSS, are usually placed near large clock drivers [1], one technique that improves tunability is to design the clock driver with the same footprint as the decoupling capacitor. Increasing the driver size is then simply a matter of swapping clock driver cells with decoupling capacitors. This creates a clock driver layout whose net size is easily tunable and enables relatively easy changes.

### 13.4.3 Yield

Yield is another issue that can benefit from proper clock driver layout techniques, although the correlation is probably weak. Generally speaking, yield is inversely proportional to area, so compact clock driver layout can positively influence yield [26]. Sharing diffusions between polysilicon fingers and using long fingers are two ways to increase area efficiency. Redundant contacts can also help yield, although the benefit is probably small. These techniques are easy, however, so there is generally no reason to neglect them.

## 13.5 VARIATION

An unavoidable aspect of microprocessor clock design is verification. Proper verification means more than ensuring that the schematic design is represented in the final product, it also means ensuring the design is manufacturable. To do this, variations must be accounted for and included in the verification simulations. This section, therefore, discusses the four types of variations that affect clock performance: process, power supply, temperature, and data-dependent noise. The causes of each type of variation are discussed below along with the design and technology steps that can be taken to mitigate their effects.

### 13.5.1 Process

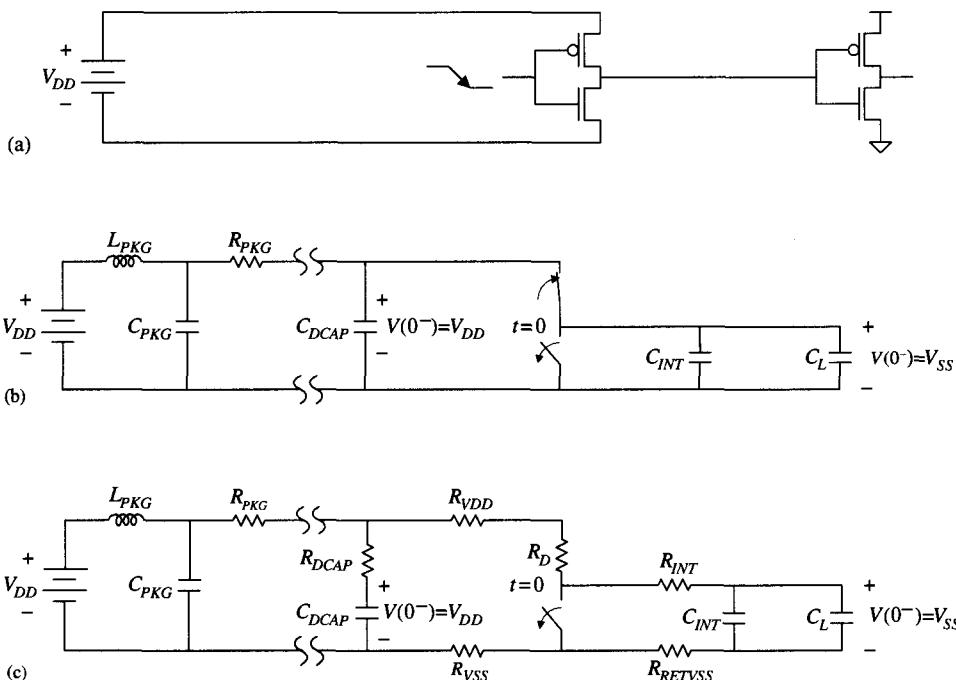
Process variations are caused by deviations of material dimensions, interfaces, or composition. The most important process variations with respect to skew are interlayer-dielectric thicknesses, interconnect thicknesses, and channel-length variations [27]. In addition, threshold voltage variation is expected to be an issue for future technology generations as power supplies continue to scale downward [3],[28].

Reducing process variation would seem to be the complete responsibility of the foundry, but steps can be taken in the design to help. Of all the mask layers, polysilicon is the most critical because it has the most influence on the device performance, and because it has the smallest feature size of any layer. It is therefore usually one of the primary targets for reducing process variation. Polysilicon density and orientation can locally influence both the effective channel length and channel-length variation, and both density and orientation are under the control of the designer. One way to keep the polysilicon density constant is having a single elemental layout cell that is used in parallel to create each clock driver. Channel-length variation can then be reduced by orienting that elemental clock driver cell the same way everywhere on the die [11]. As an additional advantage, other methods to reduce process variation as suggested by the foundry are easiest to incorporate in the clock driver layout if an elemental cell is used.

### 13.5.2 Power Supply

On-chip power supply noise can generally be divided into low-, mid-, and high-frequency terms, although the specifics depend on the technology [29]. Low-frequency components are from voltage drops proportional to the total current in a section and can be reduced by adopting C4 bump packages. Wire bond connections are only made along the periphery of the die, whereas C4 bumps cover the die and reduce the maximum voltage drop to the center. Mid-frequency terms are caused by package-die interactions, and are also more severe for wire bonds because they have greater inductance than C4 bumps [30]. Another technology solution that reduces low-frequency and mid-frequency noise is to use reference planes, that is, die layers dedicated exclusively to  $V_{DD}$  or  $V_{SS}$  [31]. A design solution that simply avoids power-supply variation, instead of reducing it, is to place clock drivers along the perimeter where voltage drop is generally smallest [22]. The last term, the high-frequency component of power-supply noise, is peaked at the clock frequency and is due to the clock itself and circuit activity triggered by the clock. The main design solution for reducing high-frequency noise is adding on-chip decoupling capacitors [29],[32].

Decoupling capacitors can take a lot of area, however, so determining how much is needed is an important design decision. Figures 13.10(a–c) show three levels of approximation for modeling the effects of decoupling capacitance. Figure 13.10(a) shows a naïve model that assumes all current is directly provided by the power supply.



**Figure 13.10** (a) Circuit diagram of clock driver pair ignoring package impedances, and equivalent circuit diagrams of clock driver pair including (b) on-chip decoupling capacitance, and (c) on-chip resistance and decoupling capacitance.

This is a faulty assumption for microprocessors because printed circuit board and package impedances are too great to provide currents at the edge rates needed on the die [29]. Figure 13.10(b) shows a more reasonable approximation in which the current for a switching event is provided locally by capacitance between  $V_{DD}$  and  $V_{SS}$ , and contributions from the off-chip power supply are ignored. Figure 13.10(c) shows an extension to this model by including resistance components. Although flawed, the model in Fig. 13.10(a) is perfectly appropriate for analyzing small gates because adequate nearby capacitance exists between  $V_{DD}$  and  $V_{SS}$  to supply small gate currents so that the model appears correct. For large clock drivers, however, explicit decoupling capacitors are probably needed.

The capacitor divider circuit in Fig. 13.10(b) can be used to provide a first-order estimate of how much decoupling capacitance is needed near the clock driver. Since the initial charge on the decoupling capacitor is equal to the final charge shared by the load, interconnect, and decoupling capacitor, then

$$C_{DCAP}V_{DD} = (C_{DCAP} + C_{INT} + C_L)(\eta V_{DD}) \quad (13.11)$$

where  $C_{INT}$  is the effective interconnect capacitance,  $C_L$  is the effective load capacitance, and  $\eta$  is the ratio of the final voltage to  $V_{DD}$ . Effective capacitances include coupling and Miller effects. From Eq. (13.11), the amount of decoupling capacitance needed is

$$C_{DCAP} = (C_{INT} + C_L) \left( \frac{\eta}{1 - \eta} \right) \quad (13.12)$$

Assuming that a 10% drop in local voltage can be tolerated and is acceptable, then  $\eta = 0.9$  and  $(\eta/1 - \eta) = 9$ . If there is a fan-out of 3 along the clock distribution chain, then this means that nearby decoupling capacitance should be equivalent to 27 times the gate capacitance of each clock driver.

The model in Fig. 13.10(b), however, ignores delay effects that can make the capacitance less effective or ineffective. To determine how far the decoupling capacitors can be placed from the driver, delays through the power grid need to be calculated. Figure 13.10(c) shows a one-dimensional model of  $V_{DD}$  and  $V_{SS}$  power grids, which, although simplistic and omits inductance, may be adequate if the parameters are conservatively approximated. The best analysis more completely models the two-dimensional delays of the power and ground grids including RLC delays [32]. To be effective, charge from the decoupling capacitance must be supplied in less time than the clock rise or fall time. As an alternative to thorough modeling, a guarded approach is to apply Eq. (13.12) to determine how much decoupling capacitance is needed assuming a conservatively high fan-out for each elemental driver cell. The decoupling capacitor can then be integrated in layout with the clock driver cell so that adequate nearby capacitance is always assured [1].

### 13.5.3 Temperature

Another type of variation to consider is temperature, which is in the form of temperature gradients and the global operating temperature. Not only should simulations be performed at the high- and low-temperature corners, but expected worst-case temperature variations between different clock drivers should be modeled because it can be a major cause of skew [27]. Fortunately, temperature usually varies slowly enough to be ignored in simulations of jitter in the clock distribution network.

Analyzing temperature gradients on the die is difficult without sophisticated CAD tools, but approaches can be used to mitigate temperature variation effects in the clock design even without this analysis. As mentioned before, clock drivers can be widely dispersed to reduce self-heating temperature gradients [1]. Another possible design improvement is to locate drivers along the perimeter of the chip where temperature (and voltage) typically does not vary as much as when surrounded by other circuitry [22].

### 13.5.4 Data-Dependent Noise

Capacitive coupling on small clock nodes is a concern that should be considered in the timing verification. Coupling on large clock nodes, on the other hand, is generally not a problem for two reasons: first, the clock node is driven by low-impedance output drivers and so should be firmly held to the power rails, and second, the clock node capacitance is large and acts as ballast against aggressors. Pathological cases can exist, of course, but generally coupling is only an issue for small clock nodes. Fixes to small clock nodes include shielding clock wires, making the small clock into a big clock instead or, if it is available in the technology, including reference planes [31].

Another contributor to skew is data-dependent clock load. Under some circumstances, it can even be the largest component of skew [27]. Gated clocks and most latches exhibit clock load variation as a function of the data. The three main choices for dealing with data-dependent clock load are to balance it, to reduce it, or to increase the constant clock load. An example of balancing it is to use dual-rail data latches [20]. Reducing data-dependent clock load means replacing gated clocks with unconditional clocks, which also has the effect of increasing the constant clock load. Another method of increasing the constant clock load is to adopt a grid in the final-stage clock distribution network. Each of these methods represents a potentially major implementation change, so the first step is identifying the severity of the problem. This can be done by simulating the clock distribution network with all clock loads at maximum capacitance and comparing to a skew simulation with all clock loads minimized. Worst-case jitter at any location is given by the difference in skew between the simulations. Once the problem is characterized, the appropriate measures can be taken.

## 13.6 CONCLUSION

Clock distribution networks are logically simple circuits, but performance demands, enormous equivalent circuits, and non-scalability make clock design a challenging, even intimidating task. There are many issues and trade-offs in clock design, and therefore many chances to make mistakes. Errors can be avoided, however, if the design is approached methodically. First, objectives should be clearly defined and ranked to establish design specifications. Second, based on the objectives and the microprocessor floor plan, final-stage clock driver locations can be proposed, and a predriver distribution network designed. Third, for reliability, productivity, and yield reasons, a clock driver layout cell should be carefully designed. Other than running the CAD tools, the last design task is to identify variation effects, reduce them when possible, and include them in the verification analysis. If these procedures are followed, the opportunities for successfully designing a microprocessor clock distribution network are greatly increased.

## REFERENCES

- [1] P. E. Gronowski, et al., "High-performance Microprocessor Design," *IEEE J. Solid-State Circuits*, vol. 33, no. 5, pp. 676–686, May 1998.
- [2] W. F. Brinkman and M. R. Pinto, "The Future of Solid-state Electronics," *Bell Labs Technical J.*, Autumn 1997, pp. 57–75.
- [3] P. M. Zeitzoff, "Front-end Trends, Challenges, and Potential Solutions for the 180–100 nm IC Technology Generations," *Semiconductor Fabtech*, vol. 10th edition, pp. 275–282, 1999.
- [4] E. G. Friedman, "Introduction". In E. G. Friedman (ed.), *Clock Distribution Networks in VLSI Circuits and Systems*, pp. 1–36, IEEE Press, New York, 1995.
- [5] N. H. E. Weste and K. Eshraghian, *Principles of CMOS Design, A Systems Perspective*. Addison-Wesley, MA, Reading, 1985.
- [6] J. M. Rabaey, *Digital Integrated Circuits, A Design Perspective*. Prentice-Hall, Englewood Cliffs, 1996.
- [7] F. Herzl and B. Razavi, "A Study of Oscillator Jitter Due to Supply and Substrate Noise," *IEEE Trans. Circuits Systems—II: Analog Digital Signal Processing*, vol. 46, no. 1, Jan. 1999.
- [8] J. A. McNeill, "Jitter in Ring Oscillators," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, June 1997.
- [9] D. W. Bailey and B. J. Benschneider, "Clocking Design and Analysis for a 600-MHz Alpha Microprocessor," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, pp. 1627–1633, Nov. 1998.
- [10] M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power Considerations in the Design of the Alpha 21264 Microprocessor," *Proc. Design Automation Conf.*, San Francisco, 1998, pp. 726–731.
- [11] N. Nekili, Y. Savaria, and G. Bois, "Design of Clock Distribution Networks in Presence of Process Variations," *Proc. 8th Great Lakes Symp. VLSI*, Lafayette, LA, 1998, pp. 95–102.
- [12] I. A. Young, M. F. Mar, and B. Brushan, "A 0.35  $\mu\text{m}$  CMOS 3-880 MHz PLL N/2 Clock Multiplier and Distribution Network with Low Jitter for Microprocessors," *ISSCC 1997 Dig. Technical Papers*, Feb. 1997, pp. 330–331.
- [13] P. J. Restle, K. A. Jenkins, A. Deutsch, and P. W. Cook, "Measurement and Modeling of On-chip Transmission Line Effects in a 400 MHz Microprocessor," *IEEE J. Solid-State Circuits*, vol. 33, no. 4, pp. 662–665, Apr. 1998.
- [14] M. Shoji, "Elimination of Process-dependent Clock Skew in CMOS VLSI," *IEEE J. Solid-State Circuits*, vol. SC-21, no. 5, pp. 875–880, Oct. 1986.
- [15] J. P. Fishburn, "Clock Skew Optimization," *IEEE Trans. Computers*, vol. 39, no. 7, pp. 945–951, July 1990.
- [16] D. Dobberpuhl, et al., "A 200 MHz 64 b Dual Issue CMOS Microprocessor," *IEEE J. Solid-State Circuits*, vol. 27, no. 11, pp. 1555–1567, Nov. 1992.
- [17] C. F. Webb, et al., "A 400 MHZ S/390 Microprocessor," *ISSCC 1997 Dig. Technical Papers*, Feb. 1997, pp. 168–169.
- [18] J. Alvarez, et al., "450 MHz PowerPC™ Microprocessor with Enhanced Instruction Set and Copper Interconnect," *ISSCC 1999 Dig. Technical Papers*, Feb. 1999, pp. 96–97.
- [19] H. Fair and D. Bailey, "Clocking Design and Analysis for a 600-MHz Alpha Microprocessor," *ISSCC 1998 Dig. Technical Papers*, Feb. 1998, pp. 398–399.
- [20] W. Athas, et al., "AC-1: A Clock-powered Microprocessor," *Proc. Low Power Electronic Design*, 1997, pp. 328–333.
- [21] L. Cao and J. P. Krusius, "A Novel 'Double-decker' Flip-chip/BGA Package for Low Power Giga-hertz Clock Distribution," *1997 Electronics Components Technology Conf.*, Seattle, WA, 1997, pp. 1152–1157.
- [22] G. Geannopoulos and X. Dai, "An Adaptive Digital Deskewing Circuit for Clock Distribution Networks," *ISSCC 1998 Dig. Technical Papers*, Feb. 1998, pp. 400–401.
- [23] M. Horowitz, "Clocking Strategies in High Performance Processors," in *1992 Symp. VLSI Circuits Dig. Technical Papers*, Seattle, WA, 1992, pp. 50–53.

- [24] Q. Zhu and S. Tam, "Package Clock Distribution Design Optimization for High-speed and Low-power VLSI's," *IEEE Trans. Components Packaging Manufacturing Technology—Part B*, vol. 20, no. 1, pp. 56–63, Feb. 1997.
- [25] R. V. Hesketh, "Electromigration: The Electron Wind," *Phys. Rev. B*, vol. 19, p. 1727, 1979.
- [26] W. J. Bertram, "Yield and Reliability". In S. M. Sze (ed.) *VLSI Technology 2nd Edition*, pp. 612–655. McGraw-Hill, New York, 1988.
- [27] P. Zarkesh-Ha, T. Mule, and J. D. Meindl, "Characterization and Modeling of Clock Skew with Process Variations," *IEEE 1999 Custom Integrated Circuits Conf.*, 1999, pp. 441–444.
- [28] J. D. Meindl, et al., "The Impact of Stochastic Dopant and Interconnect Distributions on Gigascale Integration," *ISSCC 1997 Dig. Technical Papers*, Feb. 1997, pp. 182–183.
- [29] W. D. Becker, et al., "Modeling, Simulation, and Measurement of Mid-frequency Simultaneous Switching Noise in Computer Systems," *IEEE Trans. Components Packaging Manufacturing Technology—Part B*, vol. 21, no. 2, pp. 157–163, May 1998.
- [30] L. Zu, et al., "Improving Microprocessor Performance with Flip Chip Package Designs," *IEEE Symp. IC/Packaging Design Integration*, Santa Cruz, CA, Feb. 1998.
- [31] B. A. Gieseke, et al., "A 600 MHz Superscalar RISC Microprocessor with Out-of-order Execution," *ISSSC 1997 Dig. Technical Papers*, Feb. 1997, pp. 176–177.
- [32] H. H. Chen and S. E. Schuster, "On-chip Decoupling Capacitor Optimization for High-performance VLSI Design," *Proc. Int. Symp. VLSI Technology Systems Applications*, Taipai, Taiwan, ROC, June 1995, pp. 99–103.

PART  
**V**

## **MEMORY SYSTEM DESIGN**

# REGISTER FILES AND CACHES

Ronald P. Preston  
*Compaq Computer Corporation*

As microprocessors have become larger and more complicated, internal data and program storage in the form of register files and cache arrays consume an increasing portion of the transistor count and die area [1]–[5]. (See Table 14.1 for a survey of the on-die arrays for several recent microprocessors.) Consequently performance, power, yield, and reliability of the overall die will be greatly influenced by the design of the register files and caches. Fortunately, the regular nature of memory arrays allows for a more detailed analysis of each of these issues than is typically practical for most circuit structures.

**TABLE 14.1** Survey of Recent Microprocessors

Processor	Cache/register file organization	Reported cache area	Reported cache transistor count	Reference
PA-RISC	1 MB L1 DCache, 0.5 MB L1 ICache	52% of die		[1]
IBM G5	256 KB L1 unified		18 M of 25 M (72%)	[2]
Power-PC	32 KB L1 ICache, 32 KB L1 DCache	17.5 mm <sup>2</sup> of 83.5 mm <sup>2</sup> (21%)	7.4 M of 10.5 M (71%)	[3]
Alpha 21264	64 KB L1 ICache, 64 KB L1 DCache 80 entry 64b IRF 72 entry 64b FRF		9.2 M of 15.2 M (60%)	[4]
UltraSparc	16 KB L1 ICache 16 KB L1 DCache 160 entry 64b RF		1.8 M of 5.2 M (35%)	[5]

Register file and cache array design is simultaneously both extremely simple and very complicated. Arrays are typically composed of only a very few unique and logically quite simple elements such as decoders, the memory cells, and logic to read and write those cells. Although the circuit elements are logically simple, the large number of cells in a large array creates a difficult circuit design problem. Compact layout of the core array cells is required to meet die area constraints. The reduced layout area necessitates the use of very small transistors, which must drive long capacitive bit lines resulting in very small signal output swings. Special sensing techniques are used to amplify and extract the data. The small array transistors have heightened susceptibility to process variations (line width, mask misalignments, etc.).

## 14.1 BASIC ARCHITECTURE

Embedded SRAM arrays may be divided into several categories based on their microarchitectural function. Some of these categories are: (1) general purpose register files (GPRs), (2) single-cycle cache/queue/buffers, and (3) multicycle caches/queues/buffers. The following sections highlight the microarchitectural and circuit design challenges for each of these types of storage arrays.

### 14.1.1 General Purpose Registers (GPRs)

RISC CPUs, utilizing a load-store architecture, typically require a large number of general purpose registers. Performance-boosting innovations such as register renaming or register windowing further expand the number of general purpose registers. The Alpha 21264, for example, contains 80 integer and 72 floating-point registers each of which is 65 bits wide [4]. The integer register file of the UltraSparc CPU is even larger, containing 160 registers of 64 bits each [5].

Design techniques typically associated with large cache arrays such as differential signaling for reading and writing the cells, sense amplifier receivers, and tight cell layout are commonly used in these register file designs. In Fig. 14.1, the floorplan for two of the integer execution units of the Alpha 21264 is shown. In this design, the integer GPR arrays are located inside the datapaths of the integer execution units between the upper and lower functional units. Consequently, the register file layout must occur on the same pitch as the datapath. Because the datapath pitch is wider than a typical SRAM cell, the register can use devices that are larger than typically used in a large SRAM design. Thus some of the design issues typically associated with SRAM design such as variations induced by process offsets and mask misalignments are reduced and can largely be ignored. Larger devices also offer improved immunity to soft errors caused by alpha particles or cosmic rays.

In superscalar CPUs, the register files must typically be capable of generating several data elements every cycle and simultaneously store results for several preceding operations. Therefore, GPRs are often multiported. The integer register file in the Alpha 21264 contains six separate write ports and four read ports [4]. Figure 14.2 is a schematic of the basic cell used in this GPR array along with the circuitry used to read and write the array.

In this figure, the register file array is represented by the schematic in the bottom box. A cross-coupled pair of inverters forms the storage element in each cell. In this design, the read ports are implemented by pairs of two high pulldown stacks of NMOS devices. The write ports are implemented with NMOS pass gates in a singled-ended operation mode. The upper box in Fig. 14.2 contains sense amplifiers that interface the register file bit lines with the rest of the execution unit.

### 14.1.2 Single-Cycle Caches

Cycle times of high-performance microprocessors are significantly shorter than the off-chip memory fetch latency. By bringing part of the memory subsystem onto the CPU itself, the average latency incurred when fetching instructions or data can be significantly reduced. Most current microprocessor designs incorporate Level 1 caches for instructions and data onto the die to reduce the average fetch latency. In addition, some microprocessors such as the Alpha 21164 [6] and PowerPC X704 [7] also have a larger second-level cache on the die.

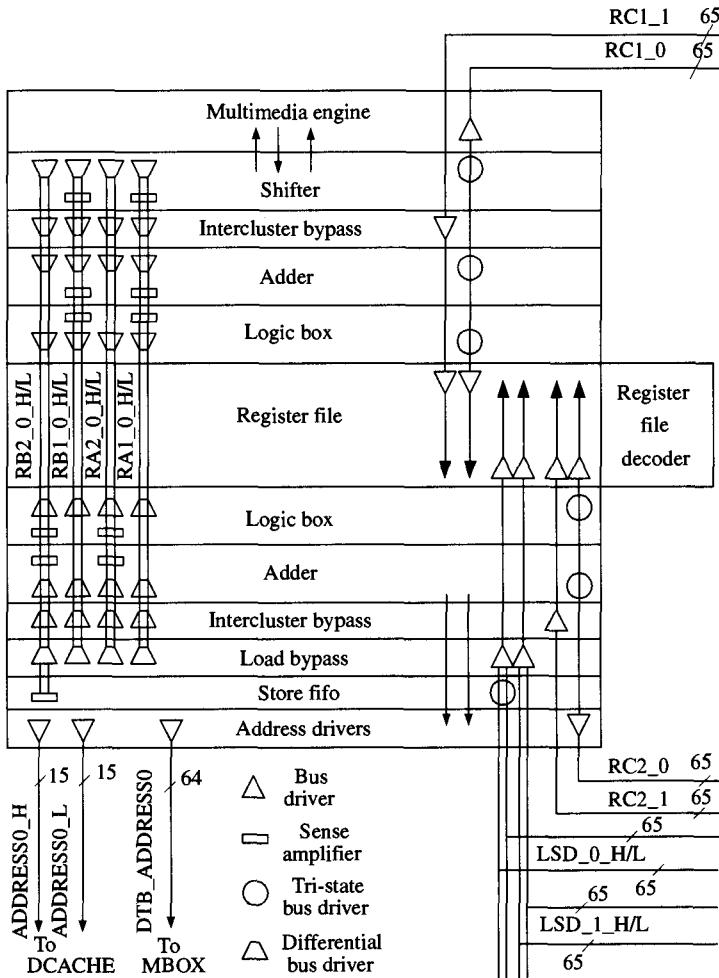


Figure 14.1 Alpha 21264 integer unit floorplan [4].

Typically, the *L1* caches are designed to operate in a single cycle. Single-cycle caches of 16 K to 64 K bytes have been implemented in many current designs [1–6]. The cache array cell is usually the smallest possible SRAM cell that meets basic stability and reliability criteria. The size of the on-chip arrays generally must be limited to somewhere around 64 to 256 rows in height and a few hundred bits in width in order to keep access to a single CPU cycle. To build larger arrays, multiple banks of smaller SRAM arrays are employed. In a multibanked scheme, the outputs from each bank are dynamically driven onto a common bank output bus sometimes called a super or a hierarchical bit line.

Logically the cache arrays are used to hold a copy of a portion of a much larger memory system. Each entry in the cache is tagged with additional bits that indicate which memory address is actually being stored in the particular cache entry and other bits indicating the status of the cached copy. These tag bits may be built into the data array or they may exist in a completely separate tag array. In either case, the bits of the tag array corresponding to the addressed entry are read simultaneously with the data

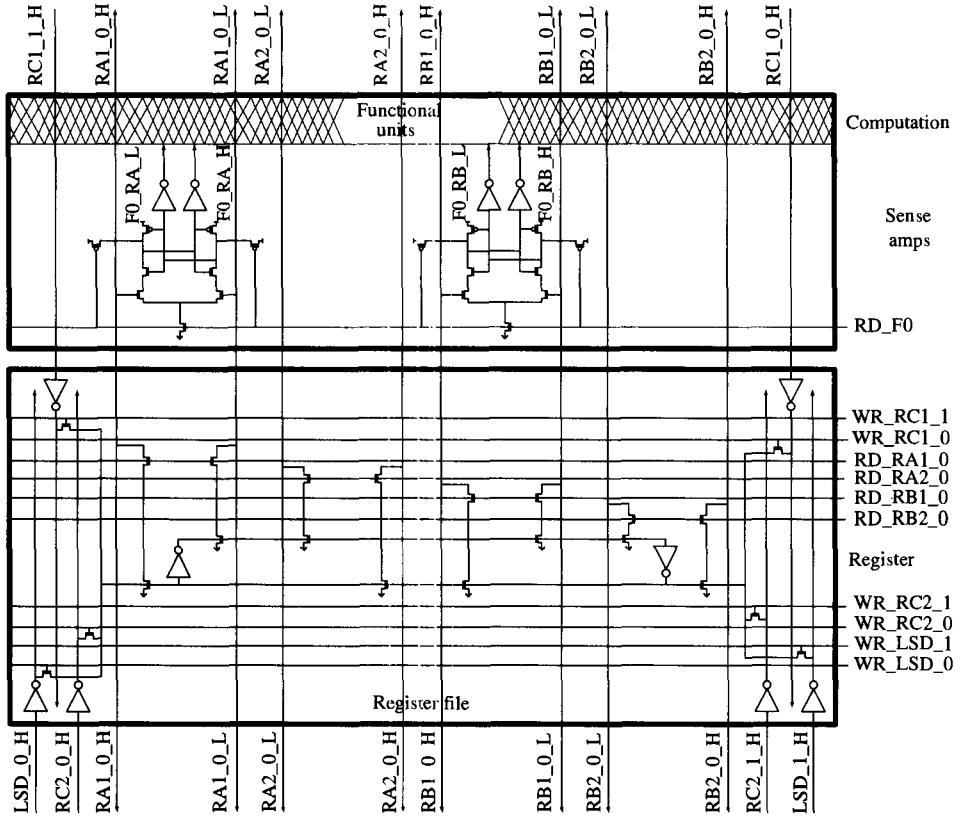


Figure 14.2 Alpha 21264 integer register file cells [4].

bits. The tag is then compared to the desired address and status bits are checked to determine if the cache entry is valid and for the correct address while the data bits are being sent to their destination. This comparison or "hit" logic is often a dynamic XOR circuit built into the tag cache datapath. The need to validate the tag information quickly often makes the read speed of tag bits more critical than the normal data bits.

Occasionally, the microarchitecture will call for multiported cache arrays. These may be implemented as a true multiported array with multiple ports in each cell. However, with the large array sizes and the difficulty in designing stable multiported cells, other solutions are often used. The Alpha 21164 mimicked a dual-ported Data Cache array by creating two identical single-ported array designs [6]. Each array can be independently read to simulate a true dual-ported architecture. Data is written to each array simultaneously.

The Alpha 21264 took a different approach and created a single-ported array that could be accessed twice per CPU cycle [4]. Figure 14.3 illustrates the design of the L1 data cache from the 21264. Two sets of address decoders are used to create the two addresses. One word line driver asserts its selected word line on the rising clock edge and then releases it after the sense amplifiers have fired. Static precharge devices restore the bit lines after the word line has de-asserted. A second word line is asserted on the falling clock edge selecting a second array entry. Demultiplexers are used to route the

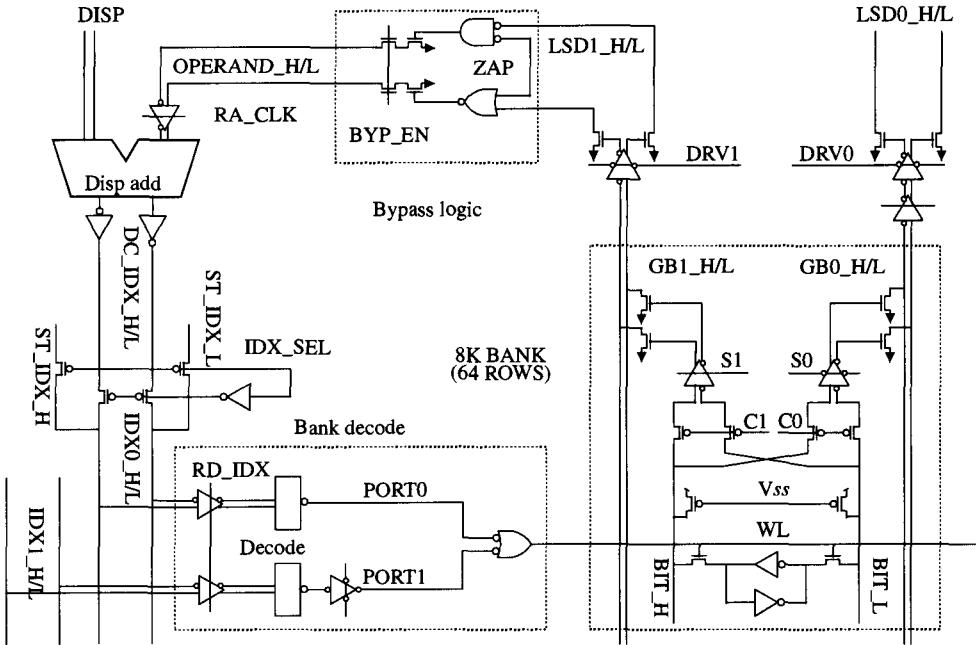


Figure 14.3 Alpha 21264 L1 DCache schematic [4].

data from the cache array onto the appropriate outputs. With this logic, two array accesses can be accomplished in a single CPU cycle.

### 14.1.3 Multicycle Cache Arrays

Cache arrays larger than a few tens of kilobytes are difficult to implement in a single CPU cycle. By pipelining access, the bandwidth of these larger caches can be comparable to that of a single-cycle cache of the same size. The latency for a particular access is obviously greater in this case than for a single-cycle cache but many times the throughput and array size (with the resulting higher “hit” rate) may be more important to overall CPU performance.

Multicycle access also provides the opportunity to save significant power by pre-decoding the access address and selectively enabling only the relevant banks and/or sets of the array. The large *L*2 cache of the 21164 employs this type of selective enabling technique to activate only a single 4 K byte bank of the 96 K byte total array. Figure 14.4 is a block diagram for the organization of the one half of the *L*2 cache on the Alpha 21164 [6]. In this case, it takes three CPU cycles to read data from the array. In the first cycle, the address is predecoded and the tag array is read (not shown). Pre-decoding the address allows the conditional clock generation circuitry to force nine of the 12 array banks into a precharge state. The bit lines and sense amplifiers in the nine disabled banks are frozen to conserve power. In the second cycle, one of the “TAG HIT SELECT” lines asserts if there is a tag match. In the third cycle, the row line and sense amplifiers are enabled for the selected set/bank, thus reading a single one of the 12 banks. The two nonselected banks consume some clock power but since no row line is asserted and the sense amplifiers are not enabled, power consumption is significantly

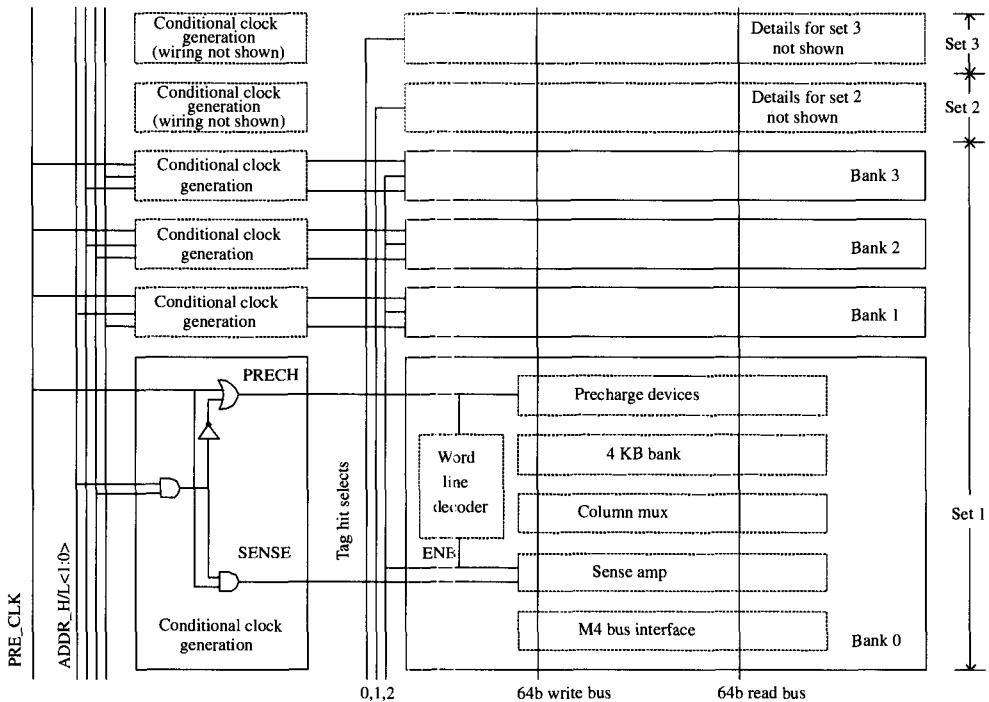


Figure 14.4 Block diagram of the Alpha 21164 L2 cache.

lower than that for the selected bank. In the event of a tag miss, no banks are selected and the entire cache consumes very little power. The multicycle pipelining and power reduction scheme saves 10 W or approximately 20% of the chip's total power consumption.

## 14.2 BASIC SRAM CELL DESIGN AND OPERATION

The core storage element used for most register file and cache designs on high-performance microprocessors is a six-transistor CMOS cell with cross-coupled inverters as storage elements and two pass gates as a combination read/write port. Figure 14.5 illustrates the basic 6T SRAM cell. For comparison a four-transistor, static poly load cell often used in commercial SRAM die is also shown in Fig. 14.5. The four-transistor cell offers more compact layout at the expense of some additional standby current. In order to maintain cell writeability and stability, the pullup resistors need to be very large, on the order of megohms to gigohms. In commercial SRAM processes, a second lightly doped and therefore highly resistive polysilicon layer is used to create the required resistance with a very small layout footprint. Most microprocessor processes offer only a single polysilicon layer that is both heavily doped and quite often silicided with a refractory metal such as cobalt or tungsten. The sheet resistivity for this highly doped and silicided poly is in the order tens of ohms. Creating megohm resistors in a small layout footprint is thus not possible in these processes.

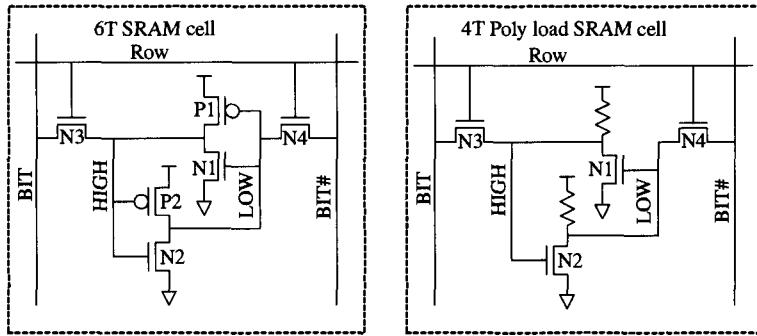


Figure 14.5 6T and 4T SRAM cells.

### 14.2.1 Read Operation

Figure 14.6 illustrates the read operation of the basic 6T SRAM cell. Assume that the cell is currently storing a logic “1” so that node “High” is being held at  $V_{DD}$  by device P1 while node “Low” is being held at  $V_{SS}$  by device N2. A read operation begins with the two bit lines, BIT and BIT#, precharged to a high value, typically  $V_{DD}$ . The address is decoded and the word line, ROW, for the selected row is driven across the array turning on the access devices N3 and N4. The series connection of ON devices N4 and N2 begins to sink current from the bit line node BIT#, thus discharging it toward ground. Since node HIGH is at  $V_{DD}$ , and device P1 is ON, very little current flows through device N3 and node BIT remains in a high state. After a sufficient time to allow a differential voltage to appear on nodes BIT and BIT#, the sense amp circuitry is activated and the value that the cell has placed onto the bit lines is read.

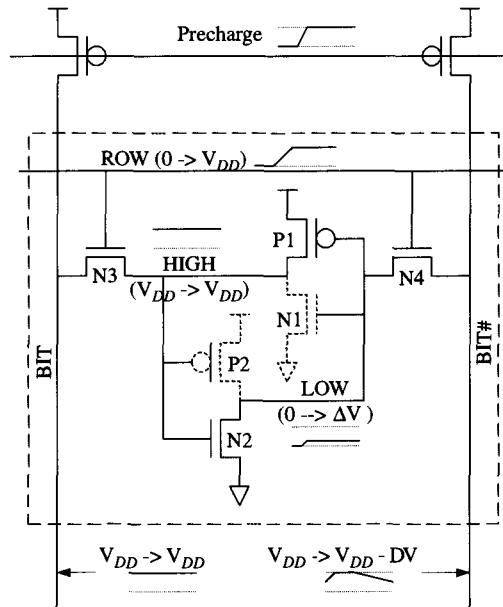


Figure 14.6 Read operation of the 6T SRAM cell.

The read operation described in the previous paragraph provides a hint at some of the design issues involved in SRAM cell design. First, notice that the bit line BIT # is being discharged through the series combination of devices N4 and N2, both of which are typically sized near the process minimum in order to minimize cell area. The charge pulled from the BIT # line and therefore the differential between BIT and BIT # will be limited by the small size of these devices. A major contribution to the read speed of the array will be the time required to build sufficient differential on the bit lines for the sense amplifier to read the correct value. Devices N4 and N2 form a resistive divider between node BIT # and node LOW. Therefore, node LOW will rise somewhat in response to the activation of the row line which can be problematic since it can potentially cause device N1 to turn on, discharging node HIGH. Node HIGH falling would pull down node BIT which should remain high during the read operation. Also, the inverter formed by P2/ N2 could begin to amplify the discharge of node HIGH and flip the cell from storing a logic “1” to a logic “0”, a disastrous condition known as a read-disturb or read-upset. The designer must therefore carefully limit the allowed voltage rise on node LOW to an acceptable value that prevents the read-upset condition from occurring while simultaneously maintaining acceptable circuit speed and area constraints.

The voltage rise on node LOW can be estimated by examining simplified  $I_{ds}$  equations for devices N2 and N4. Device N4 begins the read access with a very large  $V_{DS}$  value (nearly  $V_{DD}$ ). If we assume that the row line rapidly rises to above the threshold voltage while the bit line remains nearly at  $V_{DD}$ , we can assume that this device is in saturation. For device N2,  $V_{DS}$  is nearly 0 so this device is operating in the linear mode. To a first order, we can assume that the current through device N2 equals the current through N4. (Note: In the following analysis, first-order, long-channel device equations are employed to simplify the analysis and demonstrate the basic concepts.)

$$\frac{1}{2} \left( \frac{\mu\epsilon}{T_{ox}} \right) \left( \frac{W_{N4}}{L_{N4}} \right) (V_{ROW} - V_{LOW} - V_T)^2 = \left( \frac{\mu\epsilon}{T_{ox}} \right) \left( \frac{W_{N2}}{L_{N2}} \right) \left( V_{HIGH} - V_T - \frac{1}{2} V_{LOW} \right) V_{LOW}$$

This equation illustrates that the voltage on node LOW is affected by the sizes of both devices N4 and N2. As node LOW rises, the  $V_{GS}$  value for device N4 falls and therefore less charge is removed from the BIT # line, thus slowing down the read access. The rise on node LOW, however, increases the current flow through device N2 helping to keep node LOW from rising too far. In order to create a stable cell that meets array read speed, the designer typically needs to adjust the size for devices N2 and N4. To begin setting some sizing limits, the first step is to define the ratio of the pull down to the pass gate as the cell ratio or CR.

$$CR = \frac{W_{N2}/L_{N2}}{W_{N4}/L_{N4}}$$

If node ROW is driven to  $V_{DD}$  and node HIGH remains at nearly  $V_{DD}$ , then the first equation can be rearranged and simplified.

$$\frac{1}{2} (V_{DD} - V_{LOW} - V_T)^2 = CR \left( V_{DD} - V_T - \frac{1}{2} V_{LOW} \right) V_{LOW}$$

The next equation solves the quadratic relationship for the voltage rise on node LOW with varying cell ratio,  $V_{DD}$ , and  $V_T$ .

$$V_{LOW} = \frac{(V_{DD} - V_T)(1 + CR \pm \sqrt{CR(1 + CR)})}{1 + CR}$$

To avoid the read-disturb problem, the voltage on node LOW should remain below the trip point of the inverter pair P1/N1 for all process, noise, and operating conditions. Figure 14.7 graphs this function for the case where  $V_{dd}$  is 2.5 V and  $V_T$  is 0.5 V.

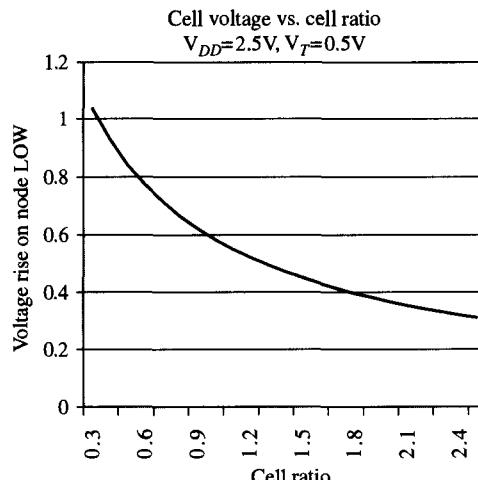


Figure 14.7 Cell ratio vs. voltage rise.

If holding node LOW to less than  $V_T$  is assumed acceptable, it can see that the cell ratio should be kept greater than about 1.28. This limit is typical of most current microprocessor fabrication technologies where a minimum cell ratio of 1.25 to 2.0 is required under normal operating conditions.

Normally the designer will perform detailed SPICE simulations to establish cell stability limits rather than the approximate analytical analysis presented above. To ensure a stable cell design, the minimum cell ratio simulation should be performed assuming the worst-case  $V_{DD}$  and  $V_T$  values, which from the equation for  $V_{LOW}$  indicates, will occur at high  $V_{DD}$  and low  $V_T$ . In addition to varying  $V_{DD}$  and  $V_T$ , cell ratio mismatches caused by process variations and misalignments must be included in the simulation. Finally, the assumed stability point of node LOW rising to  $V_T$  may not be appropriate when all noise sources are included in the analysis.

#### 14.2.1.1 Cell Sizing and Process Variations

Typically, a minimum-sized cell is required when large cache arrays are designed. In order to produce a minimum cell size, a minimum sized pulldown device (N1 and N2 in Fig. 14.6) might be selected. With these minimum pull downs, the pass-gate transistor must have a longer than minimum channel length in order to achieve the proper cell ratio. The long channel device for the pass gate increases the load seen by the row decoder and limits the current discharged from the bit lines both of which can adversely affect the read speed of the overall circuit. Another design possibility is to select a minimum sized pass gate and boost the width of the pulldown devices. This reduces the loading on the word lines and increases the storage capacitance in the cell, both of which are advantageous. The disadvantage of this sizing algorithm is that the cell may be slightly larger than with the long-channel pass-gate approach.

In either case, care must be taken in layout of these minimum-sized transistors to avoid introducing significant transistor size mismatches due to nonperfect processing and mask misalignments. Figure 14.8 illustrates a sample cell layout and some possible processing issues and misalignments that might influence the cell ratio and therefore the cell stability. In the figure, the polysilicon mask has been misaligned with respect to diffusion and significant rounding of diffusion and poly edges has occurred. In addition, the polysilicon on die is slightly narrower than originally drawn which can affect the cell ratio if longer channel devices are used for the pass gates (as shown). Careful cell layout is required to minimize device variations when all possible process and operating ranges are considered. In general, avoiding bends in either poly or diffusion, orienting all gates in the same direction, and setting all devices to an identical nonminimum channel length will reduce the variance. However, significant additional cell area may be required to create a cell that is free of biases and using nonminimum channel lengths will reduce the read speed of the array. The cell designer must balance the increase in array area against the potential biases introduced by using minimum-sized devices and cell layout.

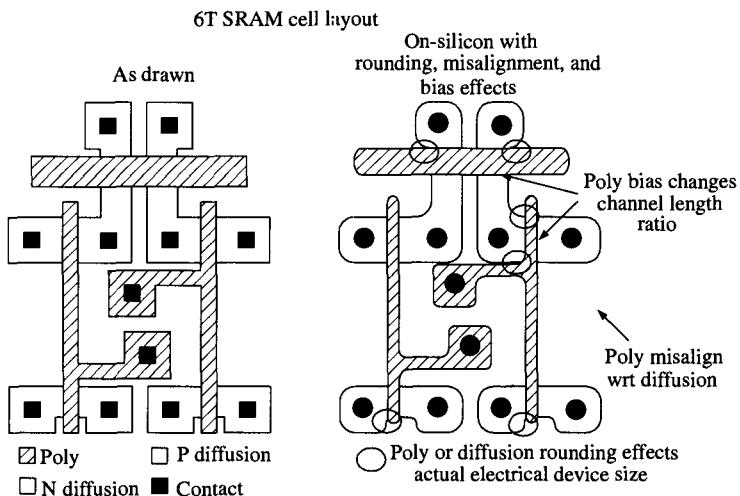
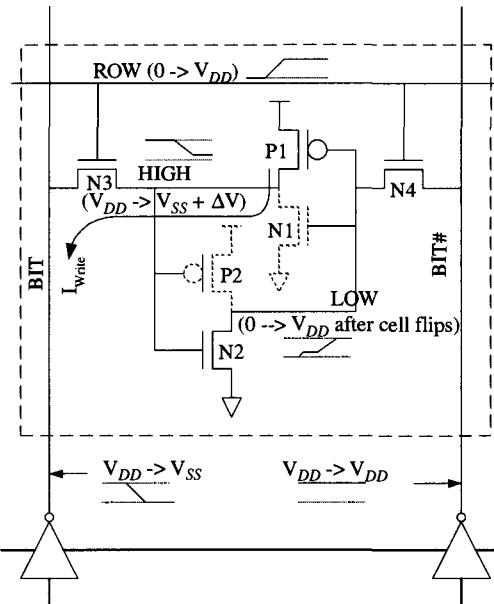


Figure 14.8 SRAM cell layout with processing effects.

### 14.2.2 Write Operation

Writing data to the six transistor SRAM cell begins with external circuitry driving complementary data onto the two bit lines **BIT** and **BIT#**. The address decoders then drive the word line **ROW** high in order to open the two pass gates. Figure 14.9 illustrates the operation where the cell initially stored a “1” (node **HIGH** is at  $V_{DD}$ ) and is written with a “0” (node **HIGH** is pulled to  $V_{SS}$ ).

In the discussion of the read operation, it was noted that a minimum cell ratio is required to prevent accidentally writing the cell. Larger cell ratios while preventing accidental updates also make it somewhat harder to accomplish a desired write operation. Note that the high going bit line (**BIT#**) in this write operation has basically the same effect as the precharged bit lines for the read operation previously described. A voltage division initially occurs between the pass gate and the pulldown in the inverter. The design of the cell ratio to prevent read disturbs holds the internal node to a very



**Figure 14.9** Write operation of the 6T SRAM cell.

low value (typically at or below  $V_{Th}$ ). Therefore, the inverter pair formed by device P1/N1 does not amplify the new write data and the high going side of the cell, at least initially, doesn't contribute to the write operation of the cell.

The low going bit line (BIT), also forms a voltage divider, this time between the PMOS inverter device P1 and the pass gate N3. In order to write the cell, the pass gate must be much more conductive than the PMOS device. This will allow node HIGH to be pulled to a value low enough for the inverter pair (P2/N2) to begin amplifying the new data. As the inverter begins amplifying the low state on node HIGH, the pulldown on node LOW turns off and the PMOS pullup turns on and node LOW rises toward  $V_{DD}$ . This action turns on the inverter pair P1/N1, thus forcing node HIGH further toward  $V_{SS}$ , completing the cell update.

A rough analysis can establish the maximum ratio of the pullup size to that of the pass gate required to guarantee that the cell is writeable. Since the case for a successful write is that node HIGH is pulled to a low value very near the BIT value, we can assume that the pass gate ends up in the linear region. At the end of the write operation, the PMOS device ends up operating in saturation. The next equation illustrates this condition. (Once again long-channel device equations are used for simplicity.)

$$\begin{aligned} & \left( \frac{\mu_n \epsilon}{T_{ox}} \right) \left( \frac{W_{N3}}{L_{N3}} \right) \left[ V_{ROW} - V_{BIT} - V_{Th} - \frac{1}{2} (V_{HIGH} - V_{BIT}) \right] (V_{HIGH} - V_{BIT}) \\ &= \frac{1}{2} \left( \frac{\mu_P \epsilon}{T_{ox}} \right) \left( \frac{W_{P1}}{L_{P1}} \right) (V_{DD} - V_{LOW} - V_{Tp})^2 \end{aligned}$$

To start the simplification of this equation,  $V_{DD}$  is substituted for ROW, 0 is substituted for BIT, and finally  $V_{Th}$  is substituted for LOW. The substitution for  $V_{LOW}$  represents the voltage rise as noted previously for the read case.

$$\mu_N \left( \frac{W_{N3}}{L_{N3}} \right) \left( V_{DD} - V_{Th} - \frac{1}{2} V_{HIGH} \right) V_{HIGH} = \frac{1}{2} \mu_P \left( \frac{W_{P1}}{L_{P1}} \right) (V_{DD} - V_{Th} - V_{Tp})^2$$

As in the read example, the voltage on node HIGH is a function of the sizing of two devices in the cell as well as  $V_{DD}$  and the process parameters. We can define the size ratio of the pullup PMOS device P1 to the pass gate N3 as the pullup ratio:

$$PR = \frac{W_{P1}/L_{P1}}{W_{N3}/L_{N3}}$$

The next step is to solve for the voltage on node HIGH.

$$V_{HIGH} = (V_{DD} - V_{Tn}) \pm \sqrt{(V_{DD} - V_{Tn})^2 - \frac{\mu_P}{\mu_N} (PR)(V_{DD} - V_{Tn} - V_{Tp})^2}$$

Assuming some typical process parameters, we can graph this equation in Fig. 14.10. In order to write the cell, node HIGH must be pulled to a value low enough to trip the inverter combination P2/N2. In this case if we assume that pulling node HIGH below the  $V_{Tn}$  value of 0.5 V is required, the new cell maximum pullup ratio (PR) is limited to a value of no more than about 1.6 V.

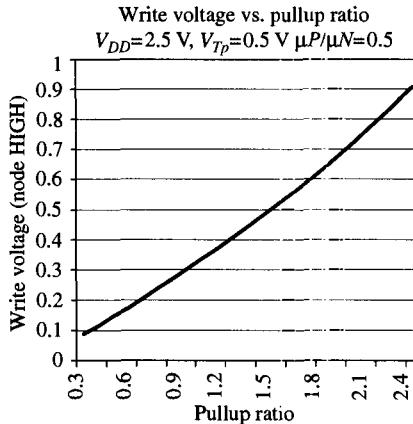


Figure 14.10 Write voltage as a function of pullup ratio.

Substituting nominal process values into the equation for  $V_{HIGH}$ , it appears that sizing for write operations would not be an issue. However, the equation for  $V_{HIGH}$  demonstrates that successfully writing the cell is a function of the transistor sizing,  $V_{DD}$ , the mobility ratio between the P and N devices, and the threshold voltages. The limiting case on the write operation occurs at high  $V_{DD}$  when the P device is strong ( $\mu_p$  high,  $V_{Tp}$  low) and the N device is weak ( $\mu_N$  low,  $V_{Tn}$  high). The cell designer typically needs to simulate the write operation using these worst-case parameters for the mobility ratio, thresholds, and the highest expected  $V_{DD}$ . And as with the read case, the designer also needs to account for possible ratio changes due to mask misalignments. When all effects and worst-case conditions are taken into account, the pullup ratio limit may become significant.

Typically in designing the SRAM cell, the widths of the pullup P devices are sized at or near the process minimum. Longer than minimum channel lengths may also be employed to further reduce the pullup ratio. This is necessary since the read sizing of the SRAM cells dictates that the pass-gate sizing should be minimized to prevent read disturbs.

### 14.2.3 Variants on the Basic 6T Cells

With the many different types of caches and registers files that may exist on a microprocessor, a number of different SRAM cell variants may be employed.

#### 14.2.3.1 Multiple Pass-gate Read/Write Ports

The most obvious solution to adding extra ports to the basic 6T SRAM cell is to add extra pairs of pass gates as shown in Fig. 14.11. Each additional pair of pass gates is connected to separate pairs of bit lines and a separate ROW line. Multiple simultaneous reads are allowed in most designs, so that the cell design must be stable for the case of multiple reads. In the analysis of the read operation, it was shown that a voltage divider occurs between the precharged bit line and the stored “0” in the cell. With more than one pass gate port open, the voltage rise in the cell will be larger and therefore the size of the pulldown in the cell will have to be increased in order to maintain an acceptable low level. To a first order, the minimum cell ratio will increase by a factor equal to the number of simultaneous open read ports. Increasing the cell ratio can be done for a moderate number of simultaneous read ports. But when more than a couple of ports are required, alternate cell configurations are typically used to minimize the penalties associated with this pass-gate port approach.

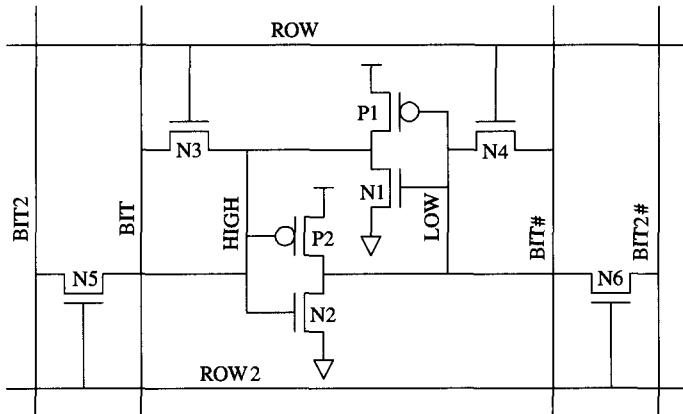


Figure 14.11 Multiported SRAM cell.

#### 14.2.3.2 Read-only Pull-down Ports

One solution to creating multiple-read ports is to employ stacked pull downs as was shown in Fig. 14.2 for the register file from the Alpha 21264. Each read port is operated by a row line (labeled RD\_xxx\_0) asserting to  $V_{DD}$ . One of the two stacks of NMOS devices will begin to pull charge from its associated bit line ( $R_{xx\_0\_L}$  or  $R_{xx\_0\_H}$ ). Since in this configuration, the cross-coupled inverter pair is not directly affected by the read operation, the read disturb problem is not exacerbated with the addition of multiple-read ports. The disadvantages of this configuration are that four devices are required for each dual-ended read port and additional transistors are required to create a write port both of which might increase cell area.

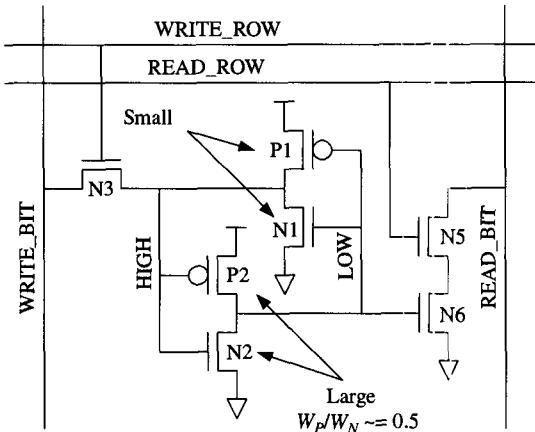


Figure 14.12 Npass write port SRAM cell.

One solution that can help mitigate the area increase is to operate the read ports in a single-ended mode. In this case, only one of the pull-down stacks is required. However, the stack must be sized large enough to create a bit line discharge that can be reliably sensed as a logical “0”. For the dual-ended read port, a differential voltage of 100 or 200 mV is generally enough to be reliably detected by a differential sense amplifier. In the single-ended case, a much larger discharge is required, typically at least twice the differential used in a single-ended mode.

In many cases, the large number of row and bit lines rather than the storage cell layout sets the array cell pitch. In these cases halving the number of bit lines may have a significant effect on overall area size.

#### 14.2.3.3 Single-Ended Write Operation

A single NMOS pass gate can be used for the write port by resizing the cell such that both a “0” and a “1” can be reliably written (see Fig. 14.12). However, the design of this cell is complicated since there is no help in flipping the cell from the second pass gate and the single NMOS pass gate will not pass a full  $V_{DD}$  high value into the cell.

To solve the writeability problem, the cell is sized asymmetrically. The feedback inverter pair (devices P1/N1) are typically reduced in size to a very minimal value, just large enough to counter expected leakage and noise effects on the storage node “HIGH.” This will allow the write port device N3 to easily fight P1 and produce a good low value on node “HIGH.” This low value causes the forward inverter pair (P2/N2) to flip state and finally the feedback inverter (P1/N1) flips state and helps hold the good low value on node “HIGH.”

For the case of writing a high value onto node HIGH, the pass gate N3 will initially be fighting device N1 of the feedback inverter. Since that device was set to a very small size, N3 will initially pull node HIGH upwards. However, as node HIGH approaches  $V_{DD} - V_{Th}$  the pass gate N3 will begin to cut off leaving node HIGH with a voltage near the  $V_{DD} - V_{Th}$  value. The forward inverter pair (P2/N2) is sized with a reduced P/N ratio (0.5 in the example of Fig. 14.12). This will lower the switch point of the inverter to a value closer to the center of the range of values passed by the write port device N3. The inverter will then flip in response to the degraded high value passed into the cell.

## 14.3 ADDRESS PATH DESIGN

The logical function of decoding the address is typically very straightforward. Normally a binary encoded address value is presented to the address decoder circuitry. The decoders may be logically thought of as a set of NAND/NOR gates that accept the true and complement values of each input bit and produce a one-hot output selecting one of N row or column select lines.

### 14.3.1 Preliminary Decoders

In practice, the design of the address decoders is somewhat more complicated. Since the address decoders need to interface with the core array cells, pitch matching with the core storage array can be difficult.

The address decoding problem is often broken into several smaller pieces. The final pitch-matched decoder cell then contains just a two- or three-input NAND structure and the buffering required to drive the final load as shown in Fig. 14.13. In this example, the address bits are predecoded into three sets of one-hot signals. The 18 lines, which constitute the predecoded values, are clocked and routed across the entire array of 128 final decoders. Each final decoder taps into one signal from each of the three sets to perform the final level of decoding. Since the 18 predecoded lines run perpendicular to the array of final drivers, the final cell can be layed out matching the tight storage array pitch.

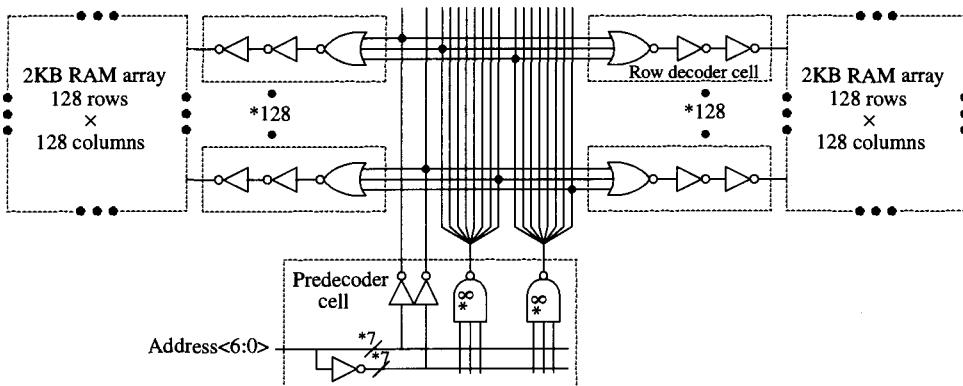
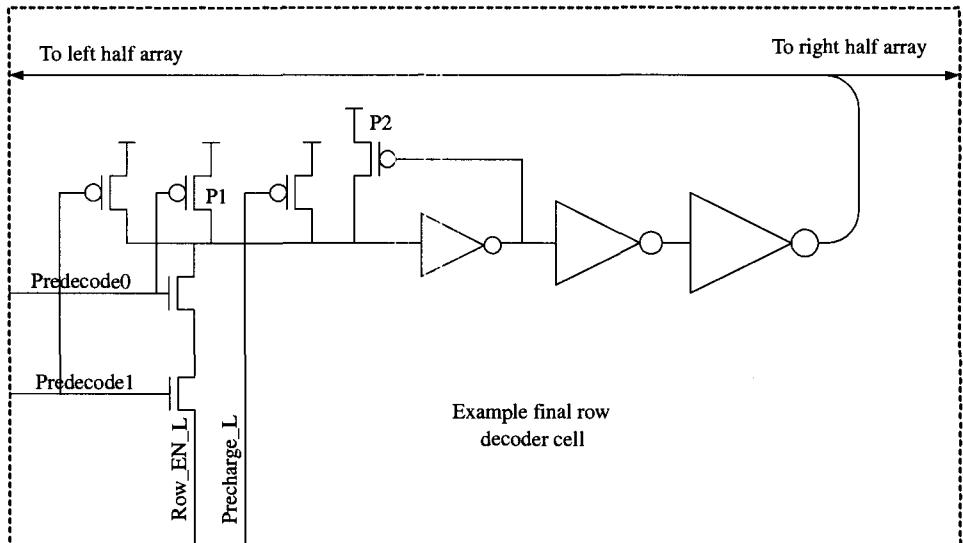


Figure 14.13 Split row decoder scheme.

### 14.3.2 Final Row Drivers

Even with predecoding, fitting the layout for the final row driver into the tight cell pitch can be challenging. Often the final row decoder must drive a very long and heavily loaded word line. Dynamic logic and virtual ground techniques are often used to exploit the one-of-N decoding.

Figure 14.14 illustrates an example row decoder where the signal `ROW_EN_L` is a virtual ground signal for the row decoder array. An array access begins with the `ROW_EN_L` line in a high state and the predecoded addresses asserting on the inputs. The `ROW_EN_L` line then falls causing the selected `WORD_LINE` to assert high.



**Figure 14.14** Example of a final row decoder cell.

Since only a single decoder will activate for each read operation, the current for the devices driving the ROW\_EN\_L signal is manageable. After the array access is complete, the ROW\_EN\_L returns high and de-asserts the enabled WORD\_LINE. In many arrays, self-timed circuitry is employed with the final decoders playing a significant role. The word line assertion period is set such that there is an adequate time for the array access to occur. The word line de-assertion period is also critical since the bit lines must precharge back to their initial state before the next access. Also since the de-assertion of the ROW\_EN\_L signal drives a high value through a stack of NMOS devices, an additional precharge device (P1 in Fig. 14.14) is often used to assist in the de-assertion of the word line. Also note in Fig. 14.14 that a second PMOS device P2 is used to counter leakage effects through the NMOS stacked devices.

### 14.3.3 Word Line Hierarchies and Layout

A single cache read access typically generates a large number of bits of data. In the case of superscalar CPUs, an instruction cache read may produce four or more 32 bit instructions. Data cache reads and writes are often expressed in quad-, octa-, or even hexa-word lengths where a “word” is a number of bytes. The address selection circuitry must therefore select a large number of bits simultaneously. The row line driven by the row decoder is therefore a very heavily loaded line driving many SRAM cells.

The RC delay of a long word line can have a significant impact on the overall read and write speed of the array. The most obvious solution to reduce the RC delay is to reduce the length of the lines. In large SRAM array design, the row decoders and row drivers are often positioned in the center of the array driving both left and right through the core storage cells (as shown in Fig. 14.14). In even larger arrays, the row driver circuitry may be replicated with each copy placed such that the lengths of the row lines are further reduced.

Another solution to mitigate the RC delay of the word line is to use a lower resistance/capacitance metal layer. Most contemporary fabrication technologies feature four or more interconnect layers where the upper metal layers are thick and consequently have lower RC delays. However, an upper interconnect layer cannot directly contact down to the read/write port transistors in the SRAM cell. Special contacting or “strapping” cells must be created and placed regularly through the array to attach the upper metal layer word line to a lower layer that routes to the transistors in the array.

## 14.4 READ PATH DESIGN

In large register file and cache arrays, low swing differential signaling techniques are typically employed in the design of the read datapath. The small size of the transistors in the array cell and the high loading on the bit lines result in a slow discharge of the bit lines discharging only a fraction of  $V_{DD}$  during a typical read cycle. Obviously longer bit lines attaching to larger numbers of cells will be slower than shorter, less heavily loaded lines. Column multiplexing, if required in the design, can further reduce the voltage drop on the bit lines. A high gain differential sense amplifier is therefore used to detect the data value being read. When creating a large array, the designer needs to carefully examine the length and loading of the bit lines to ensure that enough voltage differential can be achieved for reliable sensing.

### 14.4.1 Twisted Bit Line Architectures

In addition to gross loading values, the designer must also pay attention to non-common-mode noise sources that can reduce the apparent differential at the sense amplifiers. One of the largest sources of noncommon-mode noise on bit lines is capacitive coupling to neighboring bit lines. Often array designers will adopt a “twisted” bit line architecture in which the bit lines positions are swapped periodically as they run through the array. This will tend to equalize the coupling effect of a moving bit line in an adjacent column on both of the bit lines in the victim column.

### 14.4.2 Bit Line Precharging

After a read it is crucial to equalize the voltage on the two bit lines so that the next read begins without a built-in bias. In principle, all that is needed is for the bit lines voltages to be equalized and set to a value high enough to avoid a read disturb. (Note: this low-voltage read disturb is the opposite of the high-voltage read disturb described in Sect. 14.2.1.) For simplicity, bit line equalization is typically accomplished by precharging to the processor’s  $V_{DD}$  supply value. This precharging can be accomplished by using either clocked devices or static pullups as shown in Fig. 14.15.

The static pullup scheme has the advantage that it does not require a heavily loaded precharge clock signal to be routed across the array. A disadvantage is that the precharge device is always on, fighting against the bit line discharge for the bit lines that are moving low. Therefore when using static pullups, it is important that they be sized no larger than needed to remove the differential from a previous read operation.

The clocked scheme allows the designer to use much larger precharge devices so that bit line equalization happens more rapidly. Note also that the clocked precharge

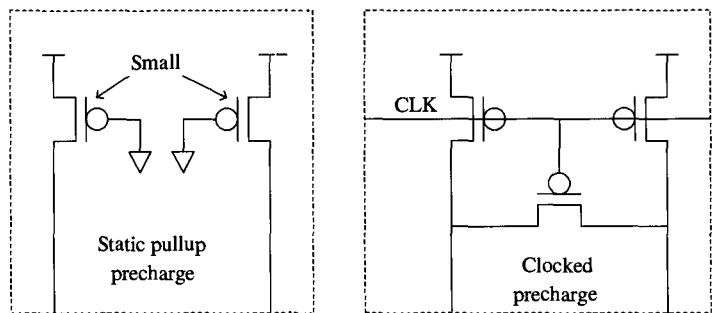


Figure 14.15 Bit line precharge examples.

circuit contains a third device connected between the two bit lines. This device further speeds equalization of the two bit lines by allowing the capacitance and pullup device on the nondischarged bit line to assist in precharging the discharged line. The disadvantages of the clocked scheme include the power consumption of the heavily loaded precharge clock signal. In addition, it is often difficult to create the precise timing needed on the precharge clock.

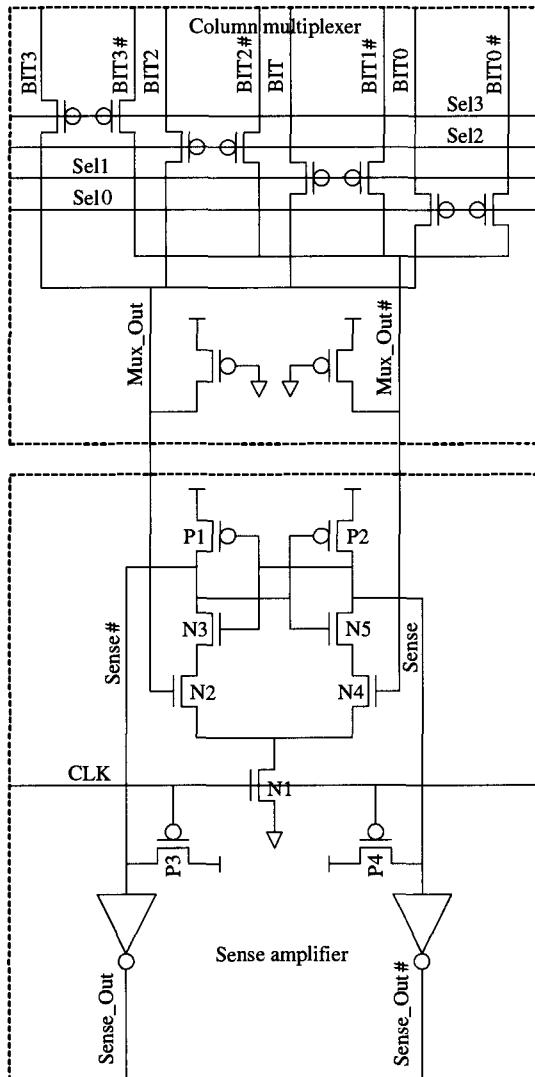
#### 14.4.3 Column Multiplexing

By implementing column multiplexing, a designer can reduce the number of rows needed in the array and therefore reduce the bit line length. Second, laying out sense amplifiers and write amplifiers on the narrow bit pitch of a 6T SRAM cell would be extremely difficult. By using column multiplexing the sense amplifiers and other circuitry can be layed on a pitch that is a multiple of the basic SRAM cell width.

With the tight bit pitches and low signal swings involved, column multiplexers tend to be implemented as simple pass-gate structures. In most cases, a designer would normally implement a pass-gate multiplexer using NMOS devices since device mobility in N-channel devices is typically at least twice that of PMOS devices. However, in this case, the bit lines are precharged to  $V_{DD}$  and discharged to a value only slightly below  $V_{DD}$ . With its source node at nearly  $V_{DD}$ , an NMOS pass-gate device would have a very low  $V_{GS}$  value and thus would conduct very little current. With PMOS devices, it is possible to create high  $V_{GS}$  values by driving the selected gate node to ground. The PMOS device operated in this manner has a higher conductivity and therefore is typically used for the read column multiplexer. Figure 14.16 illustrates a typical column multiplexer attached to a sense amplifier.

#### 14.4.4 Sense Amplifier Design

With the small array device sizes and short cycle times typical on microprocessors, the bit lines may only discharge 100 to 200 mV during a read access. Reliably sensing this small voltage differential is a difficult challenge. Many different circuits have been designed for this purpose including voltage mode differential amplifiers, current mode differential amplifiers, and current mirror amplifiers. In many of these solutions, an accurate bias voltage must be generated to ensure that the final amplifier will be well behaved. On a large microprocessor, creating and guaranteeing an accurate voltage



**Figure 14.16** Example of a read column multiplexer attached to sense amplifier.

reference is difficult given the large amount of switching noise on the power and ground grids.

Figure 14.16 illustrates a simple differential amplifier/latch that has been used on a number of Alpha microprocessors. It consists of a pair of NMOS devices (N2 and N4) which are used to sense the voltages on the bit lines. The gates of these devices are attached to the two bit lines coming from the column multiplexer. A single NMOS pulldown device (N1) is used to enable the sense amplifier by providing current to the two detection devices. A cross-coupled inverter pair (P1/N3 and P2/N5) in a cascode configuration is used to amplify the detected voltage differential. Finally, a set of inverters is used to buffer the output of the sense amplifier.

Operation of this sense amp begins with the CLK signal in a “low” state. The precharge devices (P3 and P4) set the Sense and Sense# nodes to  $V_{DD}$ . The bit lines

driving N2 and N4 are also precharged to  $V_{DD}$ . As the read access begins, one of the two input lines (Mux\_Out or Mux\_Out #) will begin to be discharged slightly by the selected SRAM cell. The sense amplifier is held in the precharge state while a small amount of differential is achieved on the two bit lines. Opening the sense amplifier prior to building sufficient differential exposes any noncommon-mode noise, device mismatches, and bit line precharge mismatches to the sense amplifier and could possibly lead to an incorrect result. After a time period chosen to allow sufficient differential to appear, the CLK line is asserted. This causes the pulldown device N1 to turn on supplying current to the two legs of the sense amp.

Referring to Fig. 14.16, assume that the Mux\_Out line remains high while the Mux\_Out # line falls 200 mV prior to the CLK enabling the sense amp. When N1 turns on, nodes Sense and Sense# begin to fall toward  $V_{SS}$ . Since the signal Mux\_Out is at a slightly higher voltage than node Mux\_Out #, the left leg of the sense amp will have a slightly lower impedance than the right leg. This difference causes node Sense# to fall slightly faster than node Sense. The falling Sense# node will begin to turn on the PMOS device P2 and begin to turn off the NMOS device N5. These devices retard the fall on node Sense, increasing the differential between nodes Sense and Sense#. Eventually P2 is “on” strong enough (and N5 is nearly off) to begin to pull node Sense back up toward  $V_{DD}$ , causing the sense amplifier to latch the correct value. Note that during the sensing operation bit line differential continues to build (further shutting off device N4 and therefore the right leg of the sense amplifier) and the left leg of the sense amplifier will continue to discharge node Sense# until it either reaches ground or the CLK line is removed.

In the design of the sense amplifier, many factors must be considered. Matching layout between the two legs of the sense amplifier is obviously required to avoid introducing offsets that can affect sensitivity. As was the case with the SRAM array cell, avoiding bends in the layout, orienting all gates in a similar manner, and setting all devices to identical nonminimum sizes will help in minimizing variations. In addition, device sizing is crucial to achieving a functional part. For example, device N1 not only switches the device on and off, but also controls both the sensitivity and latency of the structure. Increasing the size of device N1 provides more current to the two legs, which should decrease the latency of the latching operation. However, care must be taken since oversizing N1 can reduce the sensitivity of the sense amplifier since nodes Sense and Sense# will both rapidly fall toward  $V_{SS}$  shutting off devices N3 and N5 prior to a significant differential developing on the two output nodes. One solution that can be employed is to replace device N1 with two devices. The first device is turned on to begin the sensing operation. This device is sized small in order to begin to build differential between nodes Sense and Sense#. A second, larger device is switched on slightly later than the first device which helps rapidly discharge the low falling leg to a value that latches the structure.

#### 14.4.5 Hierarchical Read Paths

In designing large, high-speed arrays, it has already been noted that the designer faces difficult design issues in arranging the word and bit lines such that read access cycle time targets are achieved. Often it is impossible to implement the entire array in a single bank and still maintain the desired cycle time. In these cases, a multiple bank architecture may be employed. For the read path, the outputs from each sense amplifier

are dynamically driven onto another set of bit lines (in a “wired-or” manner) using one or more of the address bits as enabling signals. This second set of bit lines is known as a super bit line or hierarchical bit line.

Depending on the number of banks and the length of the hierarchical bit lines, they may be routed in upper level metals in order to take advantage of reduced resistance and capacitance. To minimize the cycle time impact of the extra logic, these bit lines often use low swing differential signaling techniques similar to those used by the “normal” bit lines. Sense amplifiers may be employed to receive the super bit lines and extract a logic value before a full swing on the lines has occurred.

## 14.5 WRITE PATH DESIGN

The write datapath circuitry’s function is to take new data to be written into the array and present it to the selected cells in such a manner that the cell updates to the new value.

### 14.5.1 Write Amplifier Design

Write amplifiers may be as simple as an inverter that can drive the appropriate bit line to a low value near  $V_{SS}$  quickly enough to achieve the write operation within the allocated cycle time. One complication is that following a write operation, the low asserted bit line must be restored to  $V_{DD}$  prior to the next read operation. When clocked precharge devices are used in the array, this restoration may occur automatically during the next precharge phase. When static precharge devices are employed, the designer must carefully balance the size of the precharge device against the size of the array pulldowns and pass gates in order to have successful read operations. Therefore static precharge devices may not fully restore the bit lines to the  $V_{DD}$  value prior to the beginning of the next read cycle. In these cases, the Write Amplifier circuitry may be required to drive the selected bit line back to a “high” value.

Figure 14.17 illustrates an example Write Amplifier that drives the selected bit line low and then restores it to a precharge state. The extra circuitry in the Write Amplifier box generates a negative pulse on the Pullup\_En line after the WR\_ENABLE signal is removed on the low going bit line. This Pullup\_En signal is routed to an extra set of bit line pullup devices. These devices restore the bit line to a good  $V_{DD}$  level prior to the next read access. This type of timing circuitry requires detailed analysis to ensure that the low going write data pulse is of sufficient length to update the cell and that the Pullup\_En pulse is long enough to restore the bit lines to proper values.

### 14.5.2 Column Multiplexing

In the discussion of the read datapath circuitry, it was concluded column multiplexing is often required in designing large arrays. It was also noted that PMOS devices are normally used for read path column multiplexers since higher  $V_{GS}$  values are possible and thus the PMOS devices are much more efficient at passing the read bit line values. Since writing the array is mostly a function of pulling a bit line to a low value near ground, PMOS devices tend to be ineffective just as NMOS devices were

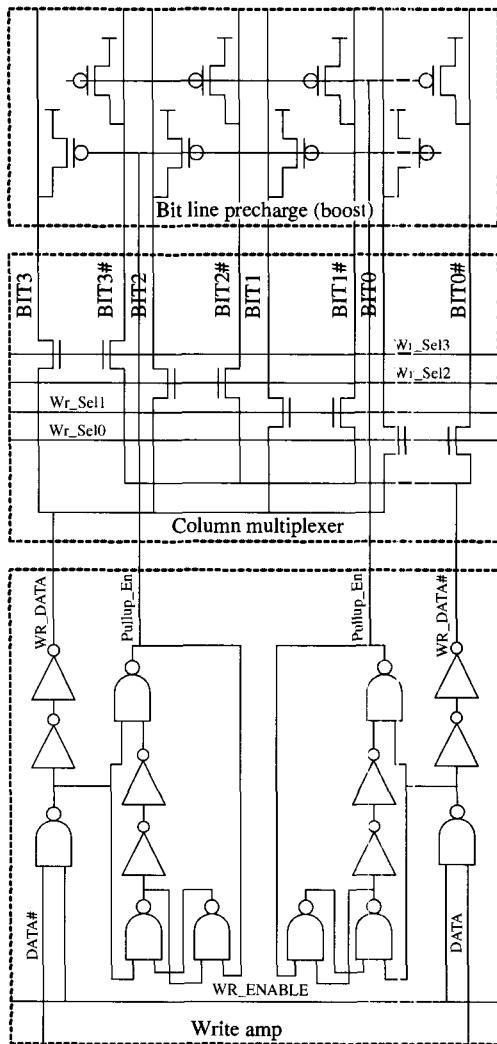


Figure 14.17 Example Write Amplifier, write column multiplexer, and precharge boost.

ineffective in the read case. Therefore, NMOS pass gates tend to be favored for column multiplexing of write values as shown in Fig. 14.17.

## 14.6 REDUNDANCY

The large area consumed by cache arrays and the extensive use of minimum geometries can adversely affect overall die yield. Fortunately, the regular nature of caches is ideal for the addition of redundant elements. These replacement elements can be mapped in to replace defective elements greatly improving the yield of the cache and the overall yield of the entire chip. Cache redundancy may take the form of adding extra redundant rows or columns to the storage array along with the necessary decoders and multiplexers. It also may take the form of adding entire redundant banks including all of the associated logic such as sense amplifiers and timing circuitry.

The decision as to what type of redundancy (row, column, bank, or some combination) and the desired number of redundant elements is made by considering the likely failure mechanism in the actual layout. This analysis may involve running layout checks or critical area analysis that looks for known cases that may be difficult to manufacture such as long lines running at minimum spacing or interconnects made with a single via. The analysis may also use Monte Carlo techniques where simulations of possible defects on the wafer and their impact are studied. The designer must also be aware of the area and possible circuit timing impacts associated with the use of redundant elements in order to balance the improved yield against circuit performance and die area costs.

### 14.6.1 Implementing Redundancy

In order to introduce the redundant elements to the design, there must first be a mechanism for detecting defective elements. Then a programming step must occur that disables the defective elements and maps the extra redundant elements into the array at the address of the defective elements.

The detection mechanism may be a self-test engine that checks the array at power-on or reset. Following testing the array, the self-repair engine can program registers built into the row decoders, column decoders, and bank select logic to disable the failing elements. The engine then loads additional registers enabling the redundant elements and programming them to the proper address.

A production test may also be used to detect the failing elements and determine an appropriate repair scheme. The main disadvantage of this scheme is that the repair mapping must somehow be permanently fixed onto the die. This is typically accomplished using either fuses or some type of programmable memory (PROM) circuit on the die. Die manufacturing cost may be somewhat higher due to the fuse or PROM programming step and retest of the array to verify that the programming and replacement of defective elements was successful.

## 14.7 RELIABILITY ISSUES

As was true for many of the basic design issues, overall chip reliability may be greatly affected by the reliability of large caches. Fortunately the regular nature of the design and the sparse switching activity can be exploited to both simplify the analysis and to grant waivers for circuits that might normally be considered reliability risks in other less regular designs. The designer of a cache must contend with the normal menu of potential reliability issues including electromigration of metal lines, device performance degradation caused by hot carriers, and time-dependent dielectric breakdown (TBD). In addition, caches may be susceptible to dynamic disruptions known as soft error upsets (or SEUs).

### 14.7.1 Alpha Particles and Cosmic Rays

The most common soft error concerns arise from alpha particles and cosmic rays. In both cases, energetic external radiation hits the silicon substrate and creates free electron-hole pairs. These electron-hole pairs represent mobile charge that can migrate to the small storage nodes in the array and degrade the values stored in the SRAM cells [8]. In SRAM cells the presence of cross-coupled inverter pairs provides a mechanism to help fight this degradation. However, the small size of the array cell devices may lead to

very small capacitance values that are easily upset by the generated electron-hole pairs. Also the PMOS pullups that are sized primarily to counter expected device leakage may not produce enough current to quickly counter the charge caused by the radiation.

Determining an SRAM array's susceptibility to SEUs is a complicated process—QCRIT, the critical charge that can cause a storage cell to be disrupted and the number of susceptible array cells must be determined. The smaller the value of QCRIT and the larger the number of cells, the more likely that an SEU will occur. The expected quantity and energy distribution of alpha particles and cosmic rays must be estimated along with the expected electron and hole production efficiency and their lifetime in the silicon substrate. The efficiency of collection, that is, how likely a storage cell is to attract the generated holes—or electrons—must also be estimated. Once all of these factors have been determined, an estimate of the susceptibility to SEUs can be determined.

Many design modifications to help mitigate the problem are possible and readily available. From a circuit standpoint, increasing QCRIT and reducing the collection efficiency can be done by modifying the layout of the array cell. Increasing transistor sizes will typically increase QCRIT. Minimizing source/drain areas of the cross-coupled inverters in the array cells may decrease the collection efficiency. Die packaging modifications such as using low-alpha emitting materials can significantly reduce the number and intensity of alpha particles reaching the silicon substrate. Special absorbing materials may be coated on the internal package surfaces or even on the die itself to further reduce the alpha particle impact. Use of silicon-on-insulator (SOI) technologies can greatly enhance soft error reliability since the insulating substrate in SOI will not generate free electron-hole pairs that can migrate to active storage cells in the array. Architectural solutions to help minimize SEU effects on the overall design may also be employed. The addition of parity or error-correction bits to an array can allow for recovery in the event that a SEU occurs.

## REFERENCES

- [1] J. Lachmann, et al., "A 500 MHz 1.5Mbyte Cache with On-chip CPU," *Paper 11.2 ISSCC Digest of Technical Papers*, pp. 192–193, Feb. 1999.
- [2] G. Northrop, et al., "600 MHz G5 S/390 Microprocessor," *Paper 5.2 ISSCC Digest of Technical Papers*, pp. 88–89, Feb. 1999.
- [3] J. Alvarez, et al., "450 MHz PowerPC Microprocessor with Enhanced Instruction Set and Copper Interconnect," *Paper 5.6 ISSCC Digest of Technical Papers*, pp. 96–97, Feb. 1999.
- [4] B. Gieseke, et al., "A 600 MHz Superscalar RISC Microprocessor with Out-of-Order Execution," *Paper 10.7 ISSCC Digest of Technical Papers*, pp. 176–177, Feb. 1997.
- [5] L. A. Lev, et al., "A 64-b Microprocessor with Multimedia Support," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1227–1238, Nov. 1995.
- [6] B. J. Benschneider, et al., "A 300 MHz 64-b Quad-Issue CMOS RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1203–1214, Nov. 1995.
- [7] E. T. Cohen, et al., "A 533 MHz BiCMOS Superscalar Microprocessor," *Paper 10.1 ISSCC Digest of Technical Papers*, pp. 164–165, Feb. 1997.
- [8] T. C. May and M. H. Woods, "Alpha-Particle-Induced Soft Errors in Dynamic Memories," *IEEE Transactions on Electron Devices*, vol. ED-26, no. 1, Jan. 1979.

Tadaaki Yamauchi and Michihiro Yamada  
*Mitsubishi Electric Corporation*

## 15.1 INTRODUCTION

Embedded DRAM, which merges dynamic random access memory and logic, is an important technology for future VLSI systems. It has the following advantages. It has large memory bandwidth between the DRAM and the microprocessor core or other application-specific logic. The power consumption of I/Os has become large in conventional systems due to the increase in their number and the operating frequency. The integration of the DRAM can reduce the number of I/Os and lower the power dissipation because of the relatively smaller load capacitance of the internal I/O bus compared to the external I/O bus. There is also a reduction of EMI (electromagnetic interference). And highly compact systems can be achieved due to the reduction in overall chip count. Graphic controllers and microcontrollers with the embedded DRAM technology have already been built and deployed in mobile personal computers and digital cameras.

We first explain the DRAM architecture, memory cell, and basic memory core operation of the commodity DRAM as an aid to understanding the embedded DRAM. Next, the internal voltage generators are discussed in detail. These circuits are specific to memory devices and complicate the circuit design of the combined DRAM and VLSI logic. Next the features of the embedded DRAM technology in the area of circuit design, memory architecture, and process issues are presented. The chapter concludes with some recent applications and a brief future perspective.

## 15.2 DRAM BASIS

This section describes the organization of a DRAM, how the memory cell functions, and how the core operates. It also covers the architecture of embedded DRAMs.

### 15.2.1 DRAM Architecture

Figure 15.1 illustrates the basic DRAM architecture, which consists of the following elements.

1. **Memory cell.** Memory cells are arrayed in a matrix manner of  $n \times m$ , forming the memory cell array of  $N (= n \times m)$  bits, where  $n$  and  $m$  are the number of row and column, respectively. Data for  $N$  bits are stored in the memory cell array.

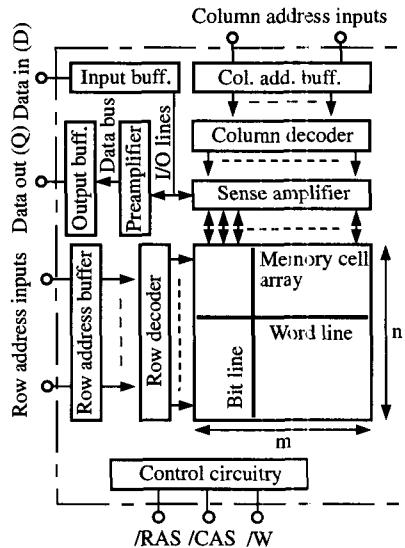


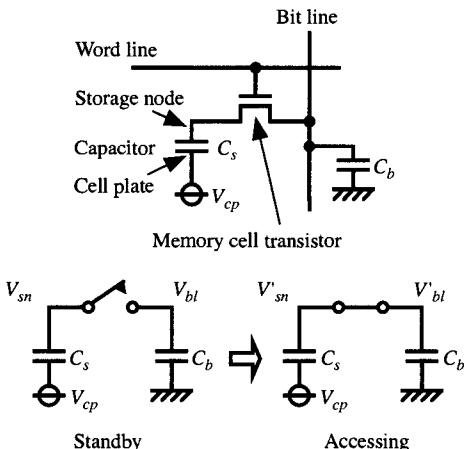
Figure 15.1 The basic DRAM architecture.

2. **Row decoder and row address buffer.** The row decoder selects one among  $n$  word lines by decoding the row address inputs stored in the row address buffer. Usually, the row address inputs are shared with the column address inputs in a time-multiplexed manner at the pins.
3. **Sense amplifier.** By selecting one word line, the memory cells of  $m$  bits are connected to the  $m$  sense amplifiers through the bit lines ( $m$  bit line pairs). The sense amplifiers detect the small signals ( $\Delta V$ ) from the memory cells and amplify them to full swing signals ( $V_{ces}$ ).  $V_{ces}$  is the internal power supply voltage for the sense amplifiers. This sensing operation will be described in the next section.
4. **Column decoder and column address buffer.** The column decoder selects one column select line (CSL) among  $m$  select lines by decoding the column address inputs stored in the column address buffer. By selecting one column select line, one of  $m$  sense amplifiers is connected to the I/O lines and amplified by the preamplifier.
5. **I/O circuitry includes the preamplifier, the output buffer, and the input buffer.**
6. **Control circuitry.** The three main control signals, namely, the row address strobe (/RAS), the column address strobe (/CAS), and the read/write control (/W), control the chip.

Note that three distinct actions occur when data at a specific location are accessed in the memory cell array. First, data are transferred by the bit lines from the memory cells ( $m$  bits) to the corresponding sense amplifiers. Then after the sensing operation, the data are transferred by the I/O lines from the sense amplifiers to the preamplifier and subsequently to the output buffer.

### 15.2.2 Memory Cell

Figure 15.2 shows the memory cell used in DRAM arrays. It consists of an MOS transistor (usually  $n$ -channel) and a capacitor. This structure, named the 1Tr/1C



**Figure 15.2** The schematic diagram of the 1Tr/1C memory cell.

memory cell, has been employed since the advent of the 4 Kbit DRAM. The memory cell array has the following three capabilities:

1. A means for storing data (storage elements).
2. A means for accessing the stored data (word lines).
3. A means for transferring data from or to the storage elements (bit lines).

The 1Tr/1C memory cell array realizes these three elements in the simplest way. The gate and the drain of the memory cell transistor are connected to the word line and the bit line, respectively.

One electrode of the capacitor is the storage node and is connected to the source of the memory cell transistor. The other electrode of the capacitor is the cell plate and is biased at the cell plate voltage level ( $V_{cp}$ ). Binary data, “0” and “1,” correspond to the amounts of charge stored on the capacitor.

The features of the 1Tr/1C memory cell are summarized as follows:

1. Small memory cell size is suitable to high-density DRAMs.
2. The memory cell itself does not have the ability to amplify the stored data on the capacitor. Therefore, the sense amplifier with high-voltage sensitivity is required to detect a small voltage swing on the bit lines.
3. The read operation from the memory cell is destructive. Therefore, a restore operation to replenish the stored charge in the memory cell must be executed by the sense amplifiers.
4. A refresh operation is needed periodically to restore the charge to the capacitor because the leakage current of the storage cell reduces the amount of the stored charge. The refresh operation is vitally important for the correct operation of DRAMs.

The small signal value delivered by the memory cell to the bit line is calculated by applying the charge-sharing principle. When the word line is disabled in the storing state, the voltage on the storage node of the memory cell ( $V_{sn}$ ) is charged to either  $V_{ccs}$  (binary data, “1”) or 0 V (binary data “0”), and the voltage on the bit line ( $V_{bl}$ ) is precharged to half  $V_{ccs}$ . Total charge stored in both the memory cell capacitor ( $C_s$ ) and

the stray capacitor of the bit line ( $C_b$ ) is given by the following equations:

$$Q = C_s(V_{sn} - V_{cp}) + C_b V_{bl}$$

$$V_{bl} = V_{ccs}/2, V_{sn} = V_{ccs} \quad \text{for "1"} \quad \text{and} \quad V_{sn} = 0 \text{ V} \quad \text{for "0"}$$

When the word line is activated, the charge of the memory cell capacitor moves to the bit line, and the voltage of the storage node ( $V'_{sn}$ ) becomes equal to that of the bit line ( $V'_{bl}$ ). The total charge is not changed, resulting in the following equation:

$$Q = C_s(V'_{sn} - V_{cp}) + C_b V'_{bl}$$

Therefore, the signal values from the memory cell ( $\Delta V$ ) are

$$\Delta V(\text{for "1"}) =: (V_{ccs}/2)/(1 + C_b/C_s)$$

$$\Delta V(\text{for "0"}) =: -(V_{ccs}/2)/(1 + C_b/C_s)$$

### 15.2.3 Basic DRAM Core Operation

Figure 15.3 shows the DRAM core circuits of the folded bit line structure including the memory cell array, the sense amplifier, I/O gate, bit line isolation transistors, and the equalizer. The sense amplifier is shared between the adjacent memory cell arrays to reduce the die area. The bit line equalizer and the I/O gates which connect the sense amplifier to the I/O lines are also shared. There are two kinds of bit line structures, folded bit lines and open ones. The open bit line structure has the advantage of smaller minimum memory cell size,  $4F^2$ , compared to  $8F^2$  of the folded bit line structure. The feature size,  $F$ , is the width and spacing of the bit line or word line. However, the folded bit line structure has been used for high density DRAM designs since the 256 Kb generation, because it has an obvious advantage in terms of noise suppression. The word line to bit line coupling degrades the S/N ratio of the open bit line structure as follows. The bit line voltage transition during a sensing operation gets coupled to the unselected word lines that are in precharged mode (usually 0 V) representing high impedance nodes. This noise gets coupled back to other bit lines. Suppose that the majority of the bit lines are reading "1", causing a positive transition on the unselected word lines, which in turn pulls up the bit lines. This is especially harmful for the minority of bit lines that are reading "0".

Next the core operation is described as follows:

1. During the standby state, the bit line isolation signals,  $BLIa$  and  $BLIb$ , and the bit line equalization signal,  $BLEQ$ , are kept at a high level in order to precharge both of the adjacent bit lines. The boosted voltage level,  $V_{pp}$ , is applied to the gates of the bit line isolation NMOS transistors in order to transfer the high level,  $V_{ccs}$ , to the selected bit line without any voltage drop during the sensing operation.
2. The memory core operation is initiated by disabling the bit line equalization and isolating the unselected bit lines. Figure 15.3 shows the case where the right-hand side of the memory cell array is selected by switching  $BLIa$  low.
3. One of the row address inputs is used to select either the right-hand side or the left-hand side of the memory cell array, and the rest of row address inputs decode a single word line (i.e.,  $WL1$  in Fig. 15.3). By activating the word line, a small signal,  $\Delta V$ , is presented to the bit lines.  $\Delta V$  is the difference in the voltage between an activated bit line and an adjacent bit line that is not activated.

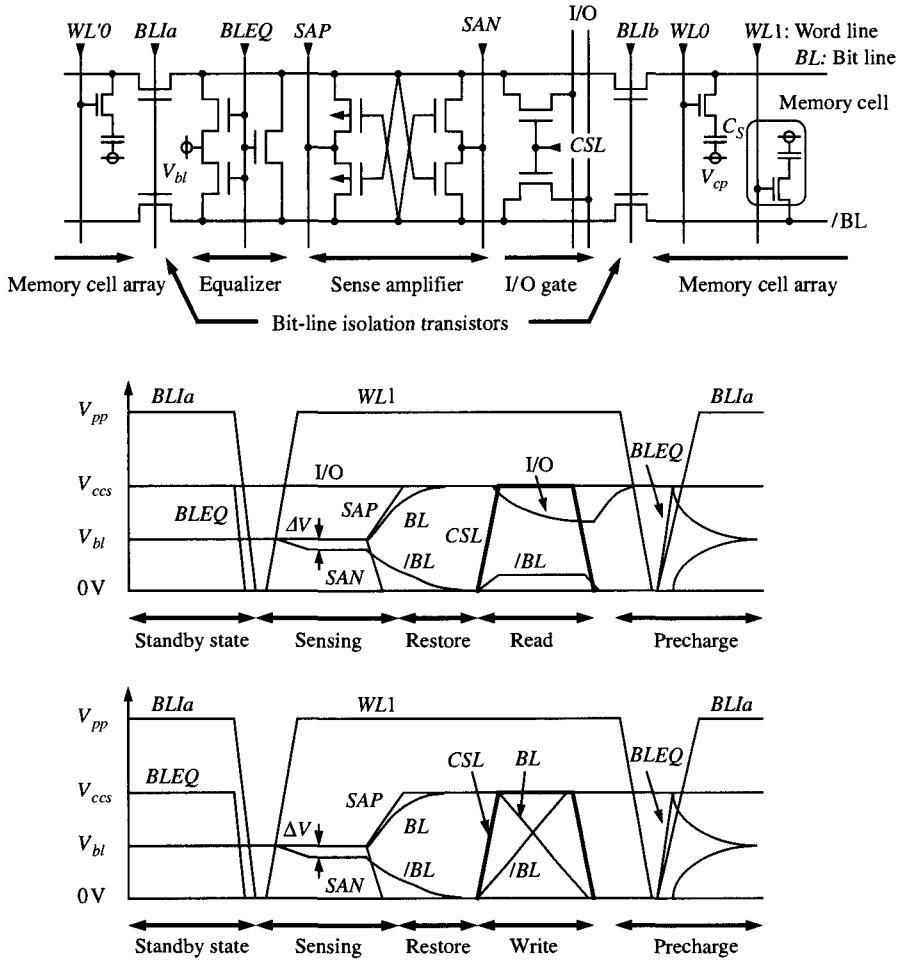


Figure 15.3 The basic DRAM core circuit and operation.

4. Amplification of  $\Delta V$  to the full swing signal ( $V_{ccs}$ ) is performed by the CMOS differential sense amplifier shown in Fig. 15.3. The common source nodes,  $SAP$  and  $SAN$ , of the sense amplifier are precharged to half  $V_{ccs}$  initially, and are driven toward  $V_{ccs}$  and 0 V, respectively, to enable the sensing operation. After the sensing operation is completed, the original stored value ( $V_{ccs}$  or 0 V) is restored back into the memory cell (restore operation). The activated word line is driven to  $V_{pp}$  so that the high data level ( $V_{ccs}$ ) can be written to the memory cell without the voltage drop caused by the threshold voltage of the memory cell transistor ( $V_{th}$ ).  $V_{pp}$  is adjusted to be higher than  $V_{ccs} + V_{th}$ .
5. Next, the column address is loading and decoded to select the column select line,  $CSL$ . A pulse control signal applied to  $CSL$  enables the I/O gate to connect the I/O lines to the selected sense amplifier and eventually to the selected bit lines.
6. In the read operation, the latched data in the sense amplifier are transferred to the preamplifier through the I/O lines (Fig. 15.3, middle).

7. In the write operation, the full voltage on the I/O lines, generated by the input buffer, overrides the data previously latched in the sense amplifier and the bit lines are driven to full swing level (Fig. 15.3, bottom).
8. The precharge operation starts when the word line is turned off. The succeeding operations (disabling the sense amplifier and equalizing the bit lines) must not begin until after the selected word line is discharged to 0 V. Otherwise, the stored data in the memory cell could be destroyed. The bit line precharge level is set to be half  $V_{cc}$  to minimize the power consumption in the memory cell array.
9. After the bit lines are equalized and precharged, the equalizer is left active to hold the bit lines until another access is initiated (standby state).

Note that the sensing, restore, and precharge operations for the selected memory cells are done sequentially in each access cycle. The timing of these operations is determined by the characteristics of the memory cell array, such as word line and bit line delay, which dictate the random access cycle time of the DRAM array.

#### 15.2.4 Memory Cell Array Architecture of Embedded DRAM

The memory cell array architecture of embedded DRAM uses many more banks than the commodity DRAM in order to supply data to the wide internal I/O bus. An example is shown in Fig. 15.4, where a wide internal I/O bus can be achieved. The internal I/O bus routed in the second metal layer is placed over the memory cell array. The column select lines are routed in the first metal above the sense amplifiers. The third

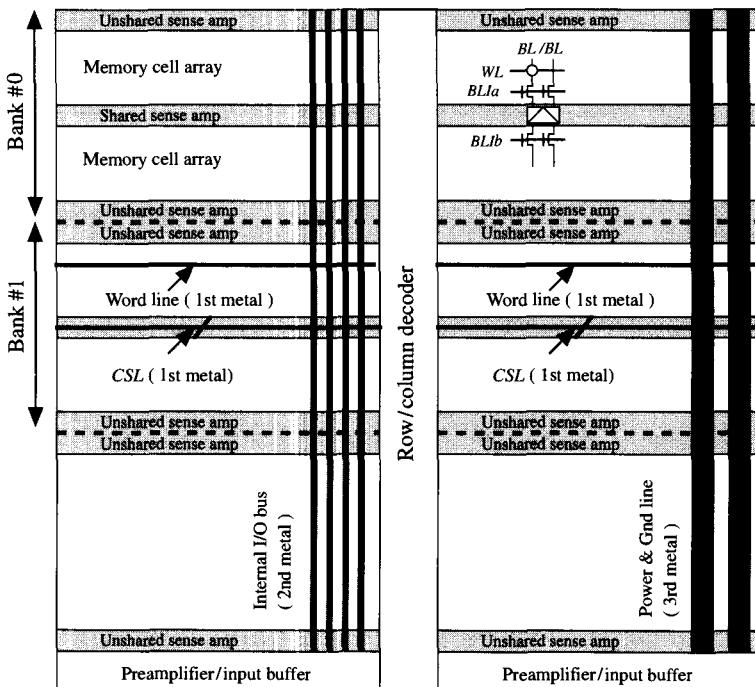


Figure 15.4 Memory cell array architecture of the embedded DRAM.

metal layer is fully utilized by the power and ground lines that supply the sense amplifiers and peripheral circuitry. This reduces power and ground noise when the wide internal I/O bus is operated at high speed and multiple banks are simultaneously activated.

A critical problem inherent in embedded DRAM lies in its inability to handle the large number of memory accesses that the integrated high-performance processor or application-specific logic can generate, as discussed below. Successive accesses may proceed immediately if they do not need to access the same bank, since each memory bank is independent. However, successive accesses that need the same bank must queue up and serialize. To make matters worse, DRAM requires that the sensing, restore, and precharge operations for the selected memory cells be done sequentially during a single access, before the next access may start, or the data stored in the memory cells might be destroyed. As a result, the cycle time of reading and restoring a row in embedded DRAM is typically larger than the access time, and thus limits the ability of individual banks to accept successive accesses [21]. The most straightforward solution is to simply increase the number of independent DRAM banks in order to lower the probability of a conflict. The cycle time can be hidden as long as succeeding references are issued to other banks while a row cycle is completing in any one bank. However, the additional sets of the unshared sense amplifiers at the ends of each bank and the amount of I/O circuitry which are required for every new bank cause an area penalty. The area penalty associated with a 16-bank embedded DRAM compared to a four-bank one is estimated to be about 20%.

In order to solve the problem of the area penalty, all DRAM banks are cascade connected with the internal I/O bus as shown in Fig. 15.4. Sharing the wide internal I/O bus and I/O circuitry (preamplifiers and input buffers) among multiple banks can minimize the area.

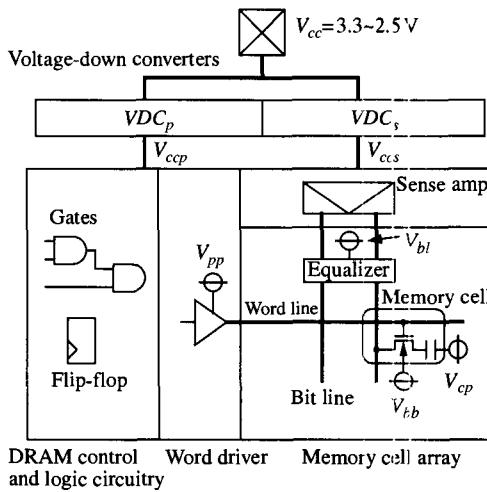
## 15.3 VOLTAGE GENERATOR

This section covers the different types of voltage generators used in the design of DRAMs.

### 15.3.1 Basic Configuration

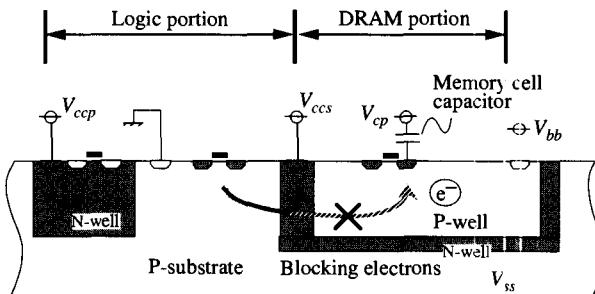
Most commodity DRAMs and embedded DRAM cores employ an internal voltage generator so that they can operate with a single external power supply voltage.

Figure 15.5 shows the basic configuration. At first, the negative back bias voltage,  $V_{bb}$ , the bit line precharge voltage level,  $V_{bl}$ , and the cell plate voltage level,  $V_{cp}$ , are applied to the memory cell array. The voltage-down converters for the sense amplifiers ( $VDCS$ ) and for the DRAM control and logic circuitry ( $VDC_p$ ) generate the  $V_{ces}$  and  $V_{ccp}$  levels, respectively. The boosted word line voltage ( $V_{pp}$ ) applied to the CMOS word line driver improves the random row access time. The value of voltage levels  $V_{ces}$  and  $V_{pp}$  are determined by the characteristics of the memory cell transistors such as the sub-threshold leakage, the threshold voltage, and the maximum applied electric field strength to its gate oxide. For reliable operation, high-speed DRAMs that are built in advanced technologies with thin oxides and short transistor length must be powered with low supply voltages. However, the gate oxide thickness of the memory cell transistor cannot be scaled down. Therefore the thinner gate oxide of the logic transistors is offered by the dual-gate oxide process when the high-performance logic gates are required.



**Figure 15.5** The basic configuration of the internal voltages in DRAM.

The typical configuration of the well structure for the embedded DRAM is shown in Fig. 15.6. The P-well region of the memory cell array is surrounded by the N-well and the bottom N-well which are structured by the triple-well process technology. This structure is quite effective for blocking electron injection from the logic to the memory cell array, which will be discussed later. The commodity DRAM is typically fabricated in a twin-well process to reduce manufacturing costs. The substrate bias of NMOS transistors for the peripheral circuitry is  $V_{bb}$ . However, the triple-well structure is indispensable to the embedded DRAM, because the switching noise caused by the high density logic gates must not interfere with the embedded DRAM operation. The P-substrate bias of NMOS transistors in the logic portion is typically  $V_{ss}$ , as shown in Fig. 15.6.



**Figure 15.6** The triple-well structure of the embedded DRAM.

The voltage generators are required to regulate the internal voltage accurately and to achieve low power consumption. The voltage fluctuation due to temperature, process variations, and external voltage level should be minimized. The operation of each voltage generator is explained in the next section.

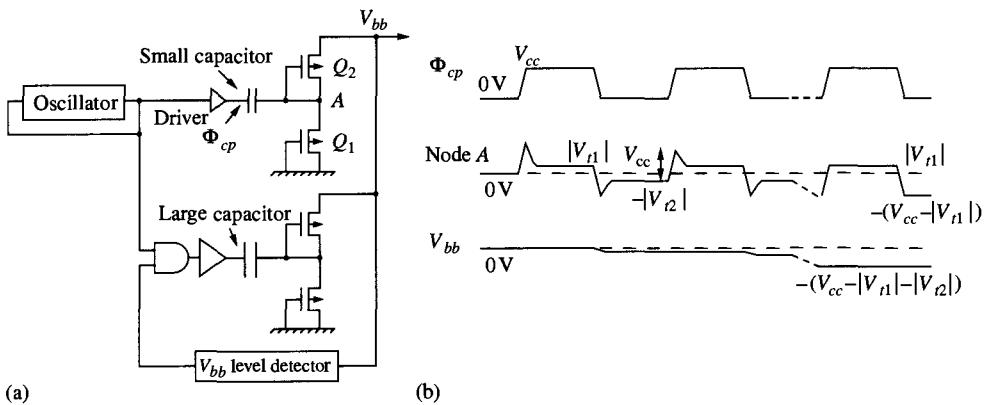
### 15.3.2 Back Bias Generator

The back bias ( $V_{bb}$ ) generator which generates a negative voltage from  $-1.5\text{ V}$  to  $-1.0\text{ V}$  is integrated in the DRAM with the power supply voltage ( $V_{cc}$ ).  $V_{bb}$  is applied to the P-well in which NMOSs are fabricated. The back bias generator is needed for the

following reasons:

1. To reduce the junction capacitance of the NMOS transistors. Since memory cell transistors are formed by NMOS transistors, the parasitic capacitance of bit lines,  $C_b$ , is reduced so that the readout signals ( $\Delta V$ ) are enlarged.
2. To reduce the changes in threshold voltages of the NMOS transistors caused by the back bias effect. This in turn, improves the operational margins of the sense amplifier and the bootstrap word driver.

The basic operation of the  $V_{bb}$  generator is explained [1], [2], [3] in Figs. 15.7(a) and (b). The  $V_{bb}$  generator consists of a ring oscillator, a driver, and a charge pump circuit. The pump capacitor and the two diode-connected NMOS or PMOS transistors are included in the charge pump circuit. No electrons are injected into the P-substrate because the holes are pulled out from P-substrate, when PMOS transistors are used. A timing diagram is shown in Fig. 15.7(b).



**Figure 15.7** (a) The schematic diagram of the back bias ( $V_{bb}$ ) generator. (b) The timing diagram.

1. When  $\Phi_{cp}$  rises from 0V to  $V_{cc}$ , the voltage of node A is boosted by the pump capacitor. The PMOS transistor  $Q_1$  is turned on and node A is discharged to the threshold voltage of  $Q_1$ ,  $|V_{t1}|$ .  $Q_2$  is off, and  $Q_1$  starts to turn off when the voltage level of node A reaches  $|V_{t1}|$ .
2. Next, the voltage of node A is driven to  $-(V_{cc} - |V_{t1}|)$  by the pump capacitor when  $\Phi_{cp}$  falls to 0V. Then, electrons are supplied to  $V_{bb}$  by  $Q_2$  turning on and node A charges to  $-|V_{t2}|$ , the threshold voltage of  $Q_2$ .  $Q_1$  is off, and  $Q_2$  turned off when the voltage level of node A reaches  $-|V_{t2}|$ . As a result,  $V_{bb}$  becomes slightly negative.
3. The operations described in 1 and 2 are repeated as the ring oscillator oscillates and  $V_{bb}$  falls gradually until  $V_{bb}$  reaches  $-(V_{cc} - |V_{t1}| - |V_{t2}|)$ .

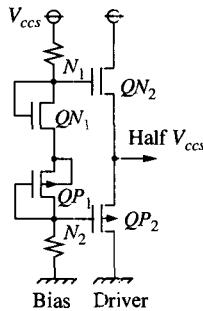
In [4], two pump circuits are connected to  $V_{bb}$  in parallel. The large one is regulated by the  $V_{bb}$  level detector while the small one continuously operates. In this design, the power consumption is reduced by disabling the large pump circuit when  $V_{bb}$  reaches the desired voltage level.

The boosted voltage level of the word line,  $V_{pp}$ , used in the CMOS word driver is also generated in a similar manner.

### 15.3.3 Bit Line Precharge and Cell Plate Voltage Generators

In order to reduce the electric charge consumed by the bit line voltage swing, bit line pairs are precharged to half  $V_{ccs}$  level. This is also the reference voltage for the sensing operation. Moreover, the cell plate voltage of the memory cell is also set to half  $V_{ccs}$  to reduce the electric field stress across the thin film of memory cell capacitors. This voltage level must be generated accurately and must not change with process variations. A low-output impedance is also required in order to maintain a stable voltage level when the power supply voltage and the load on the half  $V_{ccs}$  supply are varied during operation.

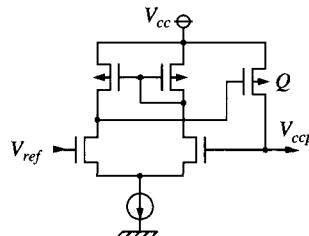
Figure 15.8 [5] shows an example circuit. The push-pull driver transistors that have large driving capability (in the several mA range) are controlled by the bias circuit whose operating current is just a few  $\mu\text{A}$ . Here the voltage levels of  $N_1$  and  $N_2$  are  $V_{ccs}/2 + V_{thn}$  and  $V_{ccs}/2 - |V_{thp}|$ , respectively, where  $V_{thn}$  and  $V_{thp}$  are the threshold voltages of NMOS and PMOS devices, respectively. As a result, half  $V_{ccs}$  is generated by the driver whose gate-source voltage is  $V_{th}$ . The accurate half  $V_{ccs}$  level is obtained even with the process variation, because the transistor characteristics of the device pair of  $QN_1$  and  $QN_2$ , and the device pair of  $QP_1$  and  $QP_2$  are almost identical.



**Figure 15.8** The schematic diagram of the half-voltage generator.

### 15.3.4 Voltage-Down Converter

When the transistors are scaled down, a lower power supply voltage is required for reliable circuit operation. However, users expect the external power supply voltage to be constant for several generations of DRAMs. In order to meet this constraint, the voltage-down converter which converts the external power supply to a lower internal one is widely used in commodity DRAMs. This also enables low power consumption and high performance. Figure 15.9 shows the typical configuration [6]. Here the voltage difference between the voltage reference level,  $V_{ref}$ , and the internal power supply voltage,  $V_{ccp}$ , is detected by the current mirror amplifier whose output regulates the PMOS driver,  $Q$ . When  $V_{ccp}$  becomes lower than  $V_{ref}$ , the gate of  $Q$  is discharged by the current mirror and  $Q$  charges the output node. On the other hand,  $Q$  is turned off by the output of the current mirror when  $V_{ccp}$  becomes higher than  $V_{ref}$ . Thus a stable voltage level of  $V_{ccp}$  is generated. It is crucial to generate a stable  $V_{ref}$  which has minimum dependency on  $V_{cc}$ , temperature change, and process variation. The voltage-down converter affords high current driving capability, and low standby current.



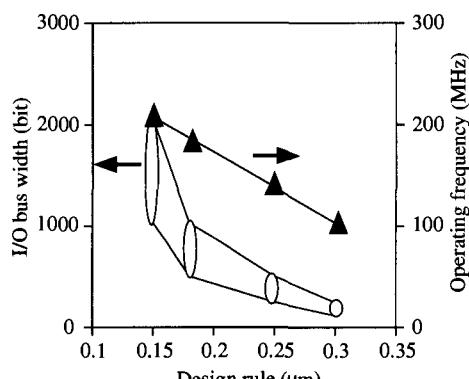
**Figure 15.9** The schematic diagram of the voltage down converter.

## 15.4 EMBEDDED DRAM

### 15.4.1 Features

More than 2 M-bytes of DRAM and a few million gates of logic have been integrated on a single chip fabricated in a  $0.35\text{ }\mu\text{m}$  process technology. The long-promised reality of system-on-chip has finally been realized. Figure 15.10 shows the possible I/O bus width and operating frequency of embedded DRAM versus process design rule. The 128–256 bit of the I/O bus at 100–133 MHz operating frequency of the embedded DRAM has been realized using  $0.25\text{--}0.3\text{ }\mu\text{m}$  design rules. I/O bandwidth of 1.6–4.3 G-byte/s has been achieved. High bandwidths of more than 10 G-bytes/s can be achieved even at 100 MHz operating frequency [7], [8], [9], [12], [13]. To the contrary, the peak bandwidth of the memory bus in the PC system using PC100 synchronous DRAM modules is 800 M-bytes/s. Furthermore, the bandwidth of embedded DRAM could theoretically reach 50 G-bytes/s using  $0.15\text{ }\mu\text{m}$  design rule, if the enormous heat problem is solved. It will be a challenge to effectively utilize such an overwhelming high bandwidth in real applications.

There are several reasons to use embedded DRAM solutions in the market. Some significant potential market benefits are explained by way of example. The first application category requires a relatively small DRAM of less than 4 Mb or 16 Mb. In the industry, there is some concern that someday 4 and 16 Mb DRAMs may disappear from the market or it becomes very difficult to secure commodity DRAM prices. Consequently embedded DRAM may be used to implement a frame buffer for moving pictures and buffer memories for CDROM and HDD. The second application category



**Figure 15.10** The memory bus characteristics of the embedded DRAM according to the design rule.

takes advantage of the significant performance improvement from the embedded DRAM because of wider bus structure with reasonable cost. A 3-dimensional graphics accelerator and a cache memory system controller are the most popular applications. 3D-RAM [7] is a representative example. Logic such as buffers and ALUs to handle the pixel and video data is implemented to obtain the high performance of 3-D rendering graphics. The home game computer and workstations could benefit from the relatively high-quality graphics capability. The third category is a low power system application such as battery operation PDA, the graphic controllers for mobile PCs, and car navigation systems. Power consumption is reduced because wide internal connection between DRAM and logic has less load capacitance and is accessed less frequently. In the near future, applications will use larger DRAM memory sizes and more logic and approach the 100 Mbit plus 1 M gate range. These requirement will increase significantly over time. The Media chip [8] is an example of this type of application. The fourth category is the combined microprocessor with embedded DRAM. M32Rx/D [9] is one example. This works well for the embedded microcontroller systems because the memory requirements are not large. One of the applications which uses M32Rx/D is the digital still camera, because the data compression and decompression can be effectively processed by the specific instruction sets. The software solution can also allow the flexibility of the system design.

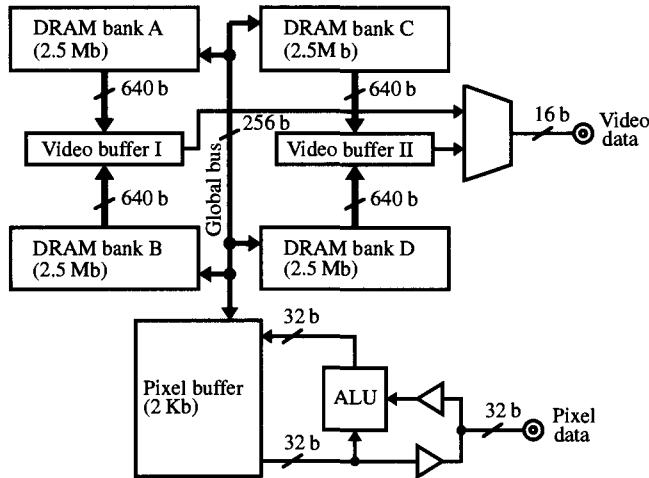
As a research topic, the embedded DRAM has been recently focused in the advanced general purpose microprocessor to overcome the memory wall [23]. The improvements in memory latency have not kept up with the almost exponential increase in processor speed, so system performance is often limited by the memory access time. The memory latency reduction of embedded DRAMs offers a large benefit for bandwidth-intensive applications, such as scientific and database applications [22]. The combination of embedded and multiprocessing systems proposed by I-RAM [10] and PPRAM [11] are examples.

#### 15.4.2 Recent Embedded DRAM LSIs

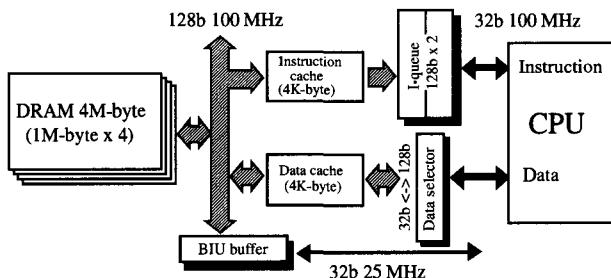
In this section some recent developments in embedded DRAM are described.

Figure 15.11 shows the 3D-RAM [7] designed for rendering 3-D graphics. It has a frame buffer memory and includes support for Z-compare and color blending. The 10 M bit DRAM is divided into four banks, video/pixel buffers, and an ALU. The 256 bit wide internal I/O bus supports the transfer of 400 M pixels/s with the maximum bandwidth of 1.6 G-bytes/s. The Media Chip [8] is another example. It has an 8 M bit DRAM integrated with four pixel processors to implement a 3-D graphics engine. A bandwidth of 6.4 G-byte/s is achieved.

The M32Rx/D [9], a 32-bit RISC processor embedded with a 32 M bit DRAM is shown in Fig. 15.12. It was developed for PDAs, digital still cameras, Internet terminals, and car telephones. A  $0.25 \mu\text{m}$  CMOS 3 metal process technology was used. The chip size measures  $9.7 \times 10.29 \text{ mm}^2$ . This chip consists of the 32-bit RISC CPU, 32-bit  $\times$  16 general purpose registers, 52-bit  $\times$  2 accumulators, a 4 M-byte DRAM, a 4 K-byte instruction cache, a 4 K-byte data cache, an external bus interface unit (BIU), a memory controller, and some peripheral circuits. The CPU, DRAM, cache, and BIU are connected with a single 128-bit 100 MHz internal I/O bus. The external I/O bus is 32 bits wide and operates at 25 MHz. The 128-bit-wide I/O bus reduces instruction and data contention. The external and internal power supplies are 3.3 V and 2.5 V, respectively.



**Figure 15.11** The block diagram of a 3D-RAM enhancing three-dimensional graphic system.



**Figure 15.12** The block diagram of the M32Rx/D which is a 32-bit RISC processor embedded with a 32 M bit DRAM.

### 15.4.3 Process Issues of Embedded DRAM

This section describes the process issues presented by embedded DRAM. There are three major factors that must be considered for embedded DRAM processes: That is, DRAM capacitor choice, the difference of gate density, and that of the transistor performance between DRAM and logic.

We will first compare the density, performance, and process steps that dictate manufacturing costs for commodity DRAMs and logic. The memory cell structure of today's commodity DRAM is three dimensional to obtain large stored charge with high density. The 3-D memory cell structure increases the process steps because of the many polysilicon layers used. By way of contrast a pure logic process can support small amounts of embedded RAM, by using the 3-Tr DRAM cell or SRAM cell which are 10 to 15 times larger than the commodity DRAM 1Tr/1C memory cell. The 3-D DRAM memory cell structure has a stepped surface topology, resulting in coarse metal pitches. DRAM processes with the additional polysilicon layers for the memory

cell and a few metal layers are designed for maximum memory density, while logic processes realize high gate density with three to five layers of fine pitch metallization.

The performance of the transistor in the DRAM process is almost one generation behind that of the logic transistor. The saturation current of the logic transistors is 1.5 to 2 times larger than for the DRAM process. Three main reasons for this difference are outlined below. First, the thermal process for the memory cell capacitor and the thick gate oxide film caused by the usage of  $V_{pp}$  degrade the transistor performance of the DRAM process. Second, in the logic process the drain and channel engineering is optimized for the logic transistor structure. The silicide process suppresses the parasitic resistance in the drain region. The dual-gate structure [24], [25] reduces the subthreshold leakage of the low threshold voltage PMOS transistor. These are key technologies of the high-performance logic process. Fewer steps and lower junction leakage in the DRAM process allow only a simple transistor structure. Third, the threshold voltage of transistors for the DRAM peripheral circuitry including the row and column decoders must be relatively high to satisfy the small standby current. The subthreshold leakage of the transistors fabricated by the DRAM process is required to be 10 to 50 times smaller than that of the logic transistors.

Finally, we compare the process steps from the viewpoint of manufacturing cost. The 3-D memory cell structure brings the complications of the DRAM process. Additional poly layers are required for the storage node, cell plate, and bit lines. As a result the DRAM process costs approximately 1.2 times more than the logic process. When advanced DRAM and logic processes are merged, the cost climbs to at least 1.4 times or higher. It is difficult to balance the conflicting requirements of the DRAM and the logic processes. The DRAM process parameters are optimized for the memory cell, especially the refresh characteristics, while fast transistor performance and high gate density are the major issues of the logic process.

The three major process issues of the embedded DRAM are described as follows.

#### **15.4.3.1 DRAM Cell Capacitor Selection**

We will begin by discussing the memory cell structure in order to understand the integration of DRAM and logic. There are two approaches to building DRAM capacitors: the trench capacitor [14] and the stacked capacitor [15]. Both have been developed for high-density commodity DRAM. The equivalent circuit of both memory cells is  $1Tr/1C$  as shown in Fig. 15.2.

The planar capacitor which was developed before the 1 M bit DRAM generation is compatible with logic process technology. However, the large cell size restricts its use to applications where only small amounts of DRAM are required.

The trench capacitor has the advantage that, due to its flatness, it is easy to fabricate the dense metal interconnects. In addition, the thermal process steps required for the fabrication of the capacitor are carried out before the fabrication of transistors, so that transistor performance is not degraded. However, the increase in the trench aspect ratio makes it difficult to scale down the design rules. For these reasons, the trench capacitor is considered superior to the stacked one for design rules of  $0.25\text{ }\mu\text{m}$  and larger. However, the stacked capacitor has been employed successfully for embedded DRAM, too. Chemical mechanical polishing (CMP) has been used to planarize the topology between the memory cell array and the logic. Also methods have been developed to fabricate the capacitor at lower temperature and the present stacked capacitor fabricated from silicon

oxide with silicon nitride (ON) films subjects the transistors to slightly less thermal stress. Looking to the future, capacitors will be fabricated from high dielectric material such as  $Ta_2O_3$ , *BST*, and *PZT*. The fabrication of transistors must precede that of capacitors because the materials used cannot tolerate high-temperature process steps. Most of the embedded DRAM of the future will be built using stacked capacitors.

#### **15.4.3.2 Gate Density for Embedded DRAM**

As discussed above, the trench capacitor has the advantage of the dense metal pitch for 0.25  $\mu m$  design rules and larger. However, a stacked capacitor is often used for embedded DRAM because the CMP method relaxes the step topology between the memory cell array and the logic. With CMP it is possible to fabricate four to five levels of metal with the tight metal pitch of a logic process.

#### **15.4.3.3 Transistor Performance for Embedded DRAM**

When the thermal stresses of fabrication are lessened by the usage of the high dielectric material as discussed above, the silicide process and the dual-gate structure can be applied to the transistors of the logic portion of the embedded DRAM. Then the problem that remains is the gate oxide thickness that is mismatched between DRAM and logic.

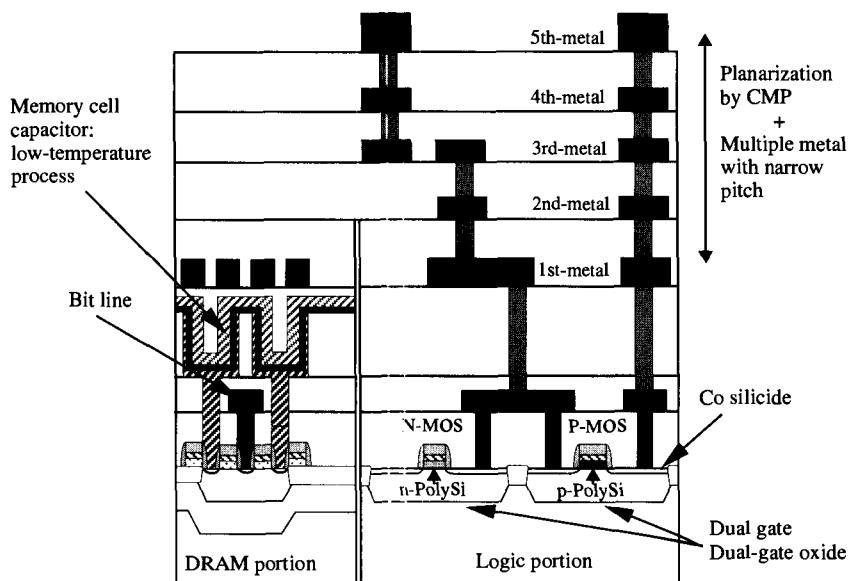
The threshold voltage of the memory cell transistor is adjusted to a higher level than those of the peripheral and logic circuits to enhance dynamic data retention. Moreover, the voltage level of the selected word line is boosted to compensate for the voltage drop caused by the high threshold voltage of the memory cell transistor. As a result, the gate oxide film of the memory cell transistor must be thicker than that of the logic to prevent an excessive electric field across the gate oxide film. The transistor performance gap between DRAM and logic devices is widened further as the process shrinks. One way to avoid penalizing the logic devices is to form the gate oxides of the memory cell transistor and the logic circuits independently. The logic transistors with the thinner gate oxide and the lower threshold voltage enable high-speed operation at the cost of subthreshold leakage current. The usage of dual-gate oxide [25] incorporated with the lower thermal process described above enhances the logic transistor performance in the embedded DRAM. The typical threshold voltage configurations are as follows. In addition to the lower threshold voltage for the logic transistors, three kinds of high threshold voltage transistors are necessary for the memory cell transistors, and PMOS and NMOS transistors used in the DRAM peripheral circuits that handle the high-voltage  $V_{pp}$  level. Compared to the logic process, the increased number of masks is estimated to be seven or more, because of the storage node, storage node contact, bit line, bit line contact, and three types of the high-threshold voltage transistors.

Finally, Fig. 15.13 shows a cross section of the device structure of the embedded DRAM with 0.18  $\mu m$  or smaller design rule.

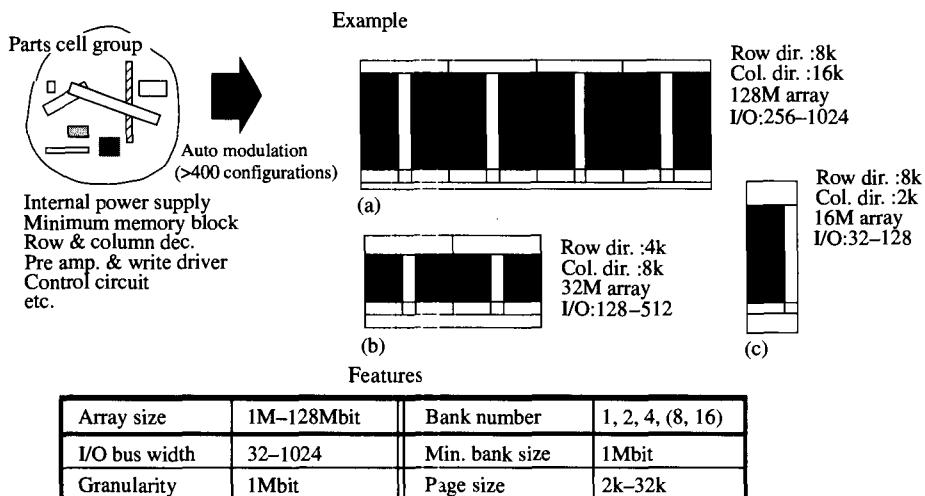
### **15.4.4 Design Issues of Embedded DRAM**

#### **15.4.4.1 DRAM Macro Architecture**

The time required to design the DRAM core must be short, because the memory density and bandwidth must be customized to satisfy the system needs. The DRAM macro provides this design flexibility. The high-density DRAM memory cell cannot be



**Figure 15.13** Example of the device structure of the embedded DRAM with  $0.18\text{ }\mu\text{m}$  or smaller device rule.



**Figure 15.14** DRAM macro modules.

generated automatically, unlike an SRAM cell. The DRAM core is made by combining a few kinds of modules, as shown in Fig. 15.14. The DRAM array module, the voltage generator module, and I/O module are automatically arranged. The DRAM array module includes memory cells, sense amplifiers, row and column decoders, and the timing control signal generator. As can be seen from the configurations of the DRAM core shown in Fig. 15.14, the size of the DRAM macro can be easily varied by changing the number and combination of modules. The data bus width is flexibly

changed when the data bus is connected in the wired-or manner. The resulting DRAM macro has an area penalty of less than 10% when compared to a full custom solution. But it can be designed in a shorter time.

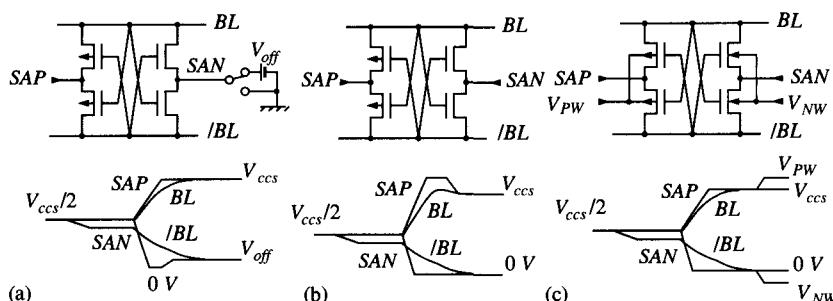
#### 15.4.4.2 Interference between DRAM Core and Logic

The high-frequency switching of the logic circuits can interfere with the operation of embedded DRAM [16]. Some sources of interference are heat transmission, power/ground noise, substrate noise, and electron injection. As for thermal interference, the package should be designed to account for the effect of junction temperature and the refresh characteristics of the DRAM. Dynamic data retention is seriously degraded by noise. To solve this problem, a triple-well structure is used as shown in Fig. 15.6. The triple-well structure under the DRAM portion also improves immunity to alpha-induced soft errors, so that less stored charge is needed for reliable operation. Noise immunity is also improved by careful floor planning of the power and ground lines. The power and ground lines of the memory cell array from those of the logic portion are often separated.

#### 15.4.4.3 Low-voltage Operation

The refresh characteristics and the operation of the sense amplifier are major issues as DRAM operating voltage is lowered. The subthreshold leakage current increases exponentially as the threshold voltage of the memory cell transistor is decreased. As a result, the refresh characteristics degrade with transistor scaling. The negative word line technique is one way to solve this problem [17]. In addition the boosted sense ground scheme inhibits the current flow from the storage node of the memory cell through the off memory cell transistor [18]. As shown in Fig. 15.15(a), the gate-source voltage of the memory cell transistor is kept at a negative level by raising the low level of the bit line to  $V_{off}$ .

The operating margin of the DRAM in terms of the voltage range and the row access time are mainly determined by the characteristics of the sense amplifier. As explained in the previous section, the bit line pairs are precharged to the half  $V_{ccs}$  level. At the beginning of the sensing operation, the gate-source voltage of the latched sense amplifier reaches half  $V_{ccs}$  at most. In addition, the bit line precharge level of half  $V_{ccs}$  raises the threshold voltage due to the backbias effect of the MOS transistor. As a result, the speed degradation of the sense amplifier will be large when the operating voltage is lowered. The overdrive sensing scheme shown in Fig. 15.15(b) enhances the



**Figure 15.15** (a) Schematic diagram of the boosted sense ground scheme. (b) The overdrive sensing scheme. (c) The well-drive sensing scheme.

PMOS amplification by boosting the common-source node *SAP* at the start of the sensing operation [19]. The well voltage of the sense amplifier is synchronized with the bit line voltage level in the well-driving scheme shown in Fig. 15.15(c) [20]. The elimination of the back gate bias reduces the threshold voltage at the start of the sensing operation and improves the sensing speed.

#### 15.4.5 The Future of Embedded DRAM

Digital electronic products require both high performance and low power consumption. The embedded DRAM technology which can integrate the memory, logic, and microprocessor on a single chip will be used to satisfy these requirements. One of most successful applications, a 2-D/3-D graphic controller for mobile PCs shown in Fig. 15.16 illustrates the merits of the embedded DRAM.

As shown in Fig. 15.16, a compact system was realized by integrating several LSI chips. Moreover the power consumed by the interconnection among the discrete LSI components is eliminated, resulting in lower power consumption. Portable system and home electronic products will be the first to benefit. However, performance enhancement is realized only when the circuit architecture of the logic portion is optimized to take advantage of the large available bandwidth. Math-intensive applications such as 3-D graphics are other candidates that can use the high bandwidth effectively. Graphic controllers for mobile PCs and home game computers can benefit by using embedded DRAM. Moreover these systems will require more memory, higher bandwidth, and larger gate counts over time. With 0.18  $\mu\text{m}$  design rules, 32 M–100 Mb DRAM and 500 K–1.5 M logic gate can be integrated on a single chip with a die size of 150  $\text{mm}^2$ . On the other hand, it is more cost effective to use SRAM cells when less than 8 M bits of integrated memory are needed because the 6-Tr CMOS-SRAM cell can be fabricated in a normal logic process. The lower manufacturing cost, faster access time, and smaller standby current of SRAMs are very attractive in portable systems and network applications where the demand on memory size is small. Thus on-chip memory capacity is determined by the process technology and die size economies. It is important to have a

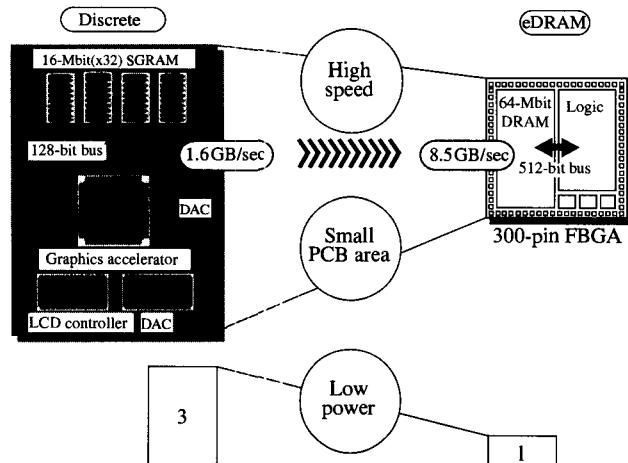


Figure 15.16 The merit of the embedded DRAM. Example: SXGA 2-D/3-D graphic controller for mobile PC.

clear understanding of the system applications that can take advantage of the embedded DRAM considering the memory capacity and the process cost.

The wide internal I/O improves the overall bandwidth of on-chip DRAM data transfers, while the initial access time is larger compared to SRAM. The hardware advances such as nonblocking memory access in advanced CPUs and software advances such as data prefetch in specific media processors have been done in order to hide the penalty of the initial memory access time. However, the clock frequency of the leading CPUs is approaching 1 GHz and the number of the instructions executed per clock is increasing. As a result, much shorter initial access time will be very desirable. A memory array organization of small subarrays with short bit lines and word lines improves the initial access time [26], [13], at the cost of increased area. Pipelined memory array operation techniques can be used to reduce the effective memory cycle time [26].

Embedded DRAM technology has great potential for future electronic products. However, advances must be made in design techniques and fabrication processes to reduce manufacturing costs. This is a big challenge for the future.

## REFERENCES

- [1] R. D. Pashely and A. McCormick, "A 70-ns 1K MOS RAM," *ISSCC Dig. Tech Papers*, pp. 138–139, Feb. 1976.
- [2] M. I. Elmasry, *Digital MOS Integrated Circuits*. IEEE Press, New York, p. 23, 1981.
- [3] Y. Konoshi, et al., "A 38-ns 4-Mb DRAM with a Battery-Backup (BBU) Mode," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1112–1117, Oct. 1990.
- [4] K. Sato, et al., "A 20ns Static Column 1Mb DRAM in CMOS Technology," *ISSCC Dig. Tech Papers*, pp. 254–255, Feb. 1985.
- [5] S. Fujii, et al., "A 50mA Standby 1MW × 1b/256kW × 4b CMOS DRAM," *ISSCC Dig. Tech Papers*, pp. 266–267, Feb. 1986.
- [6] M. Horiguchi, et al., "A Dual-Operating-Voltage Scheme for a Single 5-V 16-Mbit DRAM," *IEEE J. Solid-State Circuits*, vol. 23, no. 5, pp. 1128–1132, 1988.
- [7] K. Inoue, et al., "A 10 Mb 3D Frame Buffer Memory with Z-compare and Alpha-blend units," *ISSCC Dig. Tech Papers*, pp. 300–303, Feb. 1995.
- [8] T. Watanabe, et al., "A Modular Architecture of a 6.4-Gbyte/s, 8-Mbit Media Chip," *Symp. on VLSI Circuit Dig. Tech Papers*, Honolulu, pp. 42–43, June 1996.
- [9] T. Shimizu, et al., "M32Rx/D-A Single Chip Microcontroller with a High Capacity 4MB Internal DRAM," *Hot Chips Conf. Record*, pp. 37–48, Aug. 1998.
- [10] D. Patterson, et al., "Intelligent RAM(IRAM): Chips that Remember and Compute," *ISSCC Dig. Tech. Papers*, pp. 224–225, Feb. 1997.
- [11] K. Murakami, et al., "Parallel Processing RAM Chip with 256Mb DRAM and Quad Processors," *ISSCC Dig. Tech. Papers*, pp. 228–229, Feb. 1997.
- [12] Y. Aimoto, et al., "A 7.68GIPS 3.84GB/s 1W Parallel Image-Processing RAM Integrating a 16 Mb DRAM and 128 Processors," *ISSCC Dig. Tech. Papers*, pp. 372–373, Feb. 1996.
- [13] T. Kimura, et al., "64 Mb 6.8 ns Random ROW Access DRAM Macro for ASICs," *ISSCC Dig. Tech Papers*, pp. 416–417, Feb. 1999.
- [14] M. Taguchi, et al., "Dielectrically Encapsulated Trench Capacitor Cell," *IEDM Dig. Tech. Papers*, p. 136, 1986.
- [15] M. Koyanagi, et al., "Novel High Density, Stacked Capacitor MOS RAM," *IEDM Dig. Tech. Papers*, p. 348, 1978.

- [16] T. Tsuruda, et al., "High-Speed/High-Bandwidth Design Methodologies for On-Chip DRAM Core Multimedia System LSI's," *IEEE J. Solid-State Circuits*, vol. 32, pp. 477–482, March 1997.
- [17] T. Yamagata, et al., "Low Voltage Circuit Design Techniques for Battery-Operated and/or Giga-Scale DRAM's," *IEEE J. Solid-State Circuits*, vol. 30, pp. 1183–1188, Nov. 1995.
- [18] M. Asakura, et al., "A 34 ns 256 Mb DRAM with Boosted Sense-ground Scheme," *ISSCC Dig. Tech. Papers*, pp. 140–141, Feb. 1994.
- [19] T. Kawahara, et al., "A Circuit Technology for Sub-10 ns ECL 4 Mb BiCMOS DRAMs," *Symp. on VLSI Circuit Dig. Tech. Papers*, pp. 131–132, May 1991.
- [20] T. Ooishi, et al., "A Well-synchronized Sensing/Equalizing Method for Sub-1.0 V Operating Advanced DRAM," *Symp. on VLSI Circuit Dig. Tech. Papers*, pp. 81–82, May 1993.
- [21] T. Yamauchi, et al., "The Hierarchical Multi-Bank DRAM: A High-Performance Architecture for Memory Integrated with Processors," *Proc. Advanced Research in VLSI*, pp. 303–319, Sept. 1997.
- [22] T. Yamauchi, et al., "A Single Chip Multiprocessor Integrated with High Density DRAM," *IEICE Trans. Electron.*, vol. E82-C, no.8, pp. 1567–1577, Aug. 1999.
- [23] A. Saulsbury, F. Pong, and A. Nowatzky, "Missing the Memory Wall: The Case for Processor/Memory Integration," *23<sup>rd</sup> International Symp. on Computer Architecture*, pp. 90–101, Philadelphia, PA, 1996.
- [24] M. Tsukamoto, et al., "0.25  $\mu$ m-Polygate Dual Gate and Buried Metal on Diffusion Layer (BMD) Technology for DRAM-Embedded Logic Device," *Symp. on VLSI Tech. Dig.*, pp. 23–24, 1997.
- [25] M. Togo, et al., Multiple-Thickness Gate Oxide and Dual-Gate Technologies for High Performance Logic-Embedded DRAMs," *IEDM Tech. Dig.*, pp. 347–350, 1998.
- [26] Y. Sato, et al., "Fast Cycle RAM (FCRAM); A 20 ns Random Access, Pipe-Lined Operating DRAM," *Symp. VLSI Circuits Dig. Tech. Papers*, June 1998, pp. 22–25.

PART  
**VI**

## **INTERCONNECT AND I/O**

Noel Menezes  
*Intel Corporation*  
Lawrence Pileggi  
*Carnegie Mellon University*

## 16.1 INTRODUCTION

As has been long predicted, interconnect wiring delays rather than transistor logic delays are the major contributor to global path delays on microprocessors fabricated in current deep submicron CMOS processes. The increasing dominance of the interconnect coupled with the aggressive scaling in operating frequencies to increase microprocessor performance has fundamentally altered the nature of microprocessor design. The impact of the interconnect, therefore, needs to be considered during all stages of design and at all levels of the design hierarchy. Even during process development, interconnect performance is an important consideration in the design of the on-chip metal system [1].

Spurred by the growing importance of the interconnect, several novel techniques for analyzing various aspects of interconnect performance have been developed to supplement traditional circuit simulation methods that have been applied to this problem. However, before we study these techniques, we first explore a few issues which relate to the current state of affairs.

### 16.1.1 Process Scaling, Interconnect Scaling, and Noise

Chapter 2 of this book provides a detailed description of how CMOS processes have scaled during the past few years. Recognizing the poor scaling properties of the interconnect, process technologists have used various methods to reduce the impact of the interconnect on overall microprocessor performance. A primary means of reducing the relative delay of the interconnect has been to increase the aspect ratio of the metal layers in order to lower the per-unit-length resistance of on-chip interconnects (nets) [1]. While this method has been effective in containing the interconnect delay problem to some degree, the taller, thinner conductors of modern processes exhibit an increase in the ratio of the *coupling capacitance* between adjacent conductors to the total capacitance. Moreover, the larger die sizes made possible by advances in CMOS technology have resulted in an increase in the average interconnect length. Hence, in addition to delay, the noise induced by the interconnect has become an equally important consideration in microprocessor design. (The recent usage of electrically superior

materials such as copper and low- $\epsilon$  dielectrics to improve interconnect performance, while providing a one-time improvement in interconnect performance, will not alter the scaling trends of future processes with respect to noise.)

### 16.1.2 Local and Global Interconnect

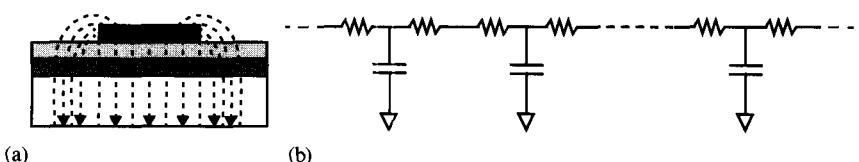
In general, all nets on current microprocessors can be divided into two rather broad categories: local nets that connect transistors and gates within a block, and global nets that interconnect the blocks that constitute the microprocessor. Busses and clock nets make up a large portion of the global nets in current microprocessor designs. (The power supply grid which can also be regarded as the ultimate global net is the subject of an entire chapter in this book!) Since they occur at a higher level of the chip hierarchy, global nets tend to be long and, therefore, exhibit delays that span several clock cycles. Furthermore, these nets are usually routed on the upper low-resistance metal layers of the chip which tend to have high relative coupling capacitances per unit length. Hence, these nets also induce the most noise. Because of their poor latency (delay) and noise characteristics, and their critical role in determining overall microprocessor throughput, careful consideration is given to these nets by different designers including architects, floorplanners, physical designers, and timing verification engineers. It is on these nets that the interconnect effects associated with the submicron era were first felt. Hence, to a large degree, in the context of microprocessor design, interconnect analysis refers to the analysis of these global nets.

Depending on the size of the block they occupy, local nets can also be fairly long but there are far fewer problematic nets at the intra-block level of the design hierarchy. Furthermore, the design techniques outlined in the next chapter are effective in alleviating the interconnect problems associated with long local nets.

### 16.1.3 Interconnect Modeling

We would like to briefly address the modeling of on-chip interconnect structures, the simplest of which is a single line over an ideal ground plane. For frequencies up to the several hundred megahertz range, this line is well modeled by a distributed RC line comprised of incremental RC sections [2] (Fig. 16.1). Furthermore, several empirical and theoretical studies have shown that such a uniform distributed RC line can be modeled by a few (3–4) segments depending on the length and the frequency range of interest [3]; that is, the electrical response of a finite-section RC ladder circuit is almost identical to that of a distributed RC transmission line.

Gigahertz processors now appear on the roadmaps of several semiconductor and computer companies. At such high frequencies, the inductive component of some interconnect structures is no longer negligible with respect to the resistance, that is,



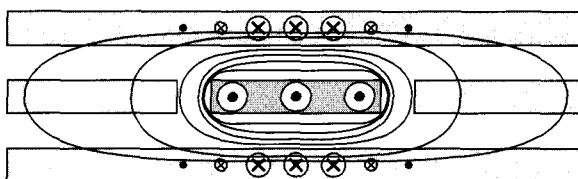
**Figure 16.1** (a) Single line over a silicon substrate; (b) lumped T-section ladder RC representation transmission.

$\omega_{max}L \approx R$ . This is especially true for clock nets which tend to have wide low-resistance wires and for buses consisting of identical parallel traces that can be considered a single entity for electrical purposes when they all switch in the same direction.

To date, the need for inductance extraction and modeling has been alleviated by design strategies which minimize the formation of significant long-range induction on chip. These approaches employed to avoid inductance as much as possible can, however, be suboptimal in terms of IC area and performance.

One straightforward attempt to limit on-chip induction is found in the recent Alpha chip design [4]. The wiring layers containing lines with high current density (hence, high inductive capability) are sandwiched between isolating metal planes above and below, as shown in Fig. 16.2. This design methodology has the following benefits:

- The magnetic field is blocked by the metal planes since currents induced in the isolation layer compensate the field outside completely. The coupling to adjacent layers and the substrate is suppressed.
- Since the layer distance is smaller than the wire width for the levels being isolated, the inductive coupling distance *within* the isolated layer is also reduced significantly. Hence, the inductance coupling is localized.
- The current return paths are now known by design. This makes calculating the loop inductance with a 2-D model reasonably accurate.

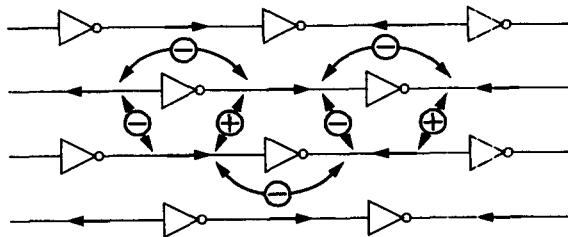


**Figure 16.2** Isolating layer approach (cross-sectional view). The magnetic field is blocked in the vertical direction and weakens quickly in the horizontal direction.

These advantages are at the manufacturing cost of two new metal layers for each original layer susceptible to major inductive interaction. These isolating layers also add significantly to the capacitive coupling of the surrounding circuitry which adversely impacts speed and power dissipation.

A more selective method of reducing the inductance between high-current wires is to introduce additional inverters (not buffers) in the lines at regular intervals. This causes the current to change direction in each adjacent interval. The self-inductance of the entire line is reduced significantly, because the mutual coupling between neighboring segments of the same line subtract from the total inductance of the line rather than adding to it. The same effect can be found for neighboring, parallel lines if the buffer inverters are staggered with respect to each other, as depicted in Fig. 16.3.

With enough added inverters, the reduction of the self-inductance by inverter insertion can cause the inductive impedance to become insignificant compared with the ohmic impedance. At such a point, the inductance models are no longer required since RC propagation dominates. The negative side effects of buffer insertion are

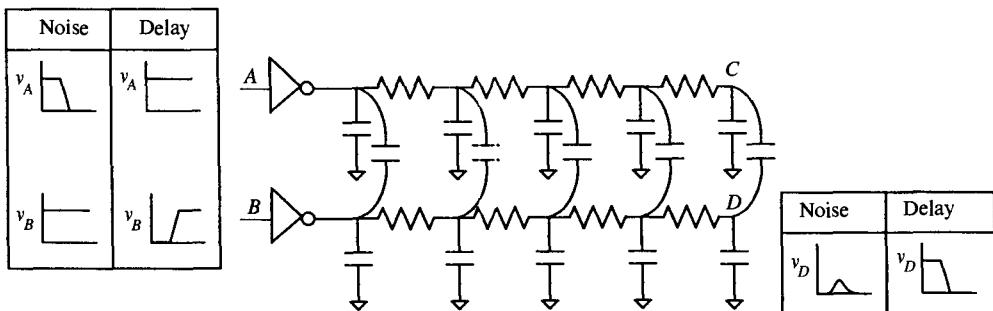


**Figure 16.3** Staggered inverter pattern with assumed current directions and mutual inductance polarities.

slightly higher power consumption and increased IC area, and a possible reduction in circuit performance due to the additional delay introduced by the inverters. Hence, inductance effects need to be taken into account during analysis in some current microprocessor designs. A complete coverage of inductance extraction is beyond the scope of this chapter. However, a recent summary can be found in [5] and [6].

#### 16.1.4 The Duality between Noise and Delay

For reasons outlined earlier, the relatively large coupling capacitances associated with current processes exacerbate the signal integrity problem. This effect is easily illustrated via the circuit of Fig. 16.4. If the input to the driver of the bottom inverter is high, node D remains low. However, if the top inverter switches from high to low, the “attacker” line (node C) transitions from low to high, which in turn causes a noise blip at the far end of the bottom “victim” line (node D) because of the coupling between the two lines. This excursion of the signal value from its stable level due to the coupling capacitances is considered *noise*, and can lead to circuit failure.



**Figure 16.4** Excitation and response waveforms for noise and delay computation.

In the presence of large coupling capacitances an identical circuit situation can be used to demonstrate the impact of coupling on delay computation on the delay of the bottom line. The only difference is that in this case the driver for the bottom line switches instead of staying at a stable level. In conclusion, we state that in the presence of coupling capacitances, delay and noise computation work in an almost identical manner. In fact, computing the noise waveform for a particular net may yield the delay value for another net. (Actually, in a later section we will show that for estimating the worst-case delay for the victim line we would want the top aggressor line to switch in the opposite direction while the victim is in transition.) With this background, we

now describe the most commonly applied techniques to efficiently analyze interconnects from the viewpoints of delay and noise.

## 16.2 SIMPLIFIED INTERCONNECT ANALYSIS

The Elmore delay model [7], [8] was applied during the early eighties to analyze the delay of transistor-level circuits, which were modeled as RC trees. It is natural, therefore, that the Elmore delay is also used to analyze the wiring of high-performance microprocessors.

### 16.2.1 The Elmore Delay and Its Properties

For the circuit of Fig. 16.5 where the driver is modeled by a resistor driven by a voltage source, the Elmore delay at node 5 can be written as follows:

$$\begin{aligned} T_{d5} &= \sum_{i \in P(5)} R_i C_{di}, \\ &= R_1(C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7) + R_2(C_2 + C_3 + C_4 + C_5) \\ &\quad + R_3(C_3 + C_4 + C_5) + R_4(C_4 + C_5) + R_5 C_5 \end{aligned} \quad (16.1)$$

In the above equation,  $P(5)$  represents the path from node 5 to the root of the tree, while  $C_{di}$  represents the sum of the capacitors that are downstream of each resistor  $R_i$  that lies along this path.

One of the appealing properties of the Elmore delay model as applied to RC trees is that it can be easily written by inspection for any node of the tree. For designers, this property enables quick back-of-the-envelope delay estimates for driver-net-receiver combinations. In addition, the Elmore delay is calculated in linear time for RC trees—only two traversals of the RC tree are required in order to calculate the Elmore delay for every node of the RC tree [8]. This allows for software programs

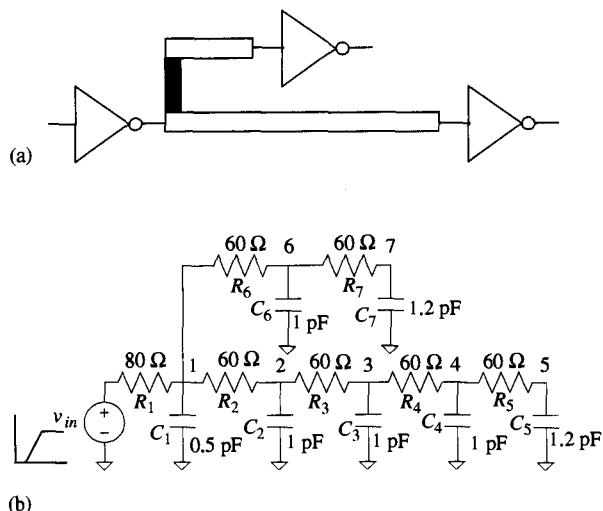
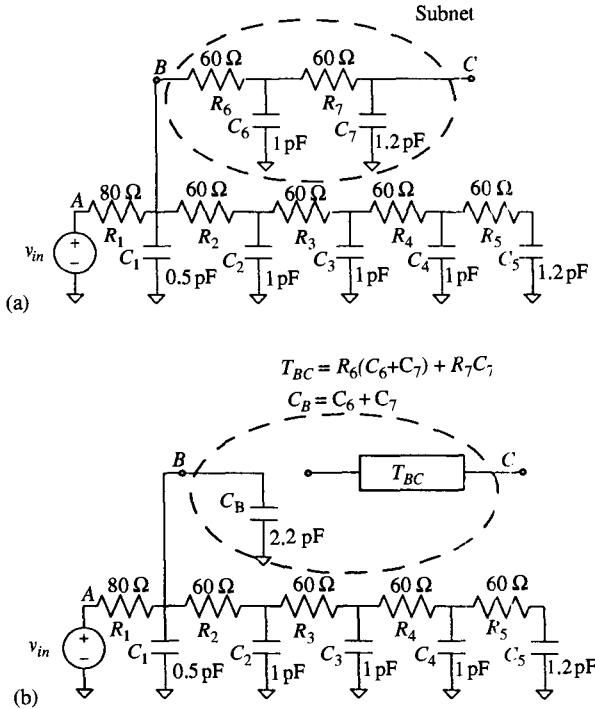


Figure 16.5 (a) Typical on-chip driver-net-receiver combination; (b) the equivalent RC model.



**Figure 16.6** (a) A partial subnet of an RC tree; (b) compact Elmore delay hierarchical representation of the subnet.

that can complete the delay analysis (rollups) for the tens of thousands of nets in current microprocessors within minutes. Another important property which can be easily derived from the closed-form formula for the Elmore delay in Eq. (16.1) is its hierarchical nature. A partial net (e.g., the subnet of Fig. 16.6(a)) can be compactly represented by its total capacitance to represent its loading effect on the main net as shown in Fig. 16.6(b). Furthermore, the Elmore delay to downstream node C of the subnet from the driver,  $T_{AC}$ , is obtained by adding the subnet Elmore delay,  $T_{BC}$ , to the delay of the branching point B,  $T_{AB}$ , in the main net of Fig. 16.6(b). This compact, hierarchical representation is especially useful in microprocessor design where parts of the same interconnect may belong to blocks and are generated by different designers or automated routing programs. The hierarchical and linear complexity properties of the Elmore delay model are exploited in several physical design and circuit optimization programs [10].

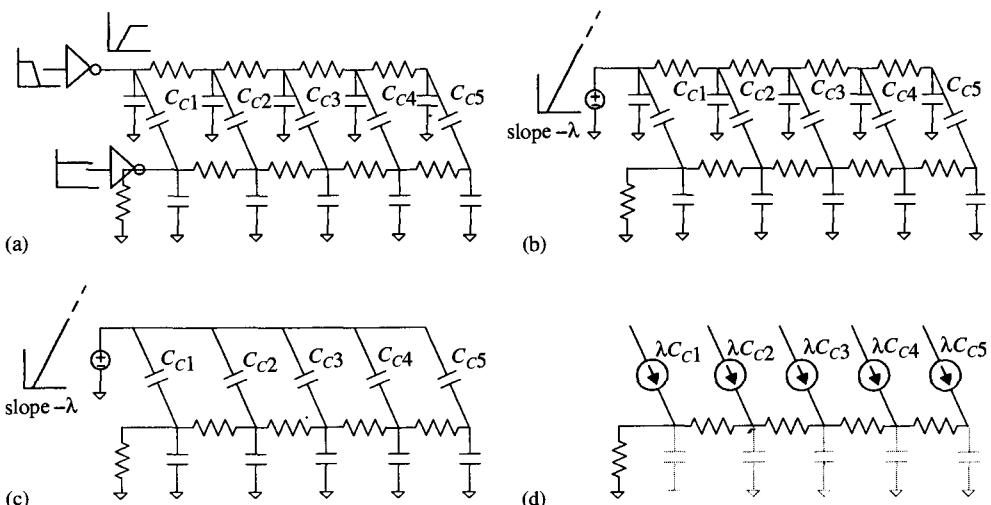
In spite of all these advantages, one of the main drawbacks of the Elmore delay model is that it can be a significant overestimate of the actual delay. It was recently shown in [11] that the Elmore delay is a theoretical upper bound for the 50% delay for a family of input waveforms. This is especially true for nodes that are closer to the driver which are driven by sharper signal waveshapes.

In conclusion, the Elmore delay has been successfully applied to several aspects of high-performance microprocessor design. Its applicability to the timing verification of high-performance microprocessor designs as a reduced accuracy but efficient timing model for transistor stacks which can be used effectively in conjunction with circuit simulation was demonstrated in the design of a state-of-the art 600 MHz commercial microprocessor [12].

### 16.2.2 A Noise Bound for Coupled Interconnect

As emphasized earlier in this chapter, the increasing aspect ratios of current CMOS processes have led to an increase in the relative contribution of the interconnect coupling capacitance. This makes noise analysis especially important for future designs. The circuit model of Fig. 16.4 can be used to describe most coupling scenarios. While these scenarios are easily analyzed by a circuit simulator, a model with the simplicity of the Elmore delay model would prove extremely useful. Just such a model recently developed in [13] provides a bound on the maximum noise for each node of a typical tree-like interconnect structure. Setting aside the rigorous derivation of [13], we derive this bound using an intuitive approach.

We begin with the circuit of Fig. 16.7(a) where the driver of the quiet victim line is modeled as a grounded resistor—a reasonable approximation since the output transistor of the victim driver is in its linear region of operation for the small excursions of the driver output voltage that characterize most noise waveforms. We then apply a sequence of conservative circuit transformations: First, we assume that the slope of the output signal of the attacker driver,  $\lambda$ , is known (or, more likely, conservatively estimated). We can, therefore, replace the attacker driver by a saturated ramp voltage source which approximates the output waveform. Replacing this saturated ramp waveform by an infinite ramp waveform of the same slope will only serve to increase the amplitude of the noise “bump” at every node along the victim line [Fig. 16.7(b)]. Finally, removing the resistors and grounded capacitors of the attacker line will ensure that the same infinite ramp source will inject noise into every attacker coupling capacitor [Fig. 16.7(c)]. This, again, is a conservative transformation since the waveshapes at the attacker coupling capacitors would have a smaller slope than the infinite ramp if all elements of the attacker line were taken into account. Observing the voltage–current



**Figure 16.7** Noise bound derivation: (a) Original coupled network; (b) modeling the attacker by an infinite ramp source; (c) applying the same infinite ramp at all coupling capacitors; (d) dc circuit for computing the steady state of the noise waveform obtained by removing grounded capacitors.

relationship,  $i_C = CdV_C/dt$ , for a capacitor, every attacker capacitor,  $C_{ci}$ , can be replaced by a constant current source of value  $\lambda C_{ci}$  as shown in Fig. 16.7(d). The maximum voltage achieved by each node along the victim line will be at  $t = \infty$  when the grounded victim capacitors are charged to their final value. In other words, solving for the steady state of this circuit yields the required noise bound. This is equivalent to solving the dc circuit obtained by removing all the grounded capacitors from the circuit of Fig. 16.7(d). The solution for this dc circuit yields the maximum noise voltage that can be induced at any node  $j$  of the victim line

$$N_{max,j} = \lambda \sum_{i \in P(j)} R_c C_{dci} \quad (16.2)$$

where  $C_{dci}$  represents the sum of all downstream *coupling* capacitors seen by resistor  $R_i$ . This noise bound may be overly conservative primarily because it does not account for the resistance of the attacking net and the grounded capacitors of the victim net. Also, in [14] it is empirically shown that this bound overestimates the far-end receiver input noise with decreasing attacker ramp slopes (higher  $\lambda$ 's) and with increasing line lengths. However, this model is especially applicable during the early stages of microprocessor design when the coupling capacitances and attacker slopes are only estimated due to lack of detailed routing information. In conclusion, this noise model does enjoy most of the appealing properties of the Elmore delay model—it is hierarchical, has linear computational complexity, is provably conservative, and is accurate enough for certain noise analysis applications.

### 16.3 MODEL ORDER REDUCTION

The previous section describes two first-order models that are widely used for interconnect delay and noise analysis. However, the aggressive scaling of microprocessor operating frequencies in interconnect-dominated submicron processes has necessitated the use of highly efficient interconnect analysis techniques with accuracy comparable to that of a circuit simulator. The conservativeness of these first-order models is a high price to pay in a design environment where picoseconds matter. Fortunately, beginning with the application of Asymptotic Waveform Evaluation (AWE) [9] to the design of the clock tree of the then state-of-the-art 200 MHz Alpha 21064 microprocessor [15], model order reduction techniques (also referred to as moment-matching techniques) have found increasing usage in a variety of interconnect analysis applications. Model order reduction offers excellent accuracy while providing reasonable runtimes as compared to Elmore-based methods. We begin with a detailed exposition of the first of these techniques. AWE was developed as an extension to the work of Penfield and Rubinstein [8] which used the Elmore delay for delay and timing prediction of RC trees. AWE is a technique for approximating the dominant poles (time constants) of linear RLC circuits. Using the dominant poles one can generate approximate time and/or frequency domain responses. AWE has dramatically improved dominant pole/zero analysis for analog circuit design optimization, but it has had an even greater impact on time domain analysis of RLC interconnect.

The approximate dominant poles are generated in AWE by moment matching. Moments, as defined in classical mechanics or probability theory, are related to frequency domain coefficients. The essence of AWE is that these coefficients, or moments,

can be calculated easily and efficiently for linear circuits in general, then mapped to a set of dominant poles and zeros.

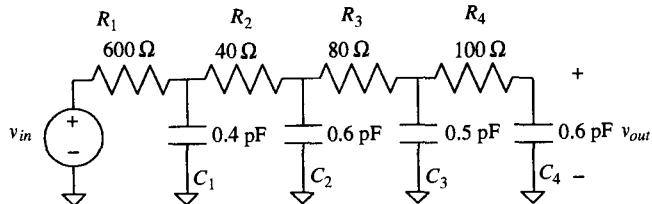
### 16.3.1 An Example

To demonstrate the steps for calculating moments and obtaining dominant poles, we will use the simple RC circuit shown in Fig. 16.8. It is usually more efficient to analyze linear circuits such as the one in Fig. 16.8 in the frequency domain as opposed to the time domain. For example, we can express the transfer function of this circuit ( $V_{in} = \delta(t)$ ) as

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{1 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{1 + b_1 s + b_2 s^2 + \cdots + b_m s^m} \quad (16.3)$$

where  $m > n$ . For a fourth-order RC circuit,  $m = 4$  and  $n \leq 3$ . We can factor the numerator and the denominator of Eq. (16.3) to display the poles and zeros explicitly:

$$H(s) = \frac{\left(1 + \frac{s}{z_1}\right)\left(1 + \frac{s}{z_2}\right) \cdots \left(1 + \frac{s}{z_n}\right)}{\left(1 + \frac{s}{p_1}\right)\left(1 + \frac{s}{p_2}\right) \cdots \left(1 + \frac{s}{p_m}\right)} \quad (16.4)$$



**Figure 16.8** A simple RC circuit example.

Coming up with the rational form in Eq. (16.3) or all of the poles and zeros in Eq. (16.4) is nontrivial for a large circuit. Therefore, we will instead approximate the transfer function in the complex frequency domain as a series in powers of  $s$ :

$$H(s) = m_0 + m_1 s + m_2 s^2 + m_3 s^3 + \cdots \quad (16.5)$$

The coefficients of the power series terms, the  $m_j$ 's, are proportional to the moments that we will match with AWE—more on this in the next section. For now, we will relate  $H(s)$  in Eq. (16.5) to an expansion of the rational transfer function form in Eq. (16.3). That is, expanding Eq. (16.3) about  $s = 0$ , or equivalently, dividing the denominator into the numerator in Eq. (16.3), yields  $H(s)$  as an infinite series in powers of  $s$ :

$$\begin{aligned} H(s) &= 1 + (a_1 - b_1)s + (a_2 - b_2 - b_1 a_1 + b_1^2)s^2 \\ &\quad + (a_3 - b_3 - a_1 b_2 + 2b_1 b_2 - a_2 b_1 + a_1 b_1^2 - b_1^3)s^3 + \cdots \end{aligned} \quad (16.6)$$

Comparing Eqs. (16.5) and (16.6) we can recognize the  $m_j$  coefficients as a function of the numerator coefficients ( $a_j$ 's) and the denominator coefficients ( $b_j$ 's) of the transfer function in Eq. (16.3). We can also set Eq. (16.3) equal to Eq. (16.5), and multiply both sides of the equation by the denominator of Eq. (16.3), resulting in the following for our

fourth-order circuit:

$$\begin{aligned} & (1 + b_1 s + b_2 s^2 + b_3 s^3 + b_4 s^4)(m_0 + m_1 s + m_2 s^2 + \dots) \\ & = 1 + a_1 s + a_2 s^2 + a_3 s^3 \end{aligned} \quad (16.7)$$

Collecting the first four power of s-terms ( $s^0 \rightarrow s^3$ ) in Eq. (16.7), we can express the numerator polynomial coefficients in terms of the  $m_j$ 's and the  $b_j$ 's:

$$\begin{aligned} a_1 &= m_0 b_1 + m_1 \\ a_2 &= m_0 b_2 + m_1 b_1 + m_2 \\ a_3 &= m_0 b_3 + m_1 b_2 + m_2 b_1 + m_3 \end{aligned} \quad (16.8)$$

Note that for this RC tree example, the dc gain is 1, hence,  $a_0 = 1$ . In general,  $a_0$  is equal to  $m_0$ . The next four power of s-terms ( $s^4 \rightarrow s^7$ ) in Eq. (16.7) express the coefficients of the pole polynomial in terms of the  $m_j$ 's:

$$\begin{aligned} 0 &= m_0 b_4 + m_1 b_3 + m_2 b_2 + m_3 b_1 + m_4 \\ 0 &= m_1 b_4 + m_2 b_3 + m_3 b_2 + m_4 b_1 + m_5 \\ 0 &= m_2 b_4 + m_3 b_3 + m_4 b_2 + m_5 b_1 + m_6 \\ 0 &= m_3 b_4 + m_4 b_3 + m_5 b_2 + m_6 b_1 + m_7 \end{aligned} \quad (16.9)$$

We have shown that if we had the first eight  $m_j$ 's for this fourth-order system (more on how to compute these  $m_j$ 's in the next section), we can uniquely specify the poles and the zeros for this circuit. That is, we can rearrange Eq. (16.9) as a matrix problem to determine the  $b_j$ 's:

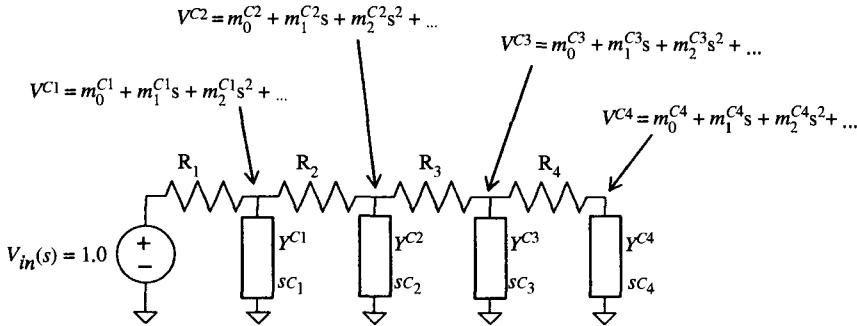
$$\begin{bmatrix} m_0 & m_1 & m_2 & m_3 \\ m_1 & m_2 & m_3 & m_4 \\ m_2 & m_3 & m_4 & m_5 \\ m_3 & m_4 & m_5 & m_6 \end{bmatrix} \begin{bmatrix} b_4 \\ b_3 \\ b_2 \\ b_1 \end{bmatrix} = -\begin{bmatrix} m_4 \\ m_5 \\ m_6 \\ m_7 \end{bmatrix}. \quad (16.10)$$

Once the  $b_j$ 's are obtained we can calculate the four poles for this circuit by solving a fourth-order polynomial (the denominator of Eq. (16.3) for  $m = 4$ ). For this example, the four poles are 0.72296572, 15.038701, 55.767850, and 119.30382, all multiplied by  $10^9$ . The zeros are evaluated using Eq. (16.8). Notice that in general, for a  $q$ th order circuit, the first  $2q$   $m_j$ 's, or moments, can uniquely specify the circuit poles and zeros.

### 16.3.2 Calculating Moments

In the previous section we showed how the first eight  $m_j$ 's for our example circuit could be used to calculate the poles and zeros for a fourth-order circuit. This is a useful approach only if the  $m_j$ 's are easy to calculate. In this section we demonstrate the ease with which these coefficients can be obtained for our lumped linear RC example. Extensions are easily made to the lumped linear RLC case [9].

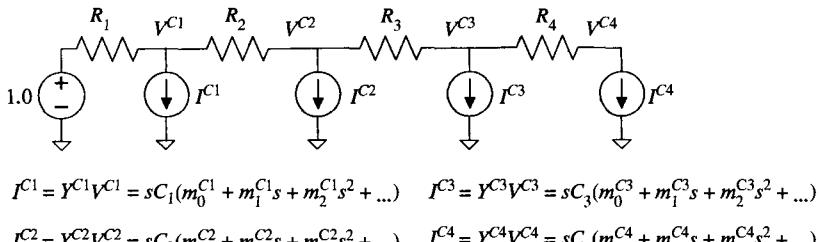
Returning to the example in Fig. 16.8, the impulse response in the complex frequency domain can be analyzed in terms of the circuit in Fig. 16.9, where capacitors have been replaced by their complex admittances. Furthermore, setting  $V_{in}(s) = 1$  makes  $V_{out}(s) = H(s)$ . Let's assume that each of the capacitor voltages (which in this circuit are also the node voltages) is expressed in terms of an infinite series in powers of  $s$ , as shown in the figure. (The superscripts for the  $m_j$ 's in Fig. 16.9 denote that all of



**Figure 16.9** The  $s$ -domain representation of the RC circuit in terms of complex admittances.

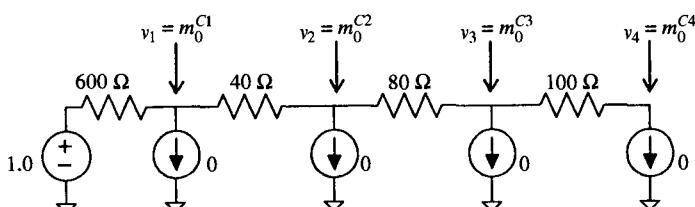
the  $m_j$ 's are different from one node to the next. That is, the  $m_j$ 's of Eq. (16.10) are the  $m_j^{C4}$  coefficients of Fig. 16.9.)

Expressing the capacitor voltages in this way and knowing the capacitor admittances, we can write similar expressions for the capacitor currents, as shown in Fig. 16.10. Moreover, knowing the capacitor currents in terms of the capacitor voltages, we can replace the complex admittances by current sources, as shown. The  $m_j$  terms are the only unknowns in Fig. 16.10.



**Figure 16.10** A circuit equivalent of Fig. 16.9 assuming the node voltages solutions of the form shown.

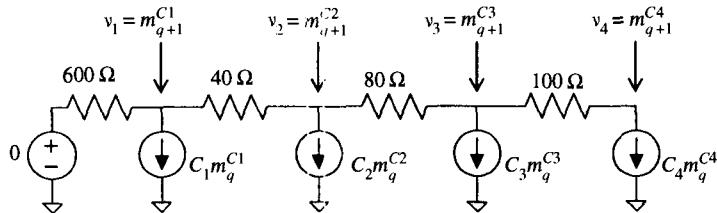
We can now solve for the  $m_0$ 's for all of the capacitor voltages in Fig. 16.10 by setting  $s = 0$  (this corresponds to a dc solution of the circuit). Since there are no constant terms ( $s^0$  terms) in the capacitor currents (they are open for  $s = 0$ ), we set the current sources to zero and solve for the  $m_0$ 's using the dc equivalent circuit in Fig. 16.11.



**Figure 16.11** The dc equivalent circuit used to calculate the  $m_0$ 's for all of the capacitor voltages.

For this RC tree, the  $m_0$ 's are all equal to 1.0. Note that this procedure for replacing capacitors by zero-valued current sources to calculate the  $m_0$ 's holds for all circuit topologies. When there are inductors in the circuit, they are replaced by zero-valued voltage sources when calculating the  $m_0$  terms for their current responses [9].

Referring back to Fig. 16.10, we now solve for the  $s^l$  coefficients, the  $m_1$ 's. It is the  $s$ -terms in the current source expressions which produce the  $m_1$ 's in the voltage responses. Since we know the  $m_0$ 's, the  $s$ -terms in the current source expressions are known. Therefore, we can evaluate the  $m_1$ 's of the voltage responses by setting all of the current sources equal to  $C_j m_0^{Cj}$ , and solving for the node voltages, which are the  $m_1^{Cj}$ 's. The voltage input is a constant, so it does not affect the calculation of any of the terms other than the  $m_0$ 's. Subsequent moments are calculated from the dc equivalent circuit shown in Fig. 16.12 following the same recursion. To generate a complete transfer function for this fourth-order circuit we would calculate the first eight moments from the circuit in Fig. 16.12 recursively.



**Figure 16.12** The dc equivalent circuit used to calculate the higher-order moments for the RC circuit in Fig. 16.8.

### 16.3.3 Moment Matching—AWE

If we expand the transfer function in Eq. (16.3) into its partial fractions (or pole-residue form),

$$H(s) = \frac{1 + a_1 s + a_2 s^2 + \dots + a_n s^n}{1 + b_1 s - b_2 s^2 + \dots + b_m s^m} = \sum_{i=1}^m \frac{k_j}{s - p_j} \quad (16.11)$$

we observe that for the case of distinct poles the time domain impulse response is

$$h(t) = \sum_{i=1}^m k_j e^{p_j t} \quad (16.12)$$

where the  $p_j$ 's are the poles and the  $k_j$ 's are the corresponding residues.

Equations (16.11) and (16.12) show the relation between the time and frequency domain responses. Of course, we can also characterize the time-frequency domain relationship directly in terms of the Laplace transform of  $h(t)$  by

$$H(s) = \int_0^\infty h(t) e^{-st} dt \quad (16.13)$$

Expanding  $e^{-st}$  about  $s = 0$  in Eq. (16.13) yields the following series in powers of  $s$ :

$$\begin{aligned} H(s) &= \int_0^\infty h(t) \left[ 1 - st + \frac{1}{2}s^2 t^2 - \frac{1}{6}s^3 t^3 + \dots \right] dt \\ &= \sum_{k=0}^{\infty} \frac{(-1)^k}{k!} s^k \int_0^\infty t^k h(t) dt \end{aligned} \quad (16.14)$$

That is, the  $q$ th coefficient of the transfer function,  $H(s)$ , is

$$m_q = \frac{(-1)^q}{q!} \int_0^\infty t^q h(t) dt \quad (16.15)$$

The reference to moment matching comes from the fact that these  $m_j$ 's are related to moments by the  $(-1)^q/q!$  term. That is, the  $n$ th moment of a function  $h(t)$  is defined to be  $\int_0^\infty t^n h(t) dt$ . As stated previously, even though the fourth-order circuit of Fig. 16.8 is described exactly by the first eight moments, we can derive a unique first-order approximation by matching the first two moments. We can, by inspection for this simple circuit, calculate the first two moments for the voltage response at  $C_4$ :

$$\begin{aligned} m_0^{C4} &= 1.000 \\ m_1^{C4} &= -1.476 \end{aligned} \quad (16.16)$$

Note that we have scaled the R's to be  $\text{k}\Omega$ 's and the C's to be pF so that the units of time would be nanoseconds when calculating the moments in Eq. (16.16). It is important to scale the moments in this way in general, otherwise the higher order moment values explode, and numerical round-off is a significant problem for moment matching.

A first-order approximation,  $h(t) = \hat{k}_1 e^{\hat{p}_1 t}$ , is obtained by matching the first two moments of the model to those of the actual circuit. In the time domain, the moments of  $h(t)$  are

$$\begin{aligned} m_0 &= \int_0^\infty h(t) dt = \int_0^\infty \hat{k}_1 e^{\hat{p}_1 t} dt = -\frac{\hat{k}_1}{\hat{p}_1} \\ m_1 &= -\int_0^\infty t h(t) dt = -\int_0^\infty t \hat{k}_1 e^{\hat{p}_1 t} dt = -\frac{\hat{k}_1}{\hat{p}_1^2} \end{aligned} \quad (16.17)$$

which yields  $\hat{p}_1 = -1/1.476$ ,  $\hat{k}_1 = 1/1.476$  for a first-order AWE approximation. In general, AWE is a  $q$ th-order extension of this dominant pole approach in that  $2q$  moments are used to generate  $q$ th-order approximations, where  $q$  is less than  $m$  (the order of the actual circuit).

Higher-order AWE approximations are more conveniently solved by applying the moment-matching formulas from Section 16.3.1, which permit us to first calculate the coefficients of the polynomial for the poles from Eq. (16.10) via moment matching. That is, while all of the moment-matching arguments in Section 16.3.1 were for an  $m$ th order approximation for an  $m$ th order circuit, they are also the moment-matching equations for a  $q$ th-order approximation too. The only difference here is that we are assuming that the system is  $q$ th-order, while it is actually  $m$ th order, where  $q < m$ .

Hence, a second-order approximation of the form

$$h(t) := \hat{k}_1 e^{\hat{p}_1 t} + \hat{k}_2 e^{\hat{p}_2 t} \quad (16.18)$$

for capacitor  $C_4$  of Fig. 16.8 is obtained by solving a lower-order matrix similar to Eq. (16.10)

$$\begin{bmatrix} m_0^{C4} & m_1^{C4} \\ m_1^{C4} & m_2^{C4} \end{bmatrix} \begin{bmatrix} b_2 \\ b_1 \end{bmatrix} = -\begin{bmatrix} m_2^{C4} \\ m_3^{C4} \end{bmatrix} \quad (16.19)$$

which yields the characteristic polynomial for the approximate poles,  $\hat{p}$ 's,

$$b_2 \hat{p}^2 + b_1 \hat{p} + 1 = 0 \quad (16.20)$$

Since  $m_2^{C4} = 2.048$  and  $m_3^{C4} = -2.834$ , we obtain the two approximate poles:  $\hat{p}_1 = -0.7227$  and  $\hat{p}_2 = -15.96$ . Notice that the first pole approximation is now extremely close to the first actual pole calculated in Section 16.3.1, while the second pole approximation is modeling the effects of the remaining poles. This type of pole convergence is usual with moment matching. We should also add that it is important to maintain as much precision as possible when solving this matrix problem, and we have used four digits here only to demonstrate the approach. When we use double precision for the moment calculations, the second approximate pole is  $-13.7791$ . Once the pole approximations are known, obtaining the residues is straightforward [9].

### 16.3.4 Applying AWE to Delay and Noise Analysis

Having obtained a reduced form of the transfer function by moment matching for an RC tree, we can easily apply it to delay analysis. For example, if we assume an input step waveform,  $v_{in}(t) = u(t)$ , for the circuit of Fig. 16.8, convolving the reduced transfer function with the input waveform

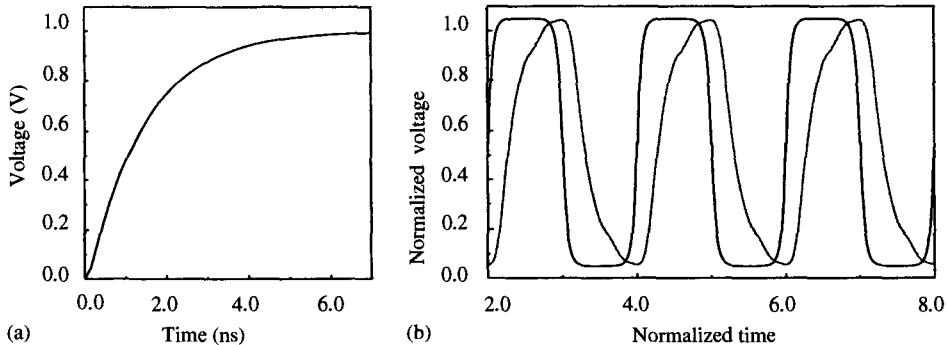
$$V_{out}(s) = V_{in}(s)H(s) = \frac{1}{s} \sum_{j=1}^q \frac{\hat{k}_j}{s - \hat{p}_j} \quad (16.21)$$

yields the output waveform

$$v(t) = \sum_{j=1}^q \frac{\hat{k}_j}{\hat{p}_j} (e^{\hat{p}_j t} - 1) \quad (16.22)$$

A second-order estimate is shown plotted with the exact response in Fig. 16.13(a). Furthermore, we could use six moments to obtain a third-order approximation, but this is hardly necessary given the accuracy of the second-order estimate. We should point out, however, that one usually has to calculate the next order of approximation in order to test the accuracy. In this case, there is little difference between second and third order, hence the approximations would conclude at this level.

Of course it may not seem significant that a fourth-order system can be accurately approximated by one of second order; however, low orders of approximation ( $q = 1$  to  $q = 4$ ) are often the range of necessity for big circuits too. This is because even a circuit with 100,000's of state variables (poles) usually has only a handful of dominant poles in the frequency range of interest. The plot in Fig. 16.13(b) shows the AWE waveform approximation obtained at an output clock node of a large clock tree taken from a commercial microprocessor. The AWE waveforms overlap the simulation waveform almost exactly.



**Figure 16.13** (a) Second-order AWE approximation for the step-input voltage response at  $C_4$  on Fig. 16.8; (b) an input-output waveform pair from a commercial RLC clock tree. The waveforms show both AWE and SPICE results which are indistinguishable at the resolution of this plot.

In general, AWE can be directly applied to any circuit that can be linearized including those similar to that of Fig. 16.7(b) encountered during noise analysis. Another advantage of model order reduction is that it provides an extremely compact characterization of the interconnect. The large RC netlists created by most extraction tools can be “crunched” significantly. Furthermore, accurate analyses can be carried out later for different input waveforms very efficiently avoiding the need for detailed circuit simulation. In fact, for some common input waveshape assumptions like the saturated ramp of Fig. 16.7(d) the output waveform is available in a closed-form manner similar to that of Eq. (16.22). This enables solving for the time-points of interest, like the 50, 20, and 80% points required for calculating delay and slope, respectively, directly without having to generate intermediate points.

We finally note here that the first moment of the impulse response is the familiar Elmore delay! In other words, a first-order AWE uses the Elmore delay as the time constant. This connection is a coincidence since Elmore arrived at his delay metric applying entirely different reasoning.

### 16.3.5 Other Model Order Reduction Techniques

We briefly alluded to some of the numerical problems that can arise if the resistor and capacitor values are not scaled appropriately during moment matching. Furthermore, moment matching, which is recognized as a Padé-type approximation [16], must be applied with care due to its inherent potential for instability. That is, given the set of moments for a stable RLC circuit, generalized moment matching can result in one or more of the approximate dominant poles occurring in the right half plane. Fortunately, the success of AWE in dealing with interconnect analysis problems sparked a flurry of research in applying different model order reduction techniques in order to overcome some of the inherent limitations of AWE. The interested reader is directed toward [17] which describes one of the more successful reduced-order modeling techniques. These techniques have also had some success in dealing with low-loss transmission-line circuits which present a problem to AWE because of the lack of any clearly dominant poles.

## 16.4 DRIVER MODELS

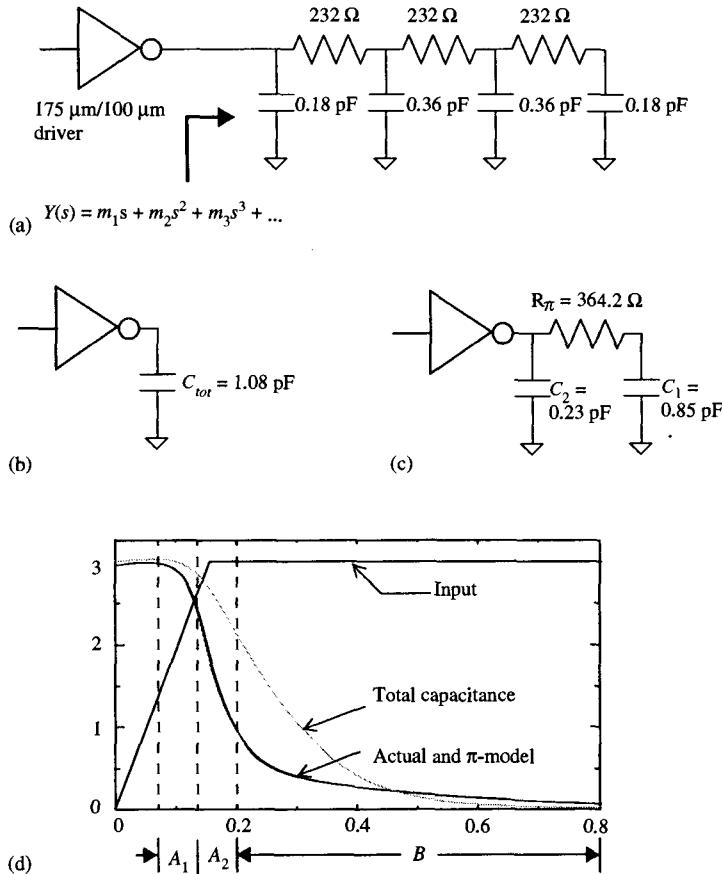
The RC interconnect, in addition to introducing a significant wiring delay in a typical CMOS circuit path, also affects the delay of a CMOS driver. For previous technologies, since the driver resistance dominated the interconnect resistance, the capacitance of the interconnect and the input gate capacitances of the fan-out gates determined the delay of a driver. With scaling, however, the resistance of the RC interconnect effectively shields some of the load capacitance, thereby affecting the CMOS driver delay.

### 16.4.1 Resistance Shielding

Since the resistive component of the RC load seen by the driver is no longer negligible, the total capacitance of the interconnect and fan-out gates is no longer a valid model of the load seen by the gate—the resistance of the RC load needs to be taken into account. The plots of Fig. 16.14 show the error that can be incurred by approximating a 10 mm long interconnect load by its total capacitance. It can be seen that the total capacitance load significantly overestimates the driver delay. The degree of pessimism introduced by this inaccurate load modeling has two effects: First, the drivers of long interconnects are unnecessarily oversized during design. Oversizing drivers has a larger effect than the area penalty incurred—if it turns out that this driver belongs to a bank of identical drivers driving a multibit interconnect bus, the large “ $Ldi/dt$ ” drop on the power grid due to several drivers simultaneously switching may be excessive, which in turn has an effect on the amount of on-chip decoupling capacitance required. In addition, there is a significant power cost due to oversized drivers. Additionally, the larger slope of the total-capacitance waveform in turn leads to an overestimate of the interconnect delay.

This difference between the total-capacitance and actual waveforms is because the *resistance* of the interconnect load *shields* a significant part of the distributed capacitance of the load from the driver. As a result the driver does not see the total capacitance of the load—it only sees an *effective capacitance* [18]. Hence, we need a load model that adequately captures the resistive nature of the interconnects encountered in current technologies. The  $\pi$ -model of the interconnect load introduced in [19] provides just such a load model. The parameters of the  $\pi$ -load are directly synthesized from the first three moments of the interconnect admittance,  $Y(s)$ , as seen by the driver [Fig. 16.14(c)]. These driving-point admittance moments are computed by a procedure similar to that of Section 16.3.2 for computing transfer-function moments. (This  $\pi$ -model load approximation is quite different from that obtained by representing an interconnect by a single  $\pi$ -section. Also, the waveform at the second capacitor  $C_1$  does not represent the waveform seen at the far end of the interconnect.) In Fig. 16.14, we see a good match between the actual waveform and the waveform obtained from a  $\pi$ -load approximation. (The sum of the capacitors of the  $\pi$ -load,  $C_1 + C_2$ , equal the total capacitance of the net,  $C_{tot}$ .)

Essentially, because of the two-pole behavior of the RC load the output waveform for long interconnects is no longer accurately approximated by the typical ramp waveform found in digital circuits. Since the driver output after a typical capacitor discharge behavior in the initial portion of the waveform exhibits a long “tail” portion [Fig. 16.14(d)], approximating the driver output waveform by a ramp leads to a large inaccuracy in estimating the interconnect delay.



**Figure 16.14** (a) Equivalent 3  $\pi$ -section interconnect representation for a 10 mm line; (b) total capacitance load approximation; (c)  $\pi$ -model driving-point admittance approximation; (d) driver output waveforms for (a), (b) and (c).

We now introduce a driver modeling technique that accurately captures the resistance shielding effect for precharacterized drivers.

### 16.4.2 The Effective Capacitance Driver Model for RC Loads

For static timing analysis purposes, in most design methodologies, every gate's delay and output signal behavior is precharacterized as a function of input signal transition time,  $t_{in}$ , and load capacitance,  $C_L$ , by carrying out multiple SPICE runs at different values of  $t_{in}$  and  $C_L$ . The data for the 50% delay,  $t_d$ , and the gate-output waveform 20–80% (or 10–90%) transition (rise or fall) time,  $t_r$  or  $t_f$ , are fitted to empirical  $k$ -factor equations [20] or stored in two-dimensional tables:

$$\begin{aligned} t_d &= k(t_{in}, C_L) \\ t_{r/f} &= k'(t_{in}, C_L) \end{aligned} \quad (16.23)$$

In the previous section, we saw that the  $\pi$ -model synthesis of the RC load is a better approximation of the driving-point admittance. Unfortunately, the obvious generalization of the  $k$ -factor equation method (that is, precharacterizing the gate in terms of the  $\pi$ -model parameters:  $C_2$ ,  $R_\pi$ , and  $C_1$ ) would be prohibitively expensive. More importantly, it is practically impossible to predict the range of interest for the  $\pi$ -circuit component values.

In order to preserve the simplicity and efficiency of the  $k$ -factor equations a more desirable approach would be to map the more complex load to a single capacitance value, the *effective capacitance* (Fig. 16.15). This effective capacitance can then be inserted in Eq. (16.23) to determine the delay of the gate as well as the gate-output waveform. Furthermore, since the waveform at the output of the driver can be approximated from the transition time obtained by applying the effective capacitance value to Eq. (16.23) then the waveform at the far end of the interconnect can be obtained using the moment-matching techniques of the previous section. We describe the modeling approach proposed in [18] which has been successfully applied to the timing verification of a family of high-speed microprocessors [20].

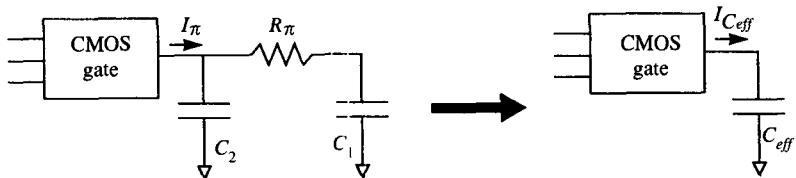


Figure 16.15 The effective capacitance paradigm.

This model is based on the observation that the waveform at the driver output for a resistive load can be split into two regions: a waveform similar to that of a purely capacitive load in the beginning of the transition [region A of Fig. 16.14(d)] which extends for a significant part of the waveform, and a tail portion which exhibits exponential decay at the end (region B). Region A can be split into two subregions: subregion  $A_1$  shows a quadratic waveshape up to the 20% time-point,<sup>1</sup> and subregion  $A_2$  exhibits ramplike behavior from the 20% time-point to the 50% time-point which yields the output waveform

$$V_{out}(t) = \begin{cases} (V_i - ct^2), & 0 \leq t \leq t_{20} \\ a + b(t - t_x), & t_{20} \leq t \leq t_{50} \end{cases} \quad (16.24)$$

where  $a$ ,  $b$ , and  $c$  are determined from Eq. (16.23). An effective capacitance approximation can be derived by equating the average current drawn by the  $\pi$ -load to that drawn by a pure capacitor in region A for the waveshape approximation of Eq. (16.24), that is,

$$\frac{1}{t_{50}} \int_0^{t_{50}} I_\pi(t) dt = \frac{1}{t_{50}} \int_0^{t_{50}} I_{C_{eff}}(t) dt \quad (16.25)$$

The value of the capacitive load that satisfies Eq. (16.25) is the effective capacitance. Algebraic manipulation of Eq. (16.25) yields the effective capacitance approximation

$$C_{eff} = C_2 + C_1 \left[ 1 - \frac{RC_1}{t_{50} - t_{20}/2} + \frac{(RC_1)^2}{(t_{50} - t_{20}/2)t_{20}} e^{-\frac{(t_{50}-t_{20})}{RC_1}} \left( 1 - e^{-\frac{t_{20}}{RC_1}} \right) \right] \quad (16.26)$$

<sup>1</sup>  $x\%$  time-point is defined as the time at which the waveform reaches  $x\%$  of its final value relative to the beginning of the driver input waveform transition.

It should be noted that in the above equation, the time-points,  $t_{20}$  and  $t_{50}$ , are functions of the capacitive load which are related to the delay and transition time of the driver by

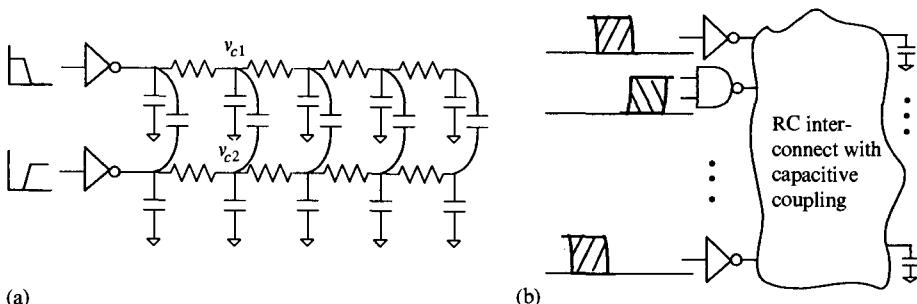
$$\begin{aligned} t_{50} &= t_d + \frac{t_{in}}{2} \\ t_{20} &= t_d + \frac{t_{in}}{2} - \frac{t_f}{2} \end{aligned} \quad (16.27)$$

Hence, the process begins with an initial guess for the effective capacitance,  $C_{eff} = C_{tot}$ . Next  $t_{20}$  and  $t_{50}$  are computed for this effective capacitance guess from Eq. (16.23) and Eq. (16.27). The time-point values are substituted into Eq. (16.26) to obtain a new estimate for  $C_{eff}$ . This process continues until the  $C_{eff}$  values from two successive iterations are within a specified tolerance. It can be shown that these iterations are guaranteed to converge to a single value if the delay and transition time of the driver computed by Eq. (16.23) increase monotonically with  $C_L$  [21].

The effective capacitance approximation captures the driver output waveform in region *A*—however, no single capacitance value can capture the exponential tail. In [18], recognizing that the driver is in the linear region of operation for region *B*, in this region the driver is modeled by a resistor driving the  $\pi$ -load with an initial voltage across  $C_2$  which corresponds to the voltage at the end of region *A*. Hence, the final driver output waveform approximation is split into three piecewise-continuous regions: an initial quadratic waveshape followed by a linear followed by an exponential. This waveform approximation is convolved with the interconnect transfer function computed in Section 16.3 to obtain the far-end response. A refinement of this modeling technique is presented in [22].

### 16.4.3 N-Port Driver Models

Given the dominance of coupling capacitances in current technologies, determining the delay of a driver and its interconnect load for a coupled circuit is an important problem. For the circuit in Fig. 16.16(a), if we assume that both balanced drivers and interconnects are identical, and the waveshapes at the driver inputs are identical but complementary, then for each coupling capacitor the voltages at its terminals are complementary. In other words, during a time interval if the voltage at one terminal



**Figure 16.16** Worst-case delay computation: (a) A symmetric coupled circuit; (b) a multidriver coupled circuit with inputs switching during different time intervals.

changes by a certain amount, the voltage at the other terminal changes by an equal amount in the opposite direction. Hence, the current through each coupling capacitor is

$$i_C(t) = C \frac{dv_C}{dt} = C \left( \frac{dv_{C1}}{dt} - \frac{dv_{C2}}{dt} \right) = 2C \frac{dv_{C1}}{dt} \quad (16.28)$$

Hence, instead of analyzing the coupled circuit of Fig. 16.16(a), a similar delay result can be achieved by analyzing a driver-interconnect circuit where each coupling capacitor has been replaced with an equivalent grounded capacitor whose capacitance equals that of the coupling capacitor multiplied by a “coupling factor” of 2. However, recognizing that the inputs to the drivers may switch at significantly different times which would make the output signals noncomplementary, timing verification engineers sometimes choose to use an empirical coupling factor of less than 2 for most interconnects, and perform a more detailed coupled-circuit analysis for interconnects that lie along the critical paths of the design. Finally, we point out that contrary to popular belief, applying a coupling factor of 2 does not necessarily yield the worst-case delay. If the two drivers are not equally sized, the interconnect for the weaker driver may require a coupling factor greater than 2 to yield the correct “coupled-circuit” delay.

A generalized version of this important coupling delay problem is depicted in Fig. 16.16(b) where there are many drivers incident on a coupled ( $n$ -port) interconnect network consisting of interconnects of different geometries. Also, the input signals to these drivers switch during different time intervals. In this scenario for the purposes of timing analysis, we are interested in the worst-case delay from any one of the driver inputs to its interconnect load output. The interested reader is directed toward [23] for a detailed description of a technique for obtaining the receiver waveforms for such scenarios. This iterative method of [23] uses the technique of Section 16.4.2 for computing the effective capacitance for a single-driver interconnect in its inner loop.

## 16.5 CONCLUSION

With the dominance of interconnect in CMOS technologies, detailed analysis of the interconnect effects has become an essential part of the design flow. Rather than compile a survey of the various techniques from the literature, a detailed view of some of the dominant analysis techniques that are applied in the design of high-performance microprocessors has been presented. While these techniques were first applied to patch the design tool flows of an earlier device-dominated era to handle interconnect-dominated designs, current EDA (electronic design automation) tools are becoming increasingly sophisticated by incorporating these techniques directly into an interconnect-centric design methodology. Alternatives and extensions to the techniques presented here can be found in recent conference and journal literature: most often in the technical digests of the ICCAD and DAC conferences, and the *IEEE Transactions on Computer-Aided Design*.

## REFERENCES

- [1] M. Bohr, “Interconnect Scaling—the Real Limiter to High-performance ULSI,” *International Electronic Devices Meeting Technical Digest*, pp. 241–244, 1995.
- [2] H. Hasegawa, M. Furukawa, and H. Yanai, “Properties of Microstrip Line on Si-SiO<sub>2</sub> Systems,” *IEEE Trans. Microwave Th. Tech.*, vol. 19, no. 11, pp. 869–881, Nov. 1971.

- [3] Vasant B. Rao, "Delay Analysis of the Distributed RC Line," *Proc. of the 32nd ACM/IEEE Design Auto. Conf.*, pp. 370–375, June 1995.
- [4] D. Bailey and B. Benschneider, "Clocking Design and Analysis for a 600-MHz Alpha Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1627–1633, Nov. 1998.
- [5] M. Kamon, et al., "Interconnect Analysis: From 3-D Structures to Circuit Models," *Proc. of the 36th ACM/IEEE Design Auto. Conf.*, pp. 910–914, June 1999.
- [6] M. Beattie and L. T. Pileggi, "IC Analyses Including Extracted Inductance Models," *Proc. of the 36th ACM/IEEE Design Auto. Conf.*, pp. 915–920, June 1999.
- [7] W. C. Elmore, "The Transient Analysis of Damped Linear Networks With Particular Regard to Wideband Amplifiers," *J. Applied Physics*, vol. 19, no. 1, pp. 55–63, 1948.
- [8] P. Penfield and J. Rubinstein, "Signal Delay in RC Tree Networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.
- [9] L. T. Pillage and R. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 4, pp. 352–366, April 1990.
- [10] J. Cong, L. He, C. -K. Koh, and P. H. Madden, "Performance Optimization of VLSI Interconnect Layout," *INTEGRATION, The VLSI Journal*, pp. 1–94, 1996.
- [11] R. Gupta, B. Tutuianu, B. Krauter, and L. T. Pileggi, "The Elmore Delay as a Bound for RC Trees With Generalized Input Signals," *Proc. 32nd ACM/IEEE Design Auto. Conf.*, pp. 364–369, June 1995.
- [12] N. Nassif, M. P. Desai, and D. H. Hall, "Robust Elmore Delay Models Suitable for Full Chip Timing Verification of a 600 MHz CMOS Microprocessor," *Proc. 35th ACM/IEEE Design Auto. Conf.*, pp. 230–235, June 1998.
- [13] A. Devgan, "Efficient Coupled Noise Estimation for On-chip Interconnects," *Proc. of the Intl. Conf. on Computer-Aided Design*, pp. 147–151, Nov. 1997.
- [14] K. Rahmat, J. Neves, and J. -F. Lee, "Methods for Calculating Coupling Noise in Early Design: A Comparative Analysis," *Proc. of the Intl. Conf. on Computer Design*, pp. 76–81, Oct. 1998.
- [15] D. W. Dobberpuhl, et al., "A 200-MHz 64-b Dual Issue CMOS Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, pp. 1555–1567, Nov. 1992.
- [16] G. A. Baker, Jr., *Essentials of Padé Approximants*. Academic Press, Reading, MA, 1975.
- [17] A. Odabasioglu, M. Celik, and L.T. Pileggi, "PRIMA—Passive Reduced-order Macromodeling Algorithm," *IEEE Trans. Computer-Aided Design*, vol. 17, no. 8, pp. 645–654, Aug. 1998.
- [18] J. Qian, S. Pullela, and L. T. Pillage, "Modeling the Effective Capacitance for the RC Interconnect of CMOS Gates," *IEEE Trans. Computer-Aided Design*, vol. 13, no. 12, pp. 1526–1535, Dec. 1994.
- [19] P. O'Brien and T. Savarino, "Modeling the Driving-point Characteristic of Resistive Interconnect for Accurate Delay Estimation," *Proc. of the Intl. Conf. on Computer-Aided Design*, pp. 512–515, Nov. 1989.
- [20] R. E. Mains, T. A. Mosher, L. P. P. van Ginneken, and R. F. Damiano, "Timing Verification and Optimization for the PowerPC Processor Family," *Proc. Intl. Conf. on Computer Design*, pp. 390–393, 1994.
- [21] C. Ratzlaff, S. Pullela, and L. T. Pillage, "Modeling the RC-interconnect Effects in a Hierarchical Timing Analyzer," *Proc. IEEE Custom Integrated Circuits Conference*, pp. 15.6.1–15.6.4, May 1992.
- [22] F. Dartu, N. Menezes, J. Qian, and L.T. Pillage, "A Gate Delay Model for High-speed CMOS Circuits," *Proc. 31st ACM/IEEE Design Auto. Conf.*, pp. 576–580, June 1994.
- [23] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi, "Determination of Worst-case Aggressor Alignment for Delay Calculation," *Proc. of the Intl. Conf. on Computer-Aided Design*, pp. 212–219, Nov. 1998.

Shannon V. Morton  
*Compaq Computer Corporation*

## 17.1 INTRODUCTION

It cannot be disputed that an ever-increasing emphasis is being placed on the design of interconnect structures for high-performance computing. In the not too distant past it was deemed adequate to represent a length of interconnect as a lumped capacitance to ground; however, as technologies scaled this model split into a fixed capacitance (to ground) and a mutual capacitance to adjacent wires. When the conductor resistance ceased to be negligible (with respect to the on resistance of the driver) this model split further into a distributed set of RC elements. We now find ourselves in a regime in which inductance/transmission-line effects also need to be considered, either through explicit modeling or appropriate design management [1]–[3].

This chapter is organized into three categories. The first (Section 17.2) deals with various technology trends that have lead to greater complexity in the design of interconnect structures. The second category (Sections 17.3–17.5) covers the problems and solutions associated with systems of interconnect which exist between repeater structures (or similar logic elements) regarding capacitance, inductance, and resistance. The third category (Section 17.6) deals with techniques by which data can be rapidly and reliably transmitted across long electrical distances.

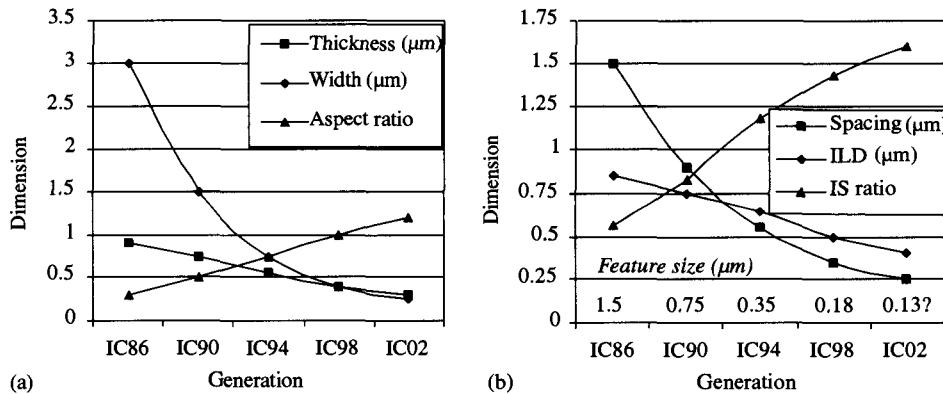
## 17.2 TECHNOLOGY SCALING TRENDS

This section provides an analysis of the general scaling trends of on-chip interconnects, with specific focus on their impact to crosstalk noise, signal delay, etc. This analysis is also closely associated with the scaling trends of CMOS devices, which is presented in a separate chapter of this text.

### 17.2.1 Relative Spatial Dimensions

Figure 17.1 illustrates the interconnect dimensions for a fine pitch layer used for routing buses, and is shown across four process technologies which are representative of multiple evolutionary generations from around 1986 to 1998, with extrapolation into the year 2002.

The aspect ratio is increasing with each technology node due primarily to yield and manufacturing limitations in scaling the interconnect thickness. Other factors include the requirement of minimizing dishing (concavity) when patterning wider metal lines,



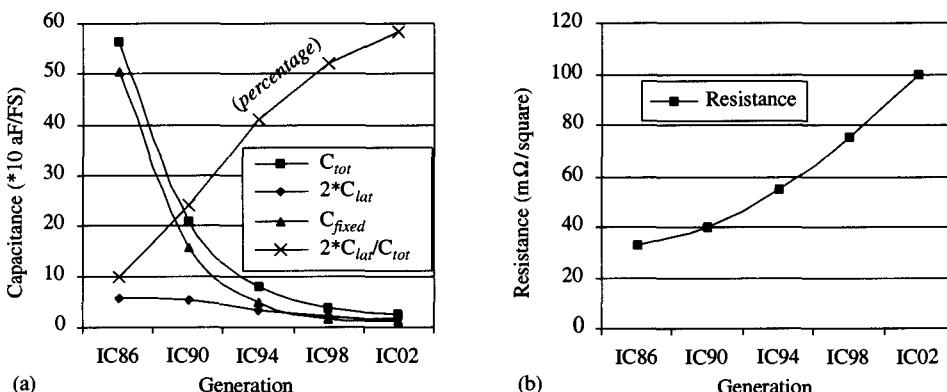
**Figure 17.1** Interconnect and insulator dimensions for a range of generic process generations.

minimizing variations in sheet resistance due to the CMP process, and trading off increased capacitance for reduced resistance. The first two arguments also apply to the dielectric thickness as is evidenced by the similar increase in IS ratio (defined as the ILD thickness divided by the minimum metal spacing). Due to the consistent increase in aspect and IS ratios it is important to analyze their impact on capacitance, resistance, and inductance.

### 17.2.1.1. Effect on Capacitance

If we assume for the moment that each layer is sandwiched above and below by a metal plane then we can extract the total, fixed, and lateral capacitances of a conductor per unit feature length (assuming a uniform dielectric constant of 4.0), as illustrated in Fig. 17.2(a).

Due to the increase in aspect and IS ratios the lateral capacitance is scaling down much slower than the fixed capacitance, and as a result  $C_{lat}/C_{tot}$  is steadily increasing.



**Figure 17.2** (a) Capacitance parameters per unit feature size, and (b) the wiring resistance per square.

This is of great concern since the switching activity of neighboring signals now needs to be considered in designing for critical paths and race conditions, whereas in the past the error associated with lumping this lateral capacitance to ground was small. It is often assumed that the capacitance to the second neighbor is negligible due to the isolation of the intermediate conductor. The validity of this assumption may be brought into question as the IS and spacing-to-width ratios continue to increase.

### 17.2.1.2 Effect on Resistance

Figure 17.2(b) plots sheet resistance ( $R_s$  in  $\Omega/\text{sq}$ ) for each technology node assuming a uniform resistivity of  $\rho = 3 \mu\Omega \cdot \text{cm}$ . The sheet resistance is steadily increasing despite the fact that the metal thickness ( $T$ ) is being scaled down slower than the feature size to trade-off an increase in capacitance for a reduction in resistance. Given that  $R_s = \rho/T$  then the only way to maintain constant sheet resistance between generations is to keep the thickness unchanged, which is of course impractical due to the extremely large aspect ratios and lateral capacitances that would result.

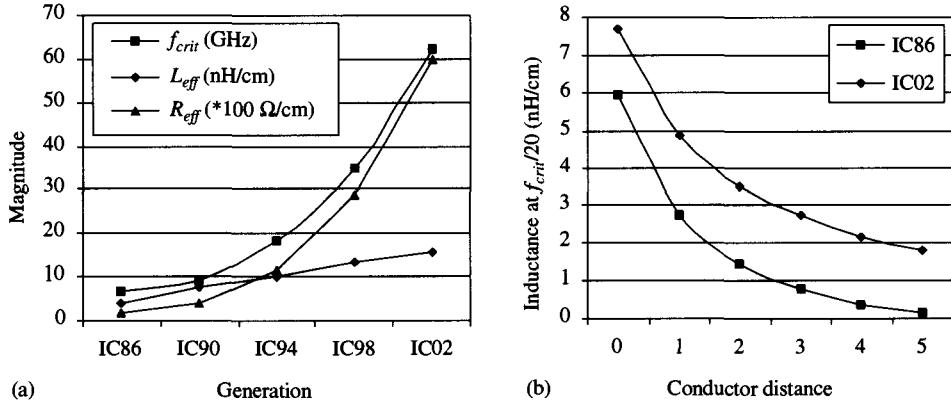
Another resistance concern regarding technology scaling is the effect of the cladding. Before and after the deposition of aluminum (Al), a thin layer of highly resistive material such as titanium or titanium nitride is deposited. The underlayers act to enhance the electromigration properties of the interconnect, and the overlayers are used to improve lithographic patterning and the yield and reliability of via contacts. The concern with scaling the thickness dimension is that the cladding thickness does not scale in the same proportion, resulting in a potential increase in the gradient of Fig. 17.2(b) through future technologies.

This issue is exacerbated with copper (Cu) interconnect since the cladding serves the additional purpose of preventing seepage into the surrounding dielectric. This requires additional barrier cladding at the sides of the copper deposition. The curve of Fig. 17.2(b) may therefore increase more dramatically for Cu than for Al. One further issue with the side-wall cladding of Cu interconnect is that the sheet resistance is now a function of width. The age-old assumption that one need only count the “number of squares” to calculate resistance no longer applies.

### 17.2.1.3 Effect on Inductance

The technology scaling trends not only result in a significant increase in conductor resistance but also give rise to a moderate increase in inductance (at low frequency) as illustrated in Fig. 17.3(a), for the case of a wide bus above a plane. This is due to the large increase in aspect ratio despite the slight reduction in ILD thickness, thereby resulting in a wider field dispersion. As such, the critical frequency ( $f_{crit}$ ) at which  $\omega L = R$  is progressively increasing ( $f_{crit}$  can be used as a figure of merit to gauge the impact of inductance relative to a target edge rate). This in fact helps prevent inductance effects from being a more problematic design consideration than they presently are. However, if the fastest edge rates ( $T_{edge\ rate}$ ) on-chip have a dominant frequency component ( $f_{dom}$ ) that is a significant fraction of  $f_{crit}$ , then inductance effects need to be considered. The dominant frequency is sometimes defined by correlating the edge rate to the fastest slope of a sinusoid:  $f_{dom} = 1/\pi T_{edge\ rate}$ .

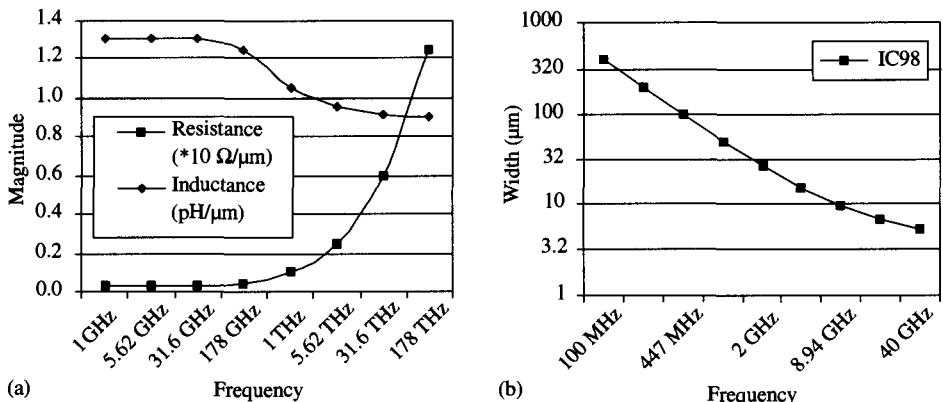
Although technology trends have led to an increase in inductance at a given frequency, when plotted as a function of  $f_{crit}$  the increase in self-inductance is less severe



**Figure 17.3** (a) Critical frequency, loop resistance, and self-inductance plotted for a wide bus above a plane, and (b) the variation in self-inductance and mutual inductance for the IC86 and IC02 generations at 1/20th the critical frequency.

[Fig. 17.3(b)]. This is due to the reduced penetration depth of return currents at higher frequencies, and is sometimes referred to as the proximity effect. The effects of mutual inductance however are of relevance over a greater conductor distance at the more advanced technology node. The impact of these distant conductors can be especially problematic for accurate interconnect modeling and signal integrity management.

As on-chip edge rates continue to decrease we encounter another issue regarding the skin effect, which with increasing frequency (at faster edge rates) results in a higher proportion of the current flow toward the periphery of the conductor rather than flowing uniformly throughout [4]. Although this phenomenon serves to slightly lower the conductor's inductance it does so at the expense of an increase in resistance [Fig. 17.4(a)]. This effect is especially relevant for wide interconnects such as clock nodes, global buses, and supply wires.



**Figure 17.4** (a) Variation in effective inductance and loop resistance, and (b) the conductor width that would be required to provide a dc resistance equivalent to that of the return path, both plotted as a function of frequency for a wide bus above a plane for the IC98 generation.

### 17.2.1.4 Alternative Technology Solutions

Recognizing the progressive increase in resistance and in particular the RC time constant, the semiconductor industry has shifted toward alternative solutions for lowering these parameters without compromising dimensional scaling. Two such technology shifts are toward replacing aluminum metalization with copper, and replacing the insulation between conductors with materials that exhibit a lower permittivity. This introduction is expected between the IC98 and IC02 generations [13].

Pure copper has a resistivity of  $\approx 1.8 \mu\Omega \cdot \text{cm}$ , whereas pure aluminum has a resistivity of  $\approx 2.9 \mu\Omega \cdot \text{cm}$  (at room temperature); therefore, for the same metal dimensions a considerable reduction in resistance can be achieved (or alternatively, for the same resistance a reduction in interconnect thickness, and hence  $C_{lat}$ , can be achieved). However, as already mentioned copper, diffuses into the surrounding dielectric if not contained on all sides by a barrier. It is therefore critical in future technologies to ensure that the thickness of this barrier remains in proportion to the overall conductor dimensions, otherwise the advantages of a lower resistivity will be offset by a reduction in the cross-sectional area of the copper core.

The insulating material between, above, and below each conductor has an implicit dielectric constant that serves to “strengthen” the electric field lines between conductors compared to their field strength in a vacuum. This in turn results in an increase in surface charge and hence an increase in capacitance. Conventional silicon dioxide with a dielectric constant of  $K \approx 4.0$  has been replaced by newer materials such as spin-on glass with  $K \approx 3.0$ , and further materials advances are expected to reduce this to  $K \approx 1.5$  [13]. Issues such as production cost and structural reliability may however be compromised with the inception of these new materials.

### 17.2.2 Faster Edge Rates

The total interconnect capacitance per unit feature size is decreasing with each technology generation, which therefore facilitates progressively faster on-chip switching speeds (assuming constant drive current per W/L of the devices). This phenomenon is further exacerbated by the relative increase in  $C_{lat}/C_{tot}$  such that a system of conductors switching in unison will have a very low effective capacitance (since all mutual capacitances effectively disappear) and a correspondingly faster edge rate. Although faster switching speeds facilitate increased clock frequencies and hence an improvement in overall performance, they pose a significant challenge to the task of the interconnect designer.

Faster edges contain a greater proportion of high-frequency components which in turn exacerbate inductance effects by virtue of the increase in inductive impedance. As mentioned in Sect. 17.2.1.3, if the edge rate contains frequency components that are a significant fraction of the critical frequency then it becomes necessary to model, or at least manage, on-chip inductance effects.

Another effect of high-frequency operation is the localization of return currents, which result from the reduced depth of penetration of the electromagnetic waves into the conducting medium, as illustrated in Fig. 17.4(b). This increases the loop resistance and reduces the distance beyond which mutual inductance effects can be considered to be negligible. This reduced interaction may at first seem advantageous, since a signal can be analyzed with less dependence on the inductive coupling from its distant neighbors, with that distance (and hence the number of influential neighbors) reducing as

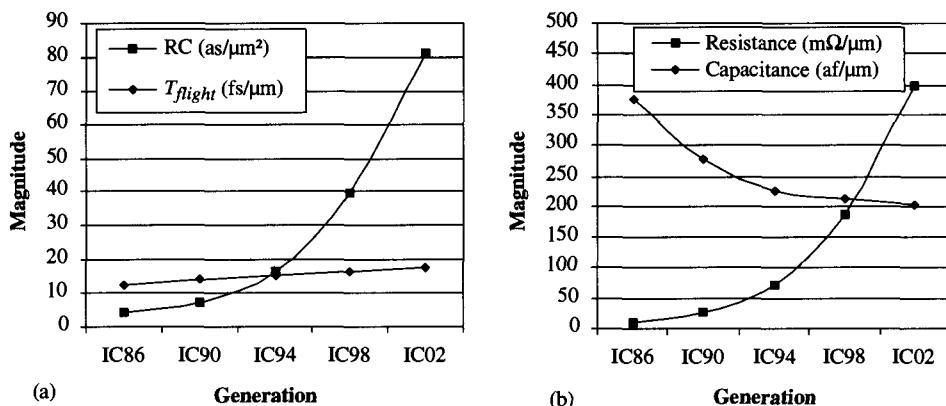
edge rates increase. However, this proximity effect also increases the statistical likelihood of a worst-case (and best-case) inductive coupling scenario, again since fewer conductors contribute to the interaction.

Capacitive crosstalk to a *static* victim node also exhibits a dependence on edge rate due to the resistive isolation between the source transistor (which recovers from the noise) and the location of the noise along the wire. This effect is also a function of device sizing and wire length.

### 17.2.3 Longer Electrical Lengths On-Chip

Although device feature sizes continue to scale down with each process generation, the available die size does not. In fact, the trend for most microprocessors has been to increase the available die size. Although this enables a larger number of devices to be incorporated on-chip it also increases the electrical distances that need to be traversed, when measured in terms of feature size.

For a 1  $\mu\text{m}$  length of interconnect the RC time constant increases with each technology node, as illustrated in Fig. 17.5(a), due to a significant increase in resistance despite a small decrease in capacitance [Fig. 17.5(b)]. Therefore driving the same physical distance in a subsequent technology would incur an increase in RC delay and a greater dispersion of the edge rate. Furthermore the time of flight delay  $T_{\text{flight}} = \sqrt{LC}$  remains relatively constant and hence comprises a greater proportion of signal delay as clock periods and edge rates improve.



**Figure 17.5** (a) RC and  $T_{\text{flight}}$  time constants for a wide bus above a plane and beneath an orthogonally routed layer, and (b) the wire resistance and total capacitance per  $\mu\text{m}$ .

These arguments assume that the same physical distance between logic elements is driven for each technology node, by virtue of the fact that the die size does not decrease. However, in actuality driving shorter distances and inserting repeaters more regularly along the wire alleviates the resulting increase in RC delay. This therefore implies that a comparison of RLC against feature size is more representative of interconnect performance than a comparison of RLC per micron. This process alleviates the majority of problems associated with driving longer electrical lengths; however, the insertion of

repeaters, although improving overall delay, still results in an *increase* in delay relative to the feature size. A more detailed discussion of repeater insertion is presented in Sect. 17.6. Note also that with techniques such as low swing signaling, current sensing, and active regenerators the physical wire length may still be very long.

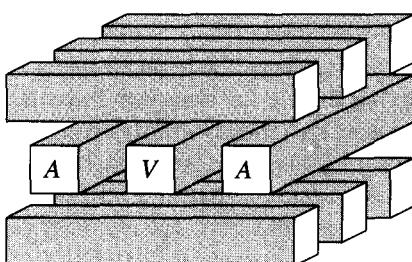
### 17.2.4 Process Variation

Process variation not only refers to perturbations in the device characteristics, it also includes perturbations in the characteristics of the interconnect. In particular, the manufacturing controllability of the CMP process is not scaling in the same degree as the interconnect and ILD thicknesses, and as such there can be significant variations in the nominal RLC parameters. Such variations occur not only between wafers, and between dies on a given wafer, but are also of considerable magnitude across a single chip. Designers must be cognizant of their effect on race, critical paths, and signal integrity to ensure functionality across a given range of device and interconnect variations. Other significant sources of interconnect variations include the etch/deposition processes, lithographic patterning, and material impurities.

### 17.2.5 Architectural Issues

Processor architectures are becoming more complex with each technology generation in the all-consuming goal of achieving higher performance. Multiple functional units, speculative execution, larger array structures, and wider data paths have increased the number of bits of information that need to be transmitted across chip. Furthermore as power dissipation levels increase with a corresponding increase in supply current, there is a need to progressively reduce the resistance and inductance of the supply grid to reduce unwanted  $IR$  and  $L\partial I/\partial t$  voltage drops, and  $I^2R$  power losses. These issues have necessitated an increase in the number of available metal layers with each successive technology node.

Although this certainly aids the interconnect designer by providing greater flexibility, it complicates the management of on-chip signal integrity. Assuming that adjacent layers are routed in orthogonal directions, then parallel wires will be routed on every second metal layer and may capacitively couple to each other if the layer in between has a wide spacing (Fig. 17.6). Also, as already mentioned, the effects of self- and mutual inductance, become more significant at each technology node, so that parallel layers may exhibit significant coupling. Therefore it may no longer be possible to model a signal on a given metal layer in isolation of the switching activity of signals on other layers.



**Figure 17.6** An example of the orthogonal routing of adjacent layers.

Another consequence of technology scaling on processor architecture is the fact that data needs to be transmitted over longer electrical distances. As such, the delay involved in communicating between major sections of logic may in fact exceed the delay of the logic within those sections. Architects need to be aware of the impact of increased communication delays as they engineer more complex architectures for future technology nodes, placing greater emphasis on data locality.

### 17.2.6 Power Dissipation

Historically the power dissipation of successive processor architectures increases despite a corresponding decrease in supply voltage, due to the various architectural issues discussed in Section 17.2.5. To cope with this increase in power a significant effort must be invested in designing the power distribution network (supply grid) and removing heat from the die. As a consequence, power management has become a first-order design constraint for high-performance microprocessors.

It therefore becomes more important for architects and interconnect designers to consider trading off speed for power. This is especially true given that the interconnect can be expected to dissipate a greater proportion of the total chip power with each successive processor architecture. Circuit techniques such as low swing signaling (see Section 17.6.3) and bit encoding [14] have been developed to address this issue at the cost of reduced noise immunity, routing density, and/or delay.

## 17.3 PROBLEMS AND SOLUTIONS REGARDING CAPACITANCE

Capacitance is perhaps the most well-understood phenomenon of on-chip interconnect because it was the first fundamental parameter which required modeling (before resistive and inductive effects became more relevant). It is a measure of the charge differential per volt that exists between two nodes, and is given by the familiar equation  $C = \partial Q / \partial V$ . More specifically, it is a measure of the distribution of charge required to support the divergence of an electric field through a dielectric medium, and can be derived from Coulomb's Law:  $\nabla \cdot D = \rho$  for a given spatial configuration of conductors and dielectric interfaces [4]. The most common manifestation of this equation relates to the capacitance between two parallel plates:  $C = \epsilon A / d$ , where  $A$  is the area of the plates and  $d$  is the distance between them (fringing fields ignored). This equation is often used as a guide to determining the relative magnitudes of fixed and lateral capacitance to the total capacitance of a wire, and can be used to explain the trends of Fig. 17.2.

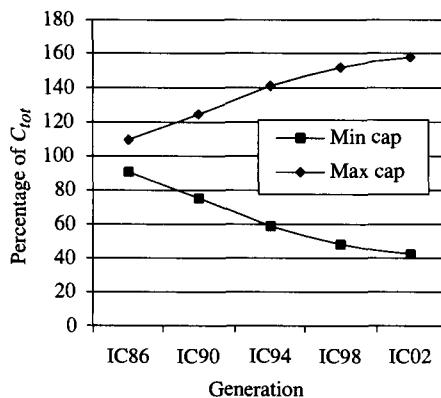
It should be noted that in the following discussion it is assumed that orthogonal layers are routed in a perpendicular direction with minimum spacing as illustrated in Fig. 17.6. Any given conductor (such as  $V$ ) will therefore have a large capacitance to its neighboring wires and a small capacitance to the *many* other wires that route orthogonally above and below it. Since these wires are unlikely to switch at precisely the right time (and in the same direction) to couple to conductor  $V$  along its entire length, these orthogonal layers can be considered static *for initial design purposes*, and the sum of their small capacitances has been lumped into a net capacitance to ground.

### 17.3.1 Power Dissipation

The dynamic switching power required to charge and discharge a capacitance  $C$  is given by the familiar equation  $P_{dyn} = n \cdot f \cdot V_{DD} \cdot V_{swing} \cdot C$ , where  $n$  is the switching activity factor ( $n \leq 1$ ),  $f$  is the switching frequency (usually equal to  $1\times$  or  $0.5\times$  the clock frequency),  $V_{DD}$  is the power supply voltage, and  $V_{swing}$  is the voltage differential between the initial and final voltages across the terminals of the capacitor. Since  $P_{dyn}$  is directly proportional to  $C$  it is important to minimize the interconnect capacitance from the perspective of power dissipation.

As indicated in Fig. 17.5(b) the trend at each technology node is such that  $C_{tot}$  per micron is only slowly decreasing. Furthermore, if one considers that the switching activity of adjacent wires could be in the opposite direction, then a voltage differential of  $2V_{swing}$  across each  $C_{lat}$  needs to be supported. This factor of two can be applied to the lateral capacitance instead of the voltage swing, resulting in a “Miller” capacitance of  $C_{var} = 2C_{lat}$  and an effective capacitance of  $C_{eff} = C_{tot} + C_{var} = C_{fixed} + 2C_{var}$ . Given that the ratio of  $C_{lat}/C_{tot}$  is steadily increasing [Fig. 17.2(b)] this Miller capacitance dominates and can significantly increase the worst-case power dissipation (and corresponding power supply requirements) of a bus.

It should also be noted that if adjacent wires were to switch in the same direction then the Miller capacitance is subtracted from the total, resulting in  $C_{eff} = C_{tot} - C_{var} = C_{fixed}$ . This results in the minimum power requirement to drive the bus and, depending on the ratio of  $C_{lat}/C_{tot}$ , can be considerably lower than the worst-case power requirement. Figure 17.7 plots the extremes of capacitance (and hence dynamic switching power) for each technology node relative to total capacitance, and highlights the increasing dependence on switching activity.



**Figure 17.7** Minimum and maximum effective switching capacitances relative to total capacitance.

### 17.3.2 Delay Variations

For a lumped RC network the time taken for the capacitor voltage to reach 50% of its final value in response to a step input is  $T \approx 0.7RC$ , which is proportional to capacitance. A similar proportionality is achieved for distributed RC models and linear-ramp inputs; therefore, the arguments of the previous section regarding the best- and worst-case power dissipation also apply to the best- and worst-case signal delays.

It is evident from Fig. 17.7 that at each technology node the variation in delay is increasing due to the increasing effect of the Miller capacitance (i.e., the increasing ratio of  $C_{lat}/C_{tot}$ ). When designing for critical path delays one must size devices for driving the worst-case capacitance in the required time, and when analyzing fast paths one must consider the resulting delay under best-case capacitance conditions. The growing disparity between these two extremes is a significant concern for race analyses.

### 17.3.3 Crosstalk Noise

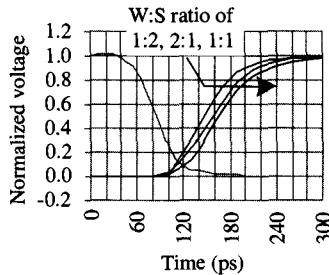
Consider the situation in which the victim wire  $V$  of Fig. 17.6 is floating and both of the aggressor wires  $A$  switch low. Through simple voltage division across the capacitances it can be shown that the victim wire  $V$  will suffer a voltage drop of  $\Delta V = 2C_{lat}/C_{tot}$  per volt of switching. A static victim node will suffer a similar but reduced voltage dip, since the drive transistor turns on to replenish the lost charge. Such crosstalk noise can be catastrophic in two regards. First, if the voltage change on the victim wire were to cause it to rise above  $V_{DD}$  (or below  $V_{SS}$ ) then hot carrier injection and time-dependent dielectric breakdown could occur, reducing the life of the chip. Second, and perhaps more importantly, if the voltage change on the victim wire were to cause it to dip below  $V_{DD}$  (or above  $V_{SS}$ ) then subsequent logic gates could turn on causing glitches and power on static nodes (and hence a potential reduction in clock frequency), irrecoverable charge depletion on dynamic nodes, and irrecoverable inversion of data stored on flip-flops. It is of *vital* importance therefore to manage the problem of capacitive crosstalk, as well as inductive crosstalk (discussed in Sect. 17.4.2).

### 17.3.4 Potential Solutions

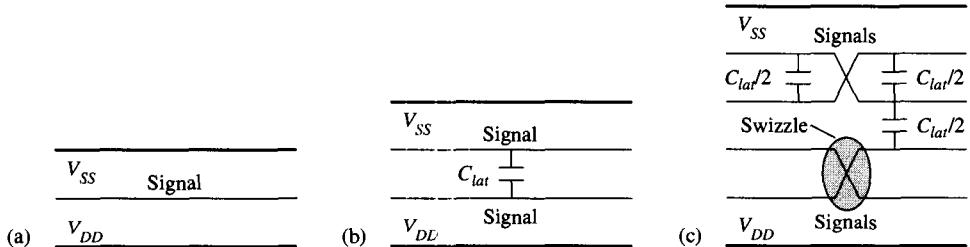
We have seen that the peak capacitance variation of a wire (given by  $C_{var} = 2C_{lat}$ ) is a concern for designing critical paths, generating race conditions, and managing power dissipation. As the ratio of  $C_{lat}/C_{tot}$  steadily increases, these concerns become more dramatic and presently demand a significant fraction of the electrical verification time. Clearly then, considerable benefits to the design of interconnect structures could be achieved by employing techniques that reduce this variation.

The most obvious solution is to increase the separation between the wires to reduce the lateral capacitance and hence reduce  $C_{var}$ . Since  $C \propto 1/d$  (to a first order) significant gains can be made in reducing lateral capacitance by just a moderate ( $1.5 \times$  to  $2 \times$ ) increase in spacing. Similar benefits can also be achieved by increasing the width of the victim wire, although this is a power-hungry technique since the total capacitance is increased—however, it is often a necessary evil since widening the wire also reduces resistance, and hence the RC time constant (within reason). Note that both of these approaches also cost in terms of routing density. Figure 17.8 illustrates the improvement in maximum delay for when the width/spacing is increased by  $2 \times$ , for a fixed driver size in the IC98 technology. Due to the large magnitude of  $C_{var}$ , a better improvement in worst-case RC delay has been achieved by increasing the spacing rather than the width.

Another obvious solution is to place a supply rail between each wire as shown in Fig. 17.9(a). This reduces  $C_{var}$  to zero and fixes the maximum capacitance at  $C_{tot}$  regardless of switching activity. Although this is an excellent solution it is of course expensive in terms of routing density. Note however that some fraction of metal would have been required anyway for chip-wide supply distribution, which offsets some of this routing cost.



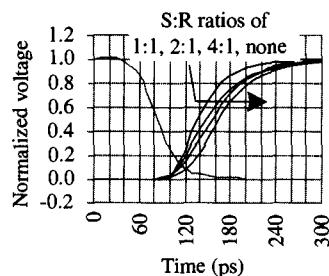
**Figure 17.8** Examples of the improvement in delay when employing various width:spacing ratios.



**Figure 17.9** Shielding signals with (a) a 1:1 SR ratio, (b) a 2:1 SR ratio, and (c) a 4:1 SR ratio with swizzling.

To enable greater routing density the implementation of Fig. 17.9(b) could be employed, which instead shields every *pair* of conductors and is termed a 2:1 signal to reference (SR) ratio. This provides for a nominal 67% routing density while still halving  $C_{var}$  from  $2C_{lat}$  (with no shields) to just  $C_{lat}$ . Note that increasing the SR ratio beyond 2:1 is of no benefit in reducing  $C_{var}$  for the central victim wire(s) unless swizzling is performed. This technique is illustrated in Fig. 17.9(c) for a 4:1 SR ratio, in which all wires now experience a 25% reduction in capacitance variation ( $C_{var} = 1.5C_{lat}$ ). The nominal routing density has also been increased to 80%. A slight implementation cost of this latter technique is that a single routing track on an orthogonal layer is now required to effect the swizzle. Extending this technique to larger SR ratios provides diminishing returns in reducing  $C_{var}$  and enabling greater routing density (being mindful of the needs of supply distribution), and also requires more frequent swizzling. Figure 17.10 illustrates the improvements in maximum delay that can be achieved with these techniques.

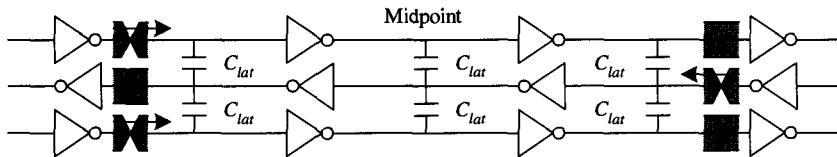
Another solution is to offset the switching times of adjacent wires such that the relative voltage transition between the wires, and hence the effective Miller capacitance,



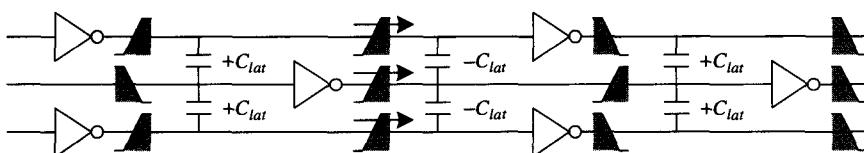
**Figure 17.10** Examples of the improvement in delay when employing swizzling with various signal:reference ratios.

is reduced. To a first-order,  $C_{var}$  will decrease by a factor of  $T_{offset}/T_{edge\ rate}$  by using this technique; however, staggering the transition times may shorten the amount of time left available in a clock cycle in which useful work can be done. The most common manifestation of this technique is to switch adjacent wires on alternate clock phases, which results in  $C_{var} = 0$ .

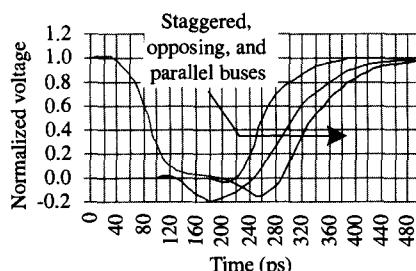
Another very useful technique for long, repeated buses is to interleave signals that travel in opposite directions, as shown in Fig. 17.11. Each signal only incurs a capacitance variation of  $\leq 2C_{lat}$  (depending on edge rates) when the two signals pass by each other, typically near the midpoint of the buses. Assuming further that inverters are employed in repeating the signal over a long distance, then a preferential solution may be to offset the inverter locations along the bus, as shown in Fig. 17.12. Every wire is subject to a variation of  $+C_{var}$  along half of its length and  $-C_{var}$  along the other half, such that to a first order the capacitance variations cancel out, resulting in an average  $C_{var} = 0$ . In practice, however, the nonzero delay of the inverter results in a time offset and hence a small increase in  $C_{var}$  along one half of the wire. The cost of this technique is that the number of repeater locations is doubled and the optimal repeater location (see Section 17.6.2) for at least one half of the wires may be compromised. Figure 17.13 illustrates the improvements in maximum delay that can be achieved for a three-inverter repeated bus when employing these techniques. Note that the staggered technique also exhibits a significantly smaller undershoot.



**Figure 17.11** Reducing capacitance variations for repeated buses by interleaving signals traveling in opposite directions.

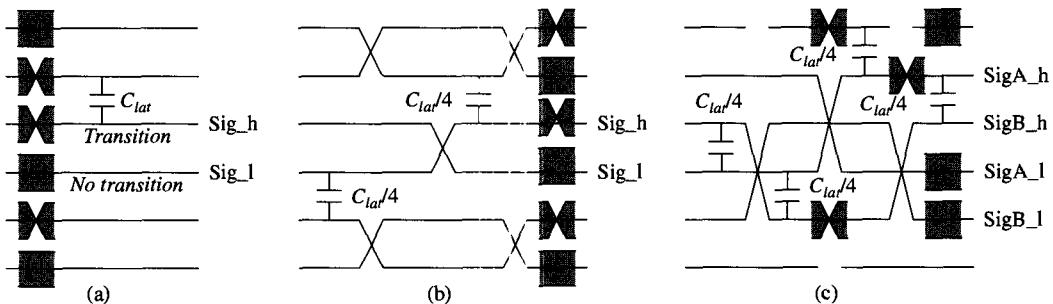


**Figure 17.12** Reducing capacitance variations for repeated buses by staggering inverter locations.



**Figure 17.13** Examples of the improvement in delay when routing buses with co-located (parallel) repeaters, staggered repeaters, and interleaving opposite directions of transmission.

The techniques proposed thus far have assumed single-rail data and no constraints on switching activity; however dual-rail signaling is often employed, for example, for the bit lines of a RAM array or with certain logic families such as DCVSL. Under these conditions it is known that exactly one wire from each bit pair will transition during the activation phase, as indicated in Fig. 17.14(a), and it is therefore possible to construct coupling minimization techniques specific to these structures. Note that for such a signaling scheme  $C_{max} = C_{tot}$  and  $C_{min} = C_{tot} - C_{lat}$ , since at most one adjacent wire can transition with the victim.



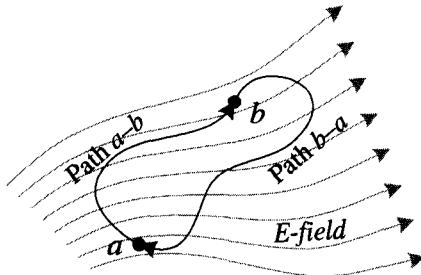
**Figure 17.14** Reducing capacitance variations with dual-rail signals by routing  
(a) without swizzling, (b) with swizzling between adjacent bit pairs every quadrant, and (c) with reordering and swizzling of the bit pairs.

The swizzling techniques that apply to the single-rail data wires discussed earlier can be modified slightly to remove *all* capacitance variation without the need for shield wires, as illustrated in Fig. 17.14(b). Every wire routes next to a switching aggressor for one half of its length and next to its stationary dual-rail partner for the other half, resulting in *no* capacitance variation as well as a net reduction in maximum capacitance from  $C_{tot}$  to  $C_{tot} - 0.5C_{lat}$ . A more advanced swizzling technique for dual-rail signals is shown in Fig. 17.14(c), which consists of *two* bit pairs as the fundamental array unit. For the same number of swizzles the capacitance variation is again zero but the maximum capacitance has been further reduced to just  $C_{tot} - C_{lat}$ . Clearly, for power reasons as well as for differential development on low swing signals, this form of swizzling is especially advantageous.

## 17.4 PROBLEMS AND SOLUTIONS REGARDING INDUCTANCE

Section 17.2 provided some insight into why the issue of on-chip inductance has become an increasing concern in modern technologies, but just what is inductance and how does it impact interconnect design? To address this question one must first define the concept of a voltage between two points in the presence of an electric field:  $V = \int_a^b E \cdot \partial l$ , as illustrated for two arbitrary paths of integration in Fig. 17.15 (path *a*–*b* might represent the current *injected* into a conductor and path *b*–*a* the current in the *return* path back to the source).

If the electric field is time invariant then the path of the integral is irrelevant, and the voltage between two points in space can be uniquely defined. This fact can also be expressed by saying that the voltage around a closed loop is zero:  $\oint E \cdot \partial l = 0$ , which



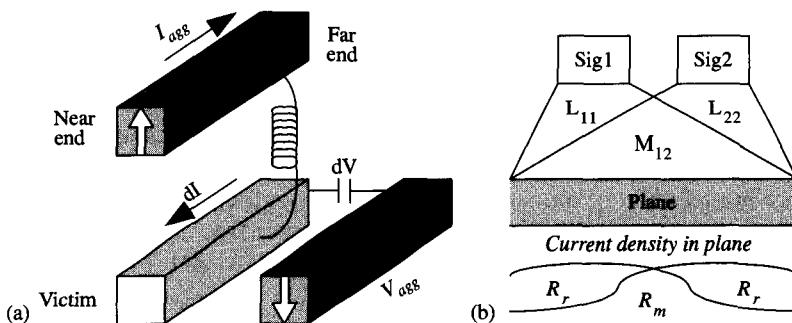
**Figure 17.15** Method of determining the voltage between two points in the presence of an electric field, for two different path integrals.

forms the basis of Kirchhoff's Voltage Law (KVL). If one now considers the following term in Maxwell's equations,  $\nabla \times E = -\mu \partial H / \partial t$ , then the presence of a time-varying magnetic field will result in a nonzero curl of the electric field around a given loop, such that  $\oint E \cdot d\ell = V_L \neq 0$ . Such a magnetic field could be produced by a time-varying current density  $J$  according to the following term in Maxwell's equations:  $\nabla \times H = \epsilon \partial E / \partial t + J$ .

Under these conditions the voltage between two points *cannot be defined* unless the path of the injected and return currents are known (for which the determination of the latter is often a difficult task), and even then KVL cannot be applied per se (for the purpose of circuit analysis) since the voltage around the closed loop is nonzero. We therefore define a fictitious element called an "inductor" whose voltage  $V_L$  is proportional to the time variation of the injected current,  $V_L \propto \partial I / \partial t = L \partial I / \partial t$ , that constant of proportionality being the *inductance* of the current loop(s). To minimize the voltage across this inductor the return current path has to be as near as possible to the injected current path (resulting in a small loop area through which the magnetic flux can flow), and/or the rate of current injection must be kept small.

Although most texts discuss inductance in the context of loss-less transmission lines no such treatment is given here; however, excellent analyses of the subject can be found in [4]–[6]. Instead, this section focuses on the impact of inductance on-chip, in which firstly even the least resistive conductors are extremely lossy so that a loss-less transmission line model is wholly inadequate, and secondly the impact of mutual coupling (or crosstalk) between signals is so severe that analyses dealing with a conductor loop in isolation are also inadequate.

In regards to crosstalk, when a negative voltage is applied to an aggressor wire as shown in Fig. 17.16(a), it couples to a victim wire through capacitance and results in a



**Figure 17.16** (a) Illustration of capacitive and inductive crosstalk, and (b) self- and mutual inductance and return path resistance.

voltage dip in that wire across its entire length, with its greatest magnitude being at the wire end which is furthest from any turned on drive transistors (due to the separation of these nodes through the wire's resistance).

Similarly, when a current is injected into an aggressor wire [also shown in Fig. 17.16(a)] it couples to a victim wire through inductance and results in a current change in that wire in the direction of the return path (toward the source). If the victim wire has a large resistance to the supply network (as is the case for CMOS devices, even when turned on) then the displaced charge effectively has nowhere to flow and results in a charge depletion (a voltage dip) at the far end of the victim wire and a charge accumulation (a voltage rise) at the near end. Continued charge displacement is eventually resisted by the potential difference generated across the length of the wire. Therefore only a small fraction of the required return current can typically be sustained by a victim wire, the remainder of which must be sustained by other, more distant victims. Through this method of reasoning it is easy to see why inductive coupling extends across great distances in the absence of sufficiently nearby, low-resistance supply rails (which facilitate the continual flow of charge while ideally developing only small potential differences).

Accurately modeling such systems therefore requires a detailed knowledge of the switching activity (direction of current injection) of many wires on the same layer *as well as* on any other parallel routing layers, since the separation between layers is relatively small for inductive purposes. As such, the influence of inductance is of great concern to modern interconnect designers because an accurate model of such systems involves many more conductors than a purely capacitive system. Note also that to a high degree of accuracy, signals which route orthogonally to an aggressor wire are unable to support a charge displacement across their small width, and can usually be ignored when considering inductance.

Because inductance is a function of the distribution of return currents, and the wires that support this return current on-chip are lossy, the resistance of the return path will also be a function of the distribution of return currents. In other words, the inductance and resistance parameters of a network are interrelated through the return path. Furthermore, because these return paths are shared by many signals, any intersecting current loops between them result in *mutual inductances*, which for dispersive returns may be of similar magnitude to the self-inductances. Another effect of the shared return paths is that due to their nonzero resistivity, the return current flow in one loop results in a *mutual resistance* with other signals whose return currents share the same path. This phenomenon is crudely illustrated for a two-conductor system above a lossy plane in Fig. 17.16(b). The task of interconnect modeling is therefore further complicated since the effective resistance of a signal, which must include its return path, is now dependent on the switching activity of other, potentially distant wires.

Historically the edge rate of on-chip signals has been small enough that the reactance of the inductive loop is small compared to the resistance of the loop even for highly dispersive return current distributions (with large inductance). A simple resistance model without inductance was therefore sufficient. Furthermore since the return currents were extremely dispersive their resistance was also negligible compared to the resistance of the signal wire, and could be ignored. Also, because of the wide dispersion of returns at these low frequencies, the effects of mutual inductance and resistance could be statistically "averaged out," since the likelihood of all signals switching at the *same time* and in the *same direction* (termed "even mode" switching) across these vast distances was small. In essence, an "odd mode" switching scenario, in which

adjacent wires switch in opposite directions, could be assumed, and hence the mutual inductance and resistance terms, when summed across all influential signals, cancel out.

As edge rates have increased to the point at which  $\omega L$  approaches  $R$  there has evolved a greater need for modeling inductance on-chip. Note that the impedance of an inductive loop is given by  $\omega L$  for *each* frequency component of the edge, which is usually bound to a fixed value through some heuristic based on the dominant frequency component ( $f_{dom} = 1/\pi T_{edge\ rate}$  is often employed). At these higher frequencies the distribution of return currents becomes less dispersive (the proximity effect) and reduces the inductance of the current loops. Note that the same phenomenon also occurs with the distribution of current *within* a given conductor, which condenses near the conductor surface at high frequencies and also reduces inductance (the skin effect). This condensation of return currents brings into question the assumption that the mutual inductance and resistance effects can be averaged out, since fewer conductors now share the same return paths and the likelihood of all influential signals exhibiting even mode switching increases, especially when this distribution condenses to within the width of a typical data bus. These mutual terms therefore become an increased concern when attempting to accurately model interconnect at higher frequencies.

### 17.4.1 Delay Variations

Crudely speaking, the incorporation of inductance introduces two new time constants in addition to the fundamental RC time constant, which itself is increased by virtue of the additional resistance of the return path. The L/R time constant is analogous to the RC time constant and results in an exponential increase in the injected current rather than an instantaneous current surge as predicted under an RC model. In practice this is already bounded by the saturation current of the drive transistor, and for on-chip interconnects is typically 1–2 orders of magnitude smaller than the RC time constant and can therefore be ignored. Note also that this time constant is length independent, whereas the RC time constant scales as the square of length. Of greater concern is the  $\sqrt{LC}$  time constant, also referred to as the time of flight ( $T_{flight}$ ), which results from the finite phase velocity of electromagnetic waves in the dielectric medium and scales with length. For on-chip interconnects this time constant may be a significant fraction of the RC time constant and cannot be ignored. It should be noted here that for lossy conductors with  $f \ll f_{crit}$  (or equivalently,  $R \gg \omega L$ ) the phase velocity is actually given by  $v \approx \sqrt{2\omega/RC}$  and is frequency dependent (dispersive) with a time constant *greater* than  $\sqrt{LC}$ ; however, these issues are beyond the scope of this text and the reader is referred to [4]–[6] for more details.

The dependence of each of these time constants on length is illustrated in Table 17.1 for the IC98 generation (wide bus above a plane, beneath an orthogonally routed layer,

**TABLE 17.1** Time Constants (in ps) for a Wide Bus above a Plane and beneath an Orthogonally Routed Layer, for the IC98 Generation

Time Constant (in ps)	1 cm	1 mm	100 $\mu$ m	Increase when Gate Gap = Wire Cap
$RC$	2900	29.0	0.3	$\times 2.0$
$L/R$	4.6	4.6	4.6	$\times 1.0$
$\sqrt{LC}$	115	11.5	1.2	$\times 1.4$
Dominated by	$RC$	$RC, \sqrt{LC}$	edge rate	

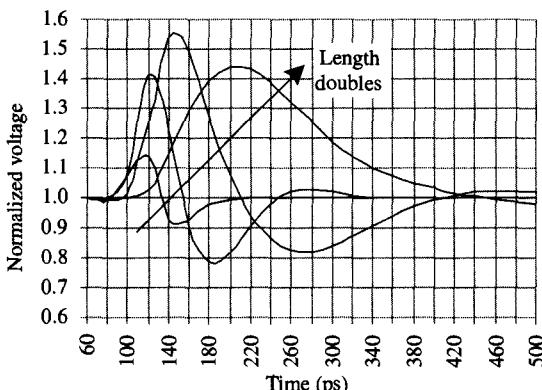
with even mode switching). For very long wire lengths the RC time constant dominates and is sufficient for interconnect analyses. At intermediate wire lengths typical of many on-chip buses the time of flight delay becomes a significant fraction of the total delay, and it is in this regime that the effects of inductance are of concern from a signal delay perspective. At short wire lengths all time constants are swamped by even the fastest edge rates of the drive transistor. It should also be noted that inductive coupling from aggressor wires on other layers could serve to further increase the time of flight delay of a victim wire, and that the RC and  $\sqrt{LC}$  time constants increase in the presence of distributed gate load.

Although on-chip inductance results in a modest increase in delay due to the time of flight effect, it also results in improved slew rates (and hence provides a delay reduction) by virtue of the reflection coefficient. On-chip interconnects do not possess a load impedance ( $Z_L = -j/\omega C_L$ ) that matches the characteristic impedance of the transmission medium ( $Z_0 \approx \sqrt{R/2\omega C}(1-j)$ , for a lossy conductor with  $f \ll f_{crit}$  or equivalently,  $R \gg \omega L$ ), and, as such, some fraction of the incoming wave is reflected from the far end of the wire as given by  $\rho_v = (Z_L - Z_0)/(Z_L + Z_0)$ , where  $\rho_v$  represents the relative magnitude and sign of the reflected voltage. For most capacitive loads (when  $C_L \ll \sqrt{C/\pi f R}$ ) this coefficient has a magnitude close to one and a phase angle close to zero. As such, the incident wave is reflected with equal amplitude at the far end of the wire, which results in an improvement in slew rate and delay. As a disadvantage, it could also result in the far-end voltage overshooting the supply rail (for a positive voltage transition) creating additional reliability concerns; however, the nonzero edge rate and attenuation factor of the interconnect serve to reduce the magnitude of this overshoot.

### 17.4.2 Crosstalk Noise

Assume now that the victim wire is initially at a static voltage rather than in transition as assumed in the preceding discussion. In addition to the capacitively coupled noise on this victim wire from its adjacent aggressors, there will also be an inductively coupled voltage change as charge is displaced along its length to support some fraction of the return current. The net result is that the noise levels predicted under an RC modeling regime will *underestimate* the noise levels which could actually occur on chip. This is arguably the most important issue to manage regarding signal integrity since if noise margins are unexpectedly violated then functional errors will occur that may not be recoverable by extending the clock frequency.

The magnitude of the inductively coupled noise is strongly dependent on the distribution of capacitive gate loading and the length of the interconnect. An increase in distributed gate loading results in an increase in the time of flight delay and hence, the period of time during which a return current must be sustained. This results in an increase in charge displacement and corresponding noise levels. Similar arguments apply to when the length of the interconnect is increased; however, this also results in an increase in resistance and hence, attenuation. Beyond a certain length the resistance of the wire dominates and the inductively coupled noise levels slowly decrease. Figure 17.17 illustrates the trend of the inductively coupled noise as the length of a line is increased. A similar trend is seen with distributed gate loading, although extremely large loads are required to reach the turning point in the trend.



**Figure 17.17** An example of the variation in inductively coupled noise with length.

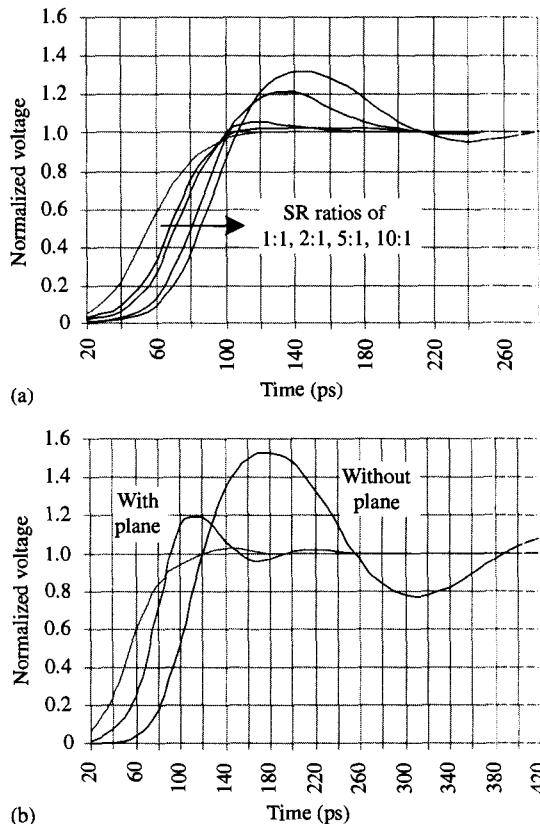
### 17.4.3 Potential Solutions

Whereas one of the most obvious solutions to overcoming capacitive coupling was to increase the spacing between conductors, it is in fact slightly detrimental to inductive coupling since the loop and mutual inductances of the system increase. The converse solution, to reduce the spacing between conductors, is clearly infeasible since minimum spacing is already assumed and capacitive coupling would then be exacerbated.

The obvious solution then is to introduce supply rails into the system that are explicitly used to conduct large portions of the return current. This technique is of benefit to capacitive coupling and is also beneficial for inductive coupling. Although an SR ratio beyond 2:1 is of no benefit to capacitive coupling unless swizzling is introduced, the same is not true for inductive coupling due its wide coupling reach. A 4:1 SR ratio will in fact provide better crosstalk management than an 8:1 ratio, for example. Figure 17.18(a) illustrates the far-end overshoots for a bus routed with varying SR ratios. Signals on either side of these supply rails will also have good inductive isolation from each other; however, if the SR ratio is too large then the effects of conductors on other parallel routing layers may still be problematic.

A similar solution is to provide explicit reference planes between orthogonal routing layers which creates a kind of on-chip stripline. This guarantees a low inductance since all signals are adjacent to a low-resistance return path. Good inductive isolation between signals on either side of the plane is also guaranteed. However, the coupling between signals on the same layer may still be significant due to the dispersion of return currents in the plane. Figure 17.18(b) illustrates the far-end overshoots for when a bus is routed above a plane. Other advantages of the plane are that it provides for a very low resistance supply grid (with low  $IR$  drop and  $I^2R$  power losses) and is able to conduct return currents for all orthogonalities.

The impact of inductance is especially problematic when all aggressor wires (potentially on multiple metal layers) exhibit even mode switching. One solution therefore is to use a signaling scheme in which complementary odd mode switching is guaranteed. One example of such a system is in predecoding address bits for a RAM array. At most only one of  $N$  bits can be high at any one time, and after each clock edge there will either be no change in state or else one wire will discharge while another wire charges. This demands opposing current flows and hence a net current injection of zero. Using dual-rail complementary logic (in which both logic states are always transmitted) will



**Figure 17.18** (a) Effect of various SR ratios on far-end overshoot, and (b) when in the presence of an underlying plane.

also produce the same effect, but at the cost of a significant increase in routing density and power dissipation. A compromise is to use dual-rail differential logic (in which only one of two wires is guaranteed to transition) which halves the maximum possible current injection into a system, and hence the delay and noise concerns.

In similar vein, the technique of staggering repeater locations for reducing capacitive coupling effects (as shown in Fig. 17.12) is also of benefit in reducing inductive coupling. Any given wire will be coupled by a current injection from its neighbor for one half of its length and a current ejection for the other half. To a first order these effects cancel out, leaving only every second aggressor to couple to the victim wire.

The technique of staggering transition times (and in particular driving adjacent signals in alternate phases) as used to reduce capacitive coupling effects is also of benefit in reducing inductive coupling, since the current injection waveforms are now offset so that the peak cumulative current is reduced. Similar arguments also apply to the technique of interleaving signals between repeated buses that travel in opposite directions.

## 17.5 PROBLEMS AND SOLUTIONS REGARDING RESISTANCE

The preceding sections of this chapter have, as part of their analyses on capacitance and especially inductance, covered many of the important interconnect concerns regarding

resistance. Some of these issues include its contribution to RC delay, the effects of cladding, the utilization of low resistivity materials such as copper, the additional resistance of the return current path, and the impact of mutual resistance.

Another important issue that relates to interconnect resistance is the phenomenon of electromigration. When electrons are accelerated under the influence of an applied electric field they periodically collide with ions in the metal lattice. This causes the current density to reach a “critical velocity” resulting in  $J = \sigma E$  (Ohm’s Law). These collisions occasionally result in the physical dislocation of ions from the lattice, which under high enough current densities can produce a depletion of metal (a hole) in one location and an accumulation of metal (a hill) elsewhere. As this hole is formed, its cross-sectional area decreases which requires a greater current density in the remaining metal to support the same current flow. This larger current density displaces even more ions from the hole until eventually, a catastrophic open circuit results. Conversely, the hill formed elsewhere in the metal may result in a catastrophic short circuit as it displaces and cracks open the surrounding dielectric.

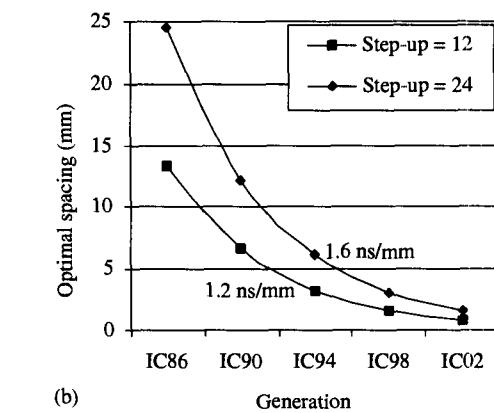
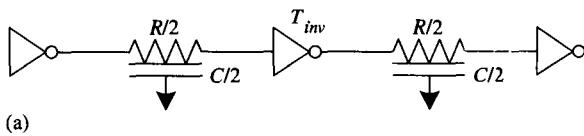
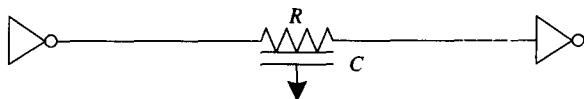
It is therefore important that limitations are set on the allowable current densities to prevent electromigration from occurring. These limits are usually set according to average and RMS currents, and for wire segments which contain unidirectional or bidirectional currents. This latter situation is less problematic since a removal of ions in one direction of current flow is then followed by a replenishment from the reverse current flow.

## 17.6 PROBLEMS AND SOLUTIONS REGARDING LONG DISTANCE ROUTING

As was shown in Table 17.1 the response of a long wire is dominated by the RC time constant which varies as the square of length. When data needs to be transmitted across very long distances this time constant may be unacceptably large, requiring multiple clock cycles to pipe the data. These long latencies can degrade processor performance and it is therefore necessary to investigate methods by which this fundamental RC delay problem can be overcome.

### 17.6.1 Repeater Insertion

Figure 17.19(a) shows a length of interconnect before and after an inverting repeater is inserted along its length, resulting in total delays of  $kRC$  and  $T_{inv} + kRC/2$ , respectively (the constant  $k$  is dependent on the interconnect and driver models). As such, a gain in signal delay can be achieved by inserting repeaters along the length of the wire whenever  $2T_{inv} < kRC$ . Under these conditions the interconnect delay comprises 50% of the total path delay *regardless of the technology*. Therefore the poor scaling of the interconnect delay (relative to the scaling of device delay) does not manifest as an increase in the *proportion* of interconnect delay on chip, but rather as an increase in the *total* path delay by virtue of the increased repeater frequency and shorter routing segments. This latter effect may in fact be beneficial from the perspective of crosstalk management. Note that the problem of interconnect scaling does in fact manifest as an increase in delay for short, local routes which may become RC dominated after scaling.



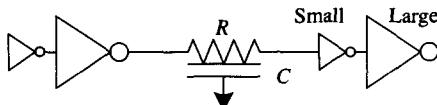
**Figure 17.19** (a) Illustration of repeater insertion, and (b) the optimal repeater spacing for two different step-up ratios.

This first-order analysis has illustrated how inserting repeaters along the length of a wire can reduce overall delay; however, various effects such as the repeater's gate capacitance and current drive capabilities, driver and repeater diffusion and overlap capacitances, and the inverter delay's dependence on input edge rate all serve to alter the conditions and locations under which repeaters should be inserted [7], [8]. Note also that a wire will often have distributed gate loads which add to the RC delay of the path and can very easily become a significant fraction of the total capacitance. Buffering off this load through smaller, stepped-up inverters (with reduced gate capacitance) is a useful technique that can be used to alleviate this problem.

It is of interest to investigate the technology trends associated with repeater insertion, particularly regarding the optimal repeater spacing (and hence the total number of repeaters) and the effects of the step-up ratio (defined as the ratio of  $N$ -device driver width to the equivalent device width of the interconnect). Figure 17.19(b) plots these trends for two different step-up ratios using device technologies indicative of the IC86 through IC98 interconnect technologies (scaled to IC02). The optimal repeater spacing is seen to decrease with each technology node due to the many scaling problems discussed in Section 17.1, which illustrates why the cross-chip transmission delay requires increasing clock cycles. It should also be noted that the optimal spacing often cannot be achieved due to the presence of critical processing logic and underlying signal routes, which therefore further increases the signal delay. It is also evident from Fig. 17.19(b) that by decreasing the step-up ratio (within reason) an improvement in overall delay can be achieved; however, this is offset by an increase in power dissipation and repeater frequency, and may not be worth the trade-off except for certain critical paths.

### 17.6.2 Buffer Insertion

If an odd number of inverters are required for optimal data transmission, or if the bus is tapped at frequent locations, then an additional inverter may be necessary to recover the original logic state. In an effort to overcome this problem, and to also reduce the contribution of the repeater's gate capacitance to the total RC delay, a buffer consisting of two inverters (with increasing size) can be employed as shown in Fig. 17.20 [15].



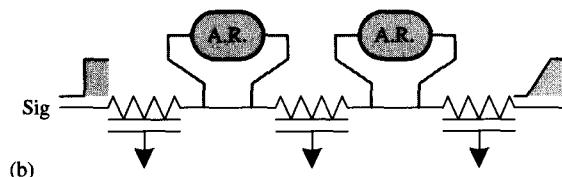
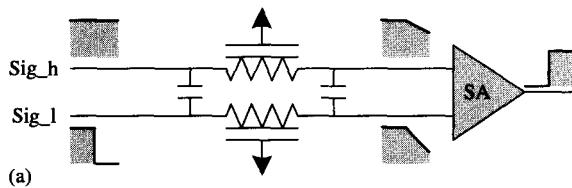
**Figure 17.20** A buffer element consisting of a small inverter load and a large inverter driver.

Due to the reduced size of the first inverter and the increased buffer delay, the optimal spacing between buffers is greater than the optimal spacing between repeaters. This results in a smaller number of buffers for driving the same distance and hence a reduction in the intrusion of this logic into underlying signal routes and processing logic. The cost associated with having to position the buffers away from their optimal locations is also reduced. The trade-off in the increased delay through the buffer network and the longer wire lengths may, however, offset the advantages of this technique for many applications.

### 17.6.3 Low Swing Signaling

In an effort to reduce the intrusion of repeater logic, reduce power dissipation, and potentially drive longer distances, low swing signaling may be employed. This technique is illustrated in Fig. 17.21(a) in which two wires are used to transmit the data differentially across the entire length of the bus (as used in an SRAM). A small voltage differential is developed across the distant terminals and is detected by a sense amplifier (SA).

This kind of technique is also often used for long buses with multiple drivers. Implementing such a bus with repeaters (or buffers) may require their locations to be



**Figure 17.21** The general concepts of (a) low swing signaling, and (b) the attachment of active regenerators.

fixed at each driver as well as being modified to multiplex the data from either the source driver or the rest of the bus, which can significantly increase delay. Note also that the differential bus enables bidirectional transmission of the data to multiple receivers.

The implementation of this technique comes with significant cost. The differential signaling requires two routing tracks instead of one, the low voltage swing developed at the receiver is highly sensitive to noise injection (even excepting common mode components) and requires extensive design management, and the technique enables data transmission in just one phase of the clock (the other being needed for the pre-charge phase). Although another phase of data transmission could follow, the intermediate SA adds delay, logic intrusion, and may still be subject to full clock skew and jitter variations which subtracts from the time available to develop the differential.

Techniques to enable greater routing density through single-ended low swing signaling have been proposed [9], [10]. These techniques effectively halve the differential available through dual-rail signaling and are especially error-prone to noise, due either to reduced supply voltages or the removal of common-mode components. Various charge-sharing techniques have also been proposed to effect the low swing signaling and hence reduce power [11], [12]. These techniques are only valid for very short buses for which the resistance of the interconnect is negligible; otherwise, at the high frequencies of modern microprocessors the differential developed at the point of charge sharing cannot extend to the distant SA.

#### 17.6.4 Active Regenerators

Another technique for improving transmission delay is illustrated in Fig. 17.21(b), in which a number of regenerator stations are dispersed across the length of the bus. These regenerators sense when a voltage transition is occurring and provide an additional current boost to speed up the transition [16]. One advantage of this technique is that the regenerators can be located almost anywhere along the bus and are therefore less constrained by underlying signal routes and processing logic. Furthermore, both bidirectional and multisource behavior of a wire are directly supported by this technique, and differential signaling is not required.

The disadvantage of this technique is primarily in its power dissipation, since the regenerators have a relatively slow response time and significant crossover current. The slow response time is a design requirement of the technique to prevent the amplification of unwanted noise injected onto the bus.

#### 17.6.5 Current Sensing

An alternative to the differential voltage sensing referred to in Section 17.6.3 is to instead sense the direction of current on these wires [17]. An advantage of this technique is that very long interconnect lengths could be driven since the L/R time constant is now of relevance rather than the RC time constant, for which the former is length independent and very small (see Table 17.1). This technique is therefore primarily limited by the signal's time of flight and device switching speeds. Disadvantages include high power dissipation due to the large potential difference across the wire which is sustained for the entire sensing phase, and noise immunity concerns akin to those of Section 17.3, especially regarding the inductively coupled return currents that extend across many conductors.

### 17.6.6 Wave Pipelining

This rather bold technique essentially consists of removing all intermediate latches between the source and receiver, resulting in a propagation delay which is larger than a clock cycle. Therefore at any given time there will be multiple data signals in flight along the bus, nominally separated in time by one clock cycle. The advantage of this technique is that it replaces the intermediate latch with a repeater, thereby reducing overall delay and clock loading.

There are some fundamental disadvantages with this technique. First, there is the concern that a subsequent wave front will, through capacitive and inductive crosstalk, propagate much faster than its predecessor and hence corrupt the earlier transmitted data. Furthermore, at a nominal clock frequency there will be at least a two-cycle delay between the generation and reception of data; however, when the clock is slowed down below a given frequency during chip debug, this will then become a single-cycle delay. Without additional control logic to conditionally pipe the data at the receiver, this phenomenon will result in functional errors.

## 17.7 CONCLUSION

The challenge of designing interconnect structures for processors in the early twenty-first century is indeed daunting. If the historic trends continue we will soon find ourselves in the unenviable situation in which communication delays far exceed the processing delays of functional units. As edge rates continue to improve we find ourselves in another unenviable situation in which the lumped RC modeling of the past becomes inaccurate, and complicated RLC(f) matrices and lossy transmission line models become necessary.

It is conceivable that future signal wires will need to be encased in a kind of on-chip coaxial cable (enveloped in planes with a 1:1 SR ratio) to reduce crosstalk effects and to simplify, or even make feasible, the analysis procedure. This of course would demand many more metal layers than are currently foreseen by the SIA roadmap [13]. As an alternative to such a metal-intensive (and therefore costly) solution one can utilize the techniques outlined in this chapter to reduce the effects of crosstalk on interconnect design. Certainly there are benefits from these techniques; however, they are not applicable in all situations and therefore only provide a partial solution to the problem. Ultimately, it would seem that on-chip latencies must be increased to facilitate shorter and less problematic wiring segments, at the expense of processor performance.

## ACKNOWLEDGMENTS

I would like to thank Mark Miller and Adam Shepela for their contributions to the technology scaling issues; and Michael Tsuk and Johann Nittmann for discussions on electromagnetism.

## REFERENCES

- [1] D. A. Priore, "Inductance on Silicon for Sub-Micron CMOS VLSI," *Proc. IEEE Symposium on VLSI Circuits*, Kyoto, Japan, pp. 17–18, May 1993.

- [2] Y. I. Ismael, E. G. Friedman, and J. L. Neves, "Figure of Merit to Characterize the Importance of On-Chip Inductance," *DAC 98*, San Francisco, CA, pp. 560–565.
- [3] S. V. Morton, "On-chip Inductance Issues in Multiconductor Systems," *DAC 99*, New Orleans, June, pp. 921–926.
- [4] R. E. Matick, *Transmission Lines for Digital and Communication Networks*. McGraw-Hill, New York, 1969.
- [5] W. Sinnema, *Electronic Transmission Technology*. Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [6] G. Miner, *Lines and Electromagnetic Fields for Engineers*. Oxford Press, Oxford, 1996.
- [7] C. J. Alpert, "Wire Segmenting for Improved Buffer Insertion," *DAC 97*, ANAHEM, CA, June 97, pp. 588–593.
- [8] J. Lillis, C. K. Cheng and T. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," *IEEE JSSC*, vol. 31, no. 3, pp. 437–446, March 1996.
- [9] Y. Nakagome, et al., "Sub-IV Swing Internal Bus Architecture for Future Low-Power ULSI's," *IEEE JSSC*, vol. 28, no. 4, pp. 414–419, April 1993.
- [10] M. Hiraki, et al., "Data-Dependent Logic Swing Internal Bus Architecture for Ultralow-Power LSI's," *IEEE JSSC*, vol. 30, no. 4, pp. 397–402, April 1995.
- [11] H. Yamauchi and A. Matsuzawa, "A Signal-Swing Suppressing Strategy for Power and Layout Area Savings Using Time-Multiplexed Differential Data-Transfer Scheme," *IEEE JSSC*, vol. 31, no. 9, pp. 1285–1294, Sep. 1996.
- [12] M. M. Khellah and M. I. Elmasry, "Low-Power Design of High-Capacitive CMOS Circuits Using a New Charge Sharing Scheme," *IEEE ISSCC*, vol. 42, pp. 286–287, Feb. 1999.
- [13] "National Technology Roadmap for Semiconductors," pp. 99–113, *SIA 97 Edition*.
- [14] M. R. Stan and W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, pp. 49–58, March 1995.
- [15] M. Nekili and Y. Savaria, "Optimal Methods of Driving Interconnections in VLSI Circuits," *Proc. IEEE ISCAS*, pp. 21–24, May 1992.
- [16] M. Nekili and Y. Savaria, "Parallel Regeneration of Interconnections in VLSI & ULSI Circuits," *Proc. IEEE ISCAS*, pp. 2023–2026, May 1993.
- [17] M. Izumikawa and M. Yamashina, "A Current Direction Sense Technique for Multiport SRAM's," *IEEE JSSC*, vol 31, no. 4, April 1996.

Stephen C. Thierauf  
*Compaq Computer Corporation*  
Warren R. Anderson  
*Compaq Computer Corporation*

## 18.1 INTRODUCTION

The signaling protocol used by chips to exchange information may employ voltages the same as or different from the voltages used throughout the chip (the “core logic”). Off-chip drivers translate core logic signals into signals having the proper voltage, current, and slew rate for faithful conveyance along communication channels formed by printed wiring board or cable transmission lines. If the channel is bidirectional the driver must have the ability to be disconnected (placed in the high impedance state) so as not to interfere with received signals and may need to withstand signaling voltages capable of destroying circuits in the core. A thorough understanding of the interconnect bringing power to the off-chip driver is vital for proper signaling and for identifying transistors exposed to unacceptably high voltage. Receivers are used to detect the presence of the signal on the channel and, if necessary, translate it to a voltage compatible with the core logic.

Off-chip driver and receiver circuits are particularly sensitive to electrostatic discharge events that may occur during handling. Special ESD circuits must dissipate these extremely high current occurrences while clamping the pin voltage to safe levels. The ESD circuit must not adversely interact with the chip power supply or off-chip-driver circuits, and should not require special masking steps that increase manufacturing complexity, and hence, cost.

This chapter begins with a discussion of the power supply distribution system, and the factors needing consideration when deciding if the off-chip drivers should be powered separately from the core logic. Techniques to mitigate noise on the off-chip-driver power supply rail are introduced, and methods commonly used to control edge rate are demonstrated using a simple driver design. These same techniques can be applied to the more complicated drivers used in mixed voltage signaling (such as the floating well driver presented in Section 18.4.1). Cascoding is introduced as a way to protect transistors from high voltages, and a circuit for level shifting from low core voltage to higher off-chip voltages is presented. Both techniques are described in the context of driving off chip, but apply equally to receiving high voltages. The chapter concludes by presenting several methods to build ESD protection for signal and power supply pins that satisfy the three industry-standard ESD tests.

## 18.2 POWER SUPPLY CONSIDERATIONS

The power supply distribution of most micropackages has high inductance and relatively low resistance, making them underdamped RLC tank circuits. Changing the state of one or more off-chip drivers causes ringing in the on-chip power supply system often called “simultaneous switching noise,” (SSN) unless steps are explicitly taken to improve the damping factor [Eq. (18.1)]. The on-chip ringing can be large enough to lead to transistor failure or degradation (due to time-dependent dielectric breakdown and hot carrier effect stresses) if the damping factor is suitably small. Maintaining a damping factor of 0.6 or larger will keep power supply ringing below 10%.

$$\text{Damping factor} = \frac{R\sqrt{LC}}{2L} \quad (18.1)$$

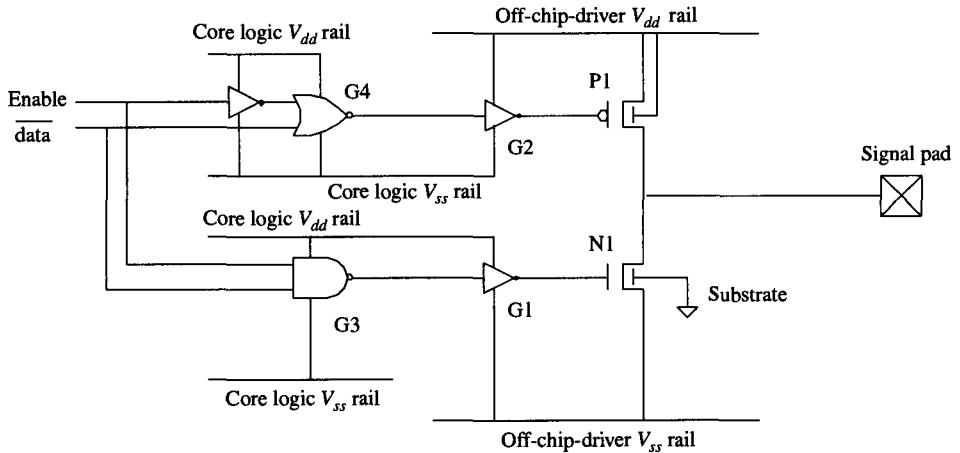
It's evident from Eq. (18.1) that increasing the series resistance has the most profound effect on the damping factor. A small resistance (usually a few ohms) individually added to the source of the final drive transistor in each output cell can be effective in reducing power supply noise by taking advantage of negative feedback due to body effect. The resistor is selected to be small enough to preserve dc noise margin, but large enough to provide a useful amount of negative feedback. Adding resistance to the drain is not as beneficial because it does not utilize body effect. The inductance of the  $V_{dd}/V_{ss}$  “loop” can be reduced by careful physical design of the micropackage. For example, using power and ground planes in the micropackage, and running  $V_{dd}$  and  $V_{ss}$  lines parallel to each other at the closest possible spacing in the package and on the die will encourage high mutual coupling. This reduces the loop inductance by a value equal to twice the mutual inductance since the current will be flowing in opposite directions. On-chip capacitance can be added [generally required in prestigious quantities since it's under the square root sign in Eq. (17.1)] to reduce ringing of the off-chip-driver supply. Adding capacitance will not alter the L/R time constant, but it will reduce the magnitude and frequency of the ringing.

### 18.2.1 Split Power Supply Systems

In general the off-chip driver can either be connected to the core supply or to a separate, dedicated supply.

Separating the supplies prevents off-chip-driver switching noise from disturbing the core power supply and is attractive for small chips, but is generally not advantageous for chips with large core  $V_{dd}/V_{ss}$  capacitance. The off-chip drivers on these larger chips can reduce SSN by taking advantage of decoupling capacitance explicitly added between the core power and ground rails.

The simplified push-pull output driver in Fig. 18.1 shows the proper split-supply wiring. The drawing assumes the core logic and off-chip-driver supply rails have the same nominal values. Gates G3 and G4 allow the driver to be placed in the high impedance state and provide the interface between the driver and the core logic. They are powered from core voltage to provide the best noise immunity and can be replaced with level shifters (Section 18.4.4) when the off-chip-driver supply has a different nominal voltage than the core logic supply. Gates G1 and G2 drive the final output transistors (N1, P1, respectively) and as shown are connected to the appropriate power and ground systems to provide the best drive capability and noise immunity for



**Figure 18.1** Proper off-chip-driver split-supply connections.

transistors P1 and N1. For example, P1 would momentarily come on when ringing on the off-chip-driver  $V_{dd}$  rail exceeds core logic  $V_{dd}$  by a threshold voltage had P1 been turned off by connecting its gate to core logic  $V_{dd}$ . Instead, this can be avoided by using G2 to connect P1's gate to its source (the off-chip-driver  $V_{dd}$  rail) as shown.

Turning on P1 by pulling its gate to the clean core logic  $V_{ss}$  gives P1 the best (and most consistent) gate drive. A complementary scheme is used for G1/N1. When employing CMOS bulk process, transistor N1 is subject to body effect because its body connection is to the bulk (tied to core  $V_{ss}$ ) while its source is connected to the off-chip driver  $V_{ss}$ . Noise here relative to the substrate will adjust the drive strength of N1. The well and source of P1 are connected to the off-chip-driver  $V_{dd}$  rail, and so are not affected in this way.

### 18.2.2 Power Supply Clamps

On-chip clamping circuits connected to the power supply feeding the off-chip drivers can be effective in suppressing power supply ringing in split-supply designs, especially if the off-chip-driver supply has inadequate power supply decoupling. The clamps come on when  $V_{dd}$  rings above the nominal value, and either shunt the excess into the core supply or to  $V_{ss}$ . Shunting to the core supply works well if it has large amounts of decoupling and has the same nominal voltage as the off-chip-driver supply. The shunt is formed by a string of diodes (or MOS transistors). The clamps to  $V_{ss}$  are useful when the core has insufficient decoupling to absorb ringing of the off-chip-driver power supply or when the nominal voltage of the core supply is too high to clamp the off-chip-driver supply to safe levels. These clamps are especially attractive when the nominal core voltage is too low, thereby requiring too many series-connected diodes to be practical. The clamp uses a fast-acting voltage sensor to detect the ringing and to activate a large NMOS connected between the power and ground rails feeding the off-chip driver. For both designs the clamps must be fast enough to effectively clip off the excess ringing energy and must not disconnect before the clamping has been fully achieved. Additionally, the clamps must not cause a secondary resonance by coming on too sharply, and they must not interfere with normal ESD operation.

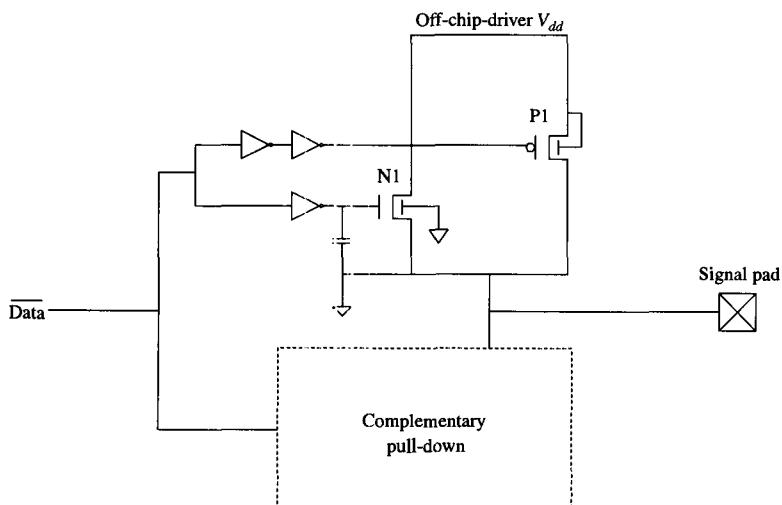
### 18.3 OFF-CHIP-DRIVER EDGE RATE CONTROL

Launching edges sharper than necessary fosters simultaneous switching noise problems, and sharp edges will cause even relatively short lengths of interconnect to appear as transmission lines, raising the likelihood of reflections. Additionally, edges sharper than required will encourage crosstalk between closely spaced off-chip conductors. The micro-package is especially prone to this coupling because of the tight wiring pitch and the likely absence of local return paths in portions of the package.

Schemes used to control the launched edge rate often do so by segmenting the final output stage into multiple parallel transistors and sequentially switching them on [1]. The switching action can be discreet (self-timed logic pulses) or based on an analog delay line. In both cases the goal is to gradually increase the off-chip-driver's output current as a way to extend the charging time of the load capacitance. This works well if the load is a capacitance, but these drivers produce a choppy waveform if the load is a transmission line driving a purely resistive load. However, the load is usually capacitive and will integrate the steps to produce a smooth waveform provided the steps are small enough and the capacitance large enough. Another way to reduce launched edge rates is to use a single transistor driven with a gate voltage having a retarded ramp.

Body effect (the increase in threshold voltage as the source to bulk voltage increases) can be used to advantage by adding resistance to the source of the final drive transistor (N1 and P1 in Fig. 18.1). The body-effect coefficient causes  $V_t$  to increase approximately as a process-dependent constant times the square root of  $V_{sb}$ . The proper resistance value (usually just a few ohms) can help reduce the driven edge rate without adversely impacting static drive levels and is increasingly effective as the drive strength increases due to process variations. Thus the proper amount of source resistance can help reduce the variation in drive strength between the fastest and slowest process corners but is not intended to impedance match the driver to the transmission line.

A source follower scheme can also be employed to reduce launched edge rates. The gate of the final stage is driven with a voltage ramp, which the load connected to the source mimics. The NMOS pull-up is shown in Fig. 18.2, with the dual of this being a



**Figure 18.2** Source follower edge rate control.

PMOS pull-down (not shown). The gate of N1 is driven with a greatly retarded edge rate, usually obtained from a grossly undersized inverter or from a current source charging a capacitor [2], [3]. Transistor N1 cannot pull all the way to the power supply rail and tends to weakly drive the last portion of the rising waveform. To correct this a self-timed PMOS (P1) is placed in parallel with the NMOS. In Fig. 18.2 the self-timing is shown to be triggered by the input signal, but it can also be taken from the pad by way of a fast-responding voltage sensor. An extension of this idea is to control the gate drive by feeding back the pad voltage to the gate drive circuitry to maintain a constant voltage rate of change [4], [5].

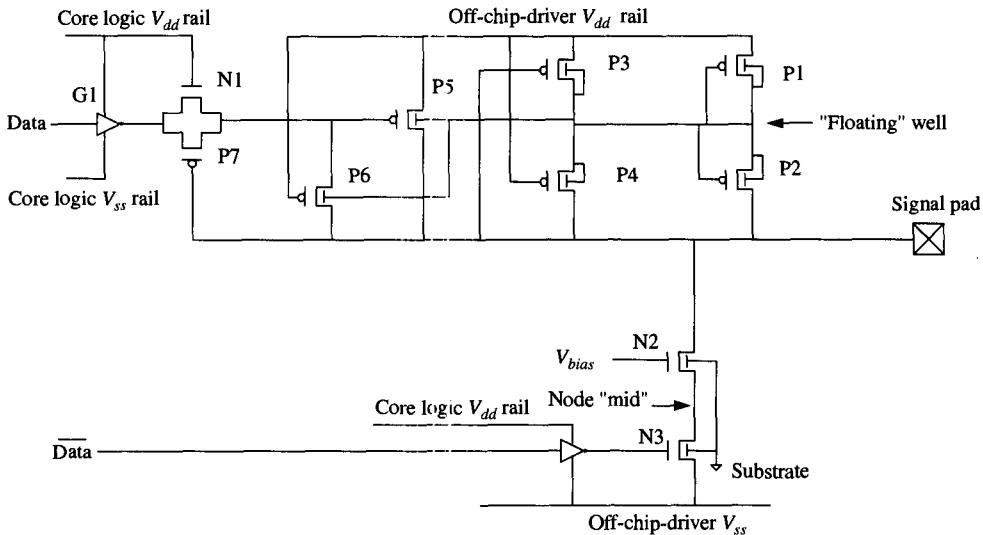
## 18.4 MIXED-VOLTAGE I/O

In some applications the core voltage is not the same as the signaling voltage, with the greatest design challenge occurring when the signaling voltage is higher than the core voltage. Generally these high signaling voltages approach or exceed the HCE and TDDB process limits. Circuit design techniques such as cascoding or transistors with thicker than usual gate oxide are used to ensure reliability.

### 18.4.1 Floating Well Driver

A PMOS pull-up is usually the best way to guarantee good edge rates and predictable  $V_{oh}$  values across a wide  $I_{oh}$  range. However, employing a PMOS is problematical in a bidirectional output driver that is in the high impedance state and is receiving voltages greater than the voltage powering the off-chip driver. In this situation the output driver PMOS and the parasitic PNP transistors formed by the N-well will become forward biased, providing a dc path from the signal line to  $V_{dd}$  and  $V_{ss}$ . This clamping would be beneficial if it only occurred transiently to clip overshoots, but the clamping is continuous when the received signal exceeds  $V_{dd}$  of the off-chip driver. Continuous clamping results in dc power dissipation in the sending device's off-chip driver, erodes high going noise margins, and is a possible latchup and electromigration concern for the receiving chip.

A “floating well driver” eliminates these problems (Fig. 18.3). The well is said to “float” because it is not firmly connected to  $V_{dd}$ , but instead is allowed to take on either the value of the pad voltage or on-chip  $V_{dd}$ , whichever is higher at any given time. A PMOS (P6) is used to selectively connect the gate of the PMOS pull-up (P5) to the pad when receiving voltages in excess of on-chip  $V_{dd}$ , and so preventing P5 from coming on. The parasitic N-well PNP is prevented from becoming forward biased by using PMOS transistors (P1–P4) to connect the well to either the signal voltage or to on-chip  $V_{dd}$  [6]. The MOS transistors used to connect the well to the pad (P2, P4) must not allow the well to be filled too quickly since doing so will cause a low going reflection to propagate back down the transmission line. However, transistors too small will not fill the well quickly enough to prevent forward biasing the parasitic PNP. Transistors P2 and P4 must be sized to satisfy both constraints. Pass-gate P7/N1 protects gate G1 from the high voltage present at the gate of P5 when the pad is exposed to an overshoot. An overshoot causes P6 to turn on but turns off P7. NMOS N1 will conduct but acts as a source follower, limiting the voltage seen by G1. When receiving a low going signal the well must be discharged back to  $V_{dd}$ . This reduces the voltage stresses and properly



**Figure 18.3** Floating well driver. Pull-up P5 is disabled by P6 when receiving overshoots.

biases PMOS P5 when it is eventually turned on to pull the bus high. The well is discharged to  $V_{dd}$  rather than to the signal line by transistor P3 to ensure that the well does not fall below  $V_{dd}$ . Making P3 too large will disturb  $V_{dd}$  when many drivers simultaneously dump the charge stored in each of their wells. Making P3 too small causes the well to be discharged too slowly to prevent damaging stresses. Transistor P1 ensures that the well does not fall too far below  $V_{dd}$  and generates a charging path during the first application of  $V_{dd}$  at power-up.

#### 18.4.2 Open Drain Signaling

Another solution to high voltage signaling is to eliminate the PMOS problem entirely by using off-chip resistors as pull-ups. When properly terminated and biased, open drain drivers can be used to bidirectionally signal point-to-point at high speed [7] and can form “wired-OR” buses. As shown in Fig. 18.4 the bus is pulled down by transistor M1 and impedance matched to the printed circuit board transmission line by resistor R2, and pulled up by resistor R1. Locating R1 off-chip eliminates the need to bring  $V_{term}$  on-chip, while locating R2 off-chip isolates the inductance of the micro-package from the signals on the printed wiring board transmission line. This reduces reflections when several bits are simultaneously in flight along the transmission line.

The ability of this scheme to signal at voltages unrelated to on-chip  $V_{dd}$  is attractive when it's necessary to employ the same signaling voltage across succeeding generations of silicon, each with progressively lower values of  $V_{dd}$ .

Because the pull-up is not under the control of the off-chip driver circuits, the bus will be pulled down faster than it will pull up, introducing a skew component that can adversely affect the overall timing budget. A second disadvantage is that the switching current only flows through the on-chip  $V_{ss}$  system. This contrasts to a push-pull scheme where on average about as much current is pulled from the on-chip  $V_{dd}$  system as is put

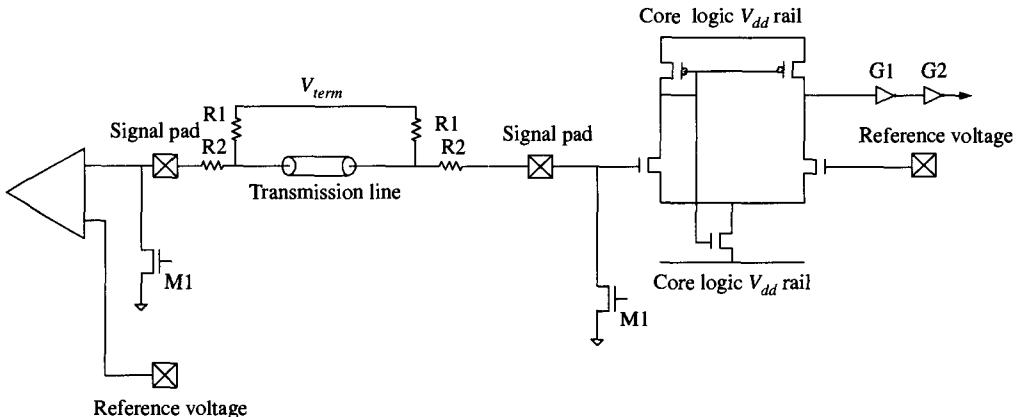


Figure 18.4 Open drain signaling with unclocked differential amplifier receiver.

into the on-chip  $V_{ss}$  system. The asymmetrical nature of the switching currents can lead to asymmetrical changes in  $V_{dd}/V_{ss}$  that are not easily countered by the on-chip decoupling capacitance.

### 18.4.3 Cascoding

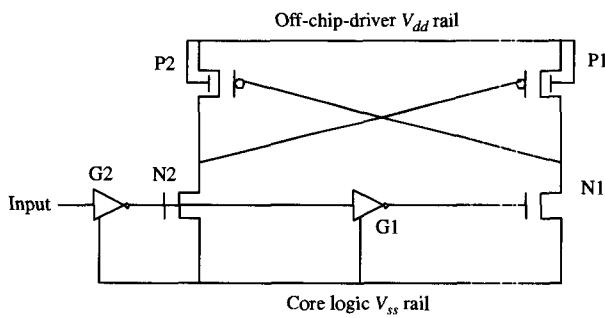
When receiving high voltages the stresses across the gates and junctions of the NMOS pull-down can become too large unless precautions are taken, such as cascoding. High voltage signals do not stress the PMOS pull-up because the gate and source are protected by their connection to the off-chip-driver supply. However, the NMOS pull-down gate voltage is zero when it's off, so high voltage at the pad will expose the gate oxide and junction to stress. The cascode (shown in the lower portion of Fig. 18.3) prevents this by dividing stress between transistors N2 and N3. The stresses on N2 can be managed with proper selection of a bias voltage ( $V_{bias}$ ). When receiving high going signals the voltage at node mid will not, in concept, rise above a body-effected threshold below the bias voltage  $V_{bias}$ . Typically  $V_{bias}$  is equal to core  $V_{dd}$ , since it's readily available and the transistors are capable of withstanding voltage to at least that level. Therefore,  $V_{dg}$  and  $V_{ds}$  of N3 should not rise to dangerously high levels. Dynamic coupling of  $V_{bias}$  noise or of the signal itself can cause node mid to rise above ( $V_{bias} - V_{t\ body}$ ). The leakage can be addressed by making N2 a long-channel device (also benefiting hot carrier effect stresses), but doing so will require an increase in its width, increasing the junction capacitance seen at the pad and further encouraging dynamic coupling into node mid. To avoid this transistor N2 must not be made too large, but if made too small nearly the entire bus voltage will be presented across N2 when N3 is turned on, negating the cascode benefits. Typically N2 is made 2 to 4 times the size of N3.

The cascode provides strong protection for high going signals that exceed  $V_{dd}$ , but does not provide protection to signals that reflect below  $V_{ss}$ . Under these circumstances the gate oxide of transistor N2 can be in jeopardy if  $V_{bias}$  is as high as core  $V_{dd}$ . Therefore,  $V_{bias}$  must be high enough to protect against the highest expected low-to-high overshoot voltage, but low enough not to stress N1 when a signal reflects below  $V_{ss}$ . Alternatively, circuits can be used to automatically adjust  $V_{bias}$  in response to the voltage present at the pad [8], [9]. These circuits must operate quickly and must contain

logic to prevent them from coming on when the driver itself is pulling the bus low. Similar circuits can be used to protect the PMOS pull-up from stress when the pad rings below  $V_{ss}$ .

#### 18.4.4 Level Shifting

The cross-coupled circuit shown in Fig. 18.5 level shifts low voltage swing core signals to the higher voltage used by the off-chip drivers. Gates G1/G2 operate from the core supply and cause transistors N1/N2 to pull down the gate of either P1 or P2 to  $V_{ss}$ . The gates of these transistors are the outputs of this circuit and are buffered with an inverter operating from the off-chip-driver supply. Transistors N1 and N2 may be cascoded if necessary to reduce voltage stresses.



**Figure 18.5** Level shifter used to cross power supply domains.

## 18.5 IMPEDANCE MATCHING

Ideally an off-chip driver is connected to a printed wiring board transmission line that is terminated at the load (the “far end” of the transmission line) in the transmission line’s characteristic impedance. Such far-end termination is usually created with resistors (or MOS transistors configured as resistors) and in various forms is often used in high performance. Impedance matching at the driving end of a transmission line can prevent (or at least mitigate) multiple round-trip reflections along the transmission line by absorbing the reflected energy. Reducing this energy is important for reducing the severity of crosstalk and intersymbol interference [10], [11]. However, the drive current of a simple CMOS off-chip driver can vary by as much as 2:1 across process, and more than that if the operating temperature range is large. Such a variation makes it impossible to exactly impedance match the driver to the transmission line. Series terminating resistors, located either on- or off-chip but electrically near the driver, can provide a tighter match. On-chip resistors are typically made from N+ or well and have a resistance variation of up to  $\pm 20\%$ . This wide range is due to variations in sheet resistivity (usually held to about  $\pm 10\%$ ), dimensional errors in processing (delta-L and delta-W), resistance increases due to self-heating, and voltage-induced resistance changes. Surface mount resistors with a tolerance of  $\pm 2\%$  or better are usually employed for off-chip termination. However, the superior tolerance will not be advantageous unless the surface mount resistors are optimally placed on the printed wiring board, immediately adjacent to the micropackage.

Biased transistors can also be used as an off-chip-driver terminator [12]. They are especially useful if the signal swing is low enough to allow the transistor to remain in the linear region while maintaining the desired resistance. These circuits can be made programmable, allowing the resistance value to be adjusted after the chip has been installed in a system. This is often beneficial during system troubleshooting, and leads to self-calibration techniques that null out errors due to processing or local heating [13].

## 18.6 PRECOMPENSATION DRIVERS

The harmonics of a pulse propagating along a printed wiring board copper etch or cable experience attenuation as a function of their frequency due to skin effect. The reconstructed pulse at the receiving end of the cable will be attenuated and distorted since it's comprised of greatly attenuated upper harmonics and less attenuated lower ones and is a cause of intersymbol interference [14].

Adjusting the drive strength as a function of the data stream [15], [16] can reduce intersymbol interference. Such digital precompensation circuits cause the off-chip driver to drive strongly when the data are transitioning but reduce their drive strength when transmitting a string of the same polarity bits. Successful implementation requires a detailed knowledge of the transmission path prior to designing the off-chip driver.

## 18.7 INPUT RECEIVERS

The circuit shown in Fig. 18.6 is useful when receiving signals greater than  $V_{dd}$ . Cascode transistor N2 couples the signal voltage present at the pad to the input of G1. Conceptually it's one transistor that has been divided into two to improve its ability to withstand ESD transients. Transistor N1 pulls the input of G1 to  $V_{dd}$  once the signal voltage exceeds  $V_{dd}$  by  $V_{t, body}$ . The imprecise switchpoint makes this circuit unsuitable for many signaling applications, but it is useful when slower-speed high-voltage signals (such as configuration pins or test pins loaded once at power-up) must be detected.

Differential amplifiers (both unclocked, as in Fig. 18.4 and clocked, as in Fig. 18.7) make excellent receivers in high-speed systems. The switch point is set by a “reference voltage” applied to one side of the diffamp, and may be generated on- or off-chip. Off-chip application of  $V_{ref}$  allows fine-tuning during system debug. To prevent excessive set-up and hold times the signals must lie within the common-mode range of the receiver.

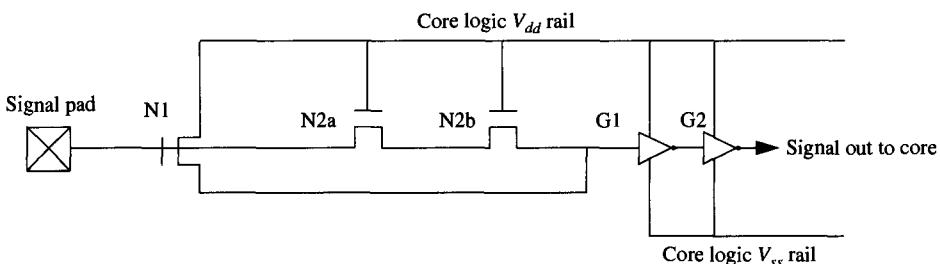


Figure 18.6 High-voltage protection for simple ratioed receiver.

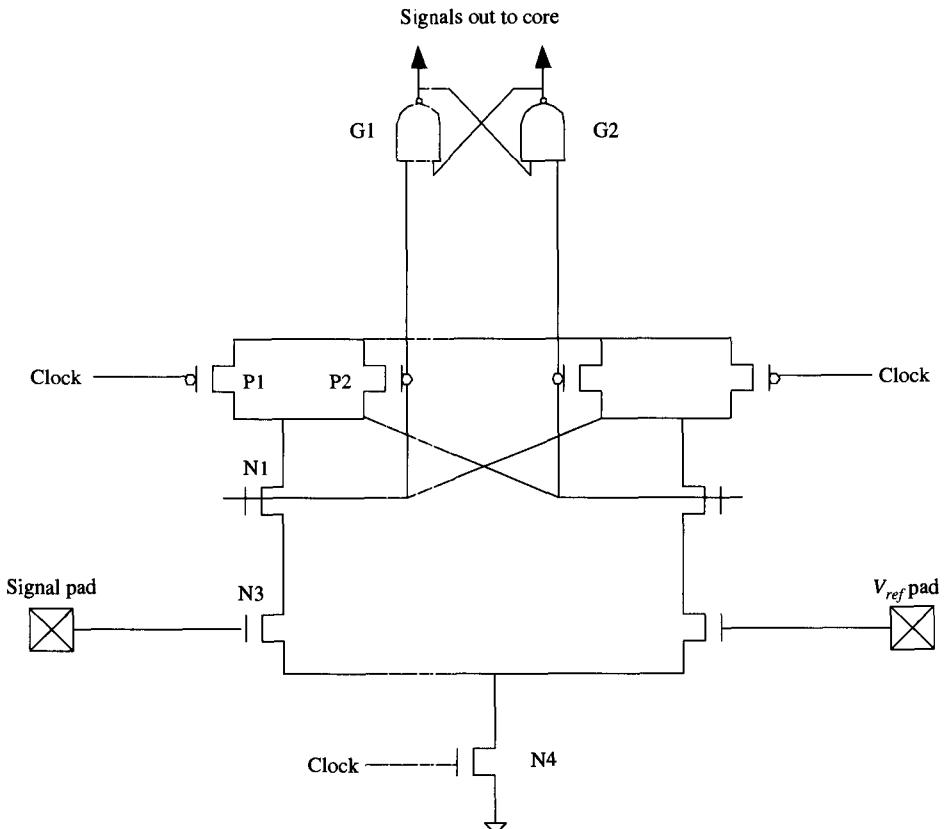


Figure 18.7 Latched differential amplifier receiver.

Clocked diffamp circuits such as Fig. 18.7 [8] can have zero (or even negative) set-up times and can precisely distinguish signals near  $V_{ref}$ . When the clock is low the amplifier is precharged by PMOS P1, allowing cross-coupled gates G1/G2 to retain the previously stored data. Once the clock rises, each of the outputs of the diffamp begin to fall, but the side with the higher voltage on N2's gate will fall faster. This allows transistors P2 and N1 to hinder the voltage drop on the opposite side of the diffamp. Eventually enough differential is obtained to flip cross-coupled gates G1/G2. The G1/G2 outputs are either immediately buffered or drive into a nearby latch. Extending the wires too far will add load and increase the susceptibility to coupling. For similar reasons the G1/N1 connections should be kept short.

## 18.8 THE ESD THREAT

I/O circuits are particularly sensitive to electrostatic discharge (ESD) events, which can cause permanent, catastrophic damage, leading to functional failures, manufacturing yield loss, and field returns. The IC handling environment may generate static voltages as high as 3 kV to 5 kV in high humidity and 35 kV in low humidity. To combat ESD, the industry has adopted a dual strategy. The first approach attempts to prevent ESD itself by controlling the buildup and release of electrostatic charge with wrist straps and

other antistatic means. The second approach decreases the ESD sensitivity of the integrated circuit by installing explicit circuits to safely dissipate ESD events. The level of explicit protection is minimal, typically 2 kV, which does not even surpass the 3 kV human sensitivity threshold.

## 18.9 ESD MODELS

An IC's ESD sensitivity may be measured with three industry-standard stress tests that attempt to reproduce the IC handling environment. These are the human body model (HBM), machine model (MM), and charged device model (CDM) [17].

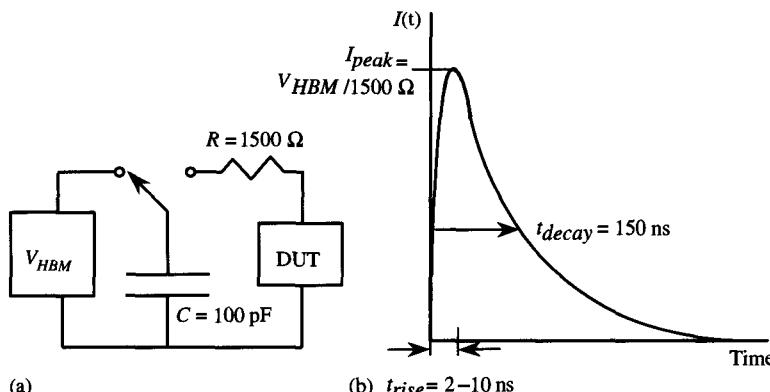
HBM, the most widely adopted model, represents the discharge to an IC through skin contact. As shown in Fig. 18.8(a), HBM consists of a 100 pF capacitor energized to the test voltage, then discharged to the integrated circuit through a 1.5 k $\Omega$  resistor. Current rises to its peak level in 2 to 10 nsec and decays exponentially with a 150 nsec time constant. The typical HBM tolerance for an IC is between 2 kV and 4 kV.

MM recreates a charged machine or human discharging through metal. It discharges a 200 pF capacitance through a 0.75  $\mu$ H inductor and minimal resistance, producing a decaying sinusoidal current with a period of roughly 60 ns. Products typically withstand 200 V to 400 V.

CDM represents a charge induced on the IC itself discharging to ground. Robotic CDM testing is performed with the IC in dead bug orientation on a field plate. After a slow charge, the CDM discharge occurs when a grounded pin on a robotic arm contacts the IC pin. The entire event takes place in less than a few nanoseconds.

The HBM and MM test standards require individual discharges between every pin and every supply rail, one by one, in both polarities, as well as an I/O to I/O pin test. Pins not participating in the discharge float. Therefore, all permutations of pins and polarities must be considered during ESD protection design.

Although the ESD-withstand level for all of these models is specified in volts, the discharge current, not its voltage, controls the response of the ESD protection circuit. In the HBM case, the load impedance of the device under test is much lower than the

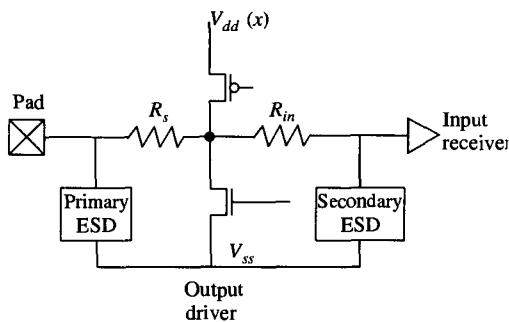


**Figure 18.8** The human body model for ESD: (a) equivalent circuit model and (b) current waveform into a short.

$1.5\text{ k}\Omega$  source resistance. The peak current is given by  $V_{HBM}/1500\Omega$ , 1.3 amps at the typical 2 kV qualification level. For the other models, the response of the device under test contributes to the discharge waveform. However, since the current through a device usually sets its failure level, it is most convenient to think of all ESD models as transient current sources and analyze the device behavior in response to the current.

## 18.10 CIRCUIT TOPOLOGY OF THE ESD PROTECTION NETWORK

Figure 18.9 shows the general configuration of the ESD protection in a bidirectional I/O circuit. The output driver is typically most susceptible to ESD failure, especially for HBM. The primary clamp shunts the majority of the ESD current to  $V_{ss}$  and limits the pad voltage below that at which the output driver fails. The series resistor  $R_s$ , if present, further limits the current that can flow through the output driver. The resistor  $R_{in}$  and the secondary clamp protect the input receiver's gate oxide against damage during CDM events.

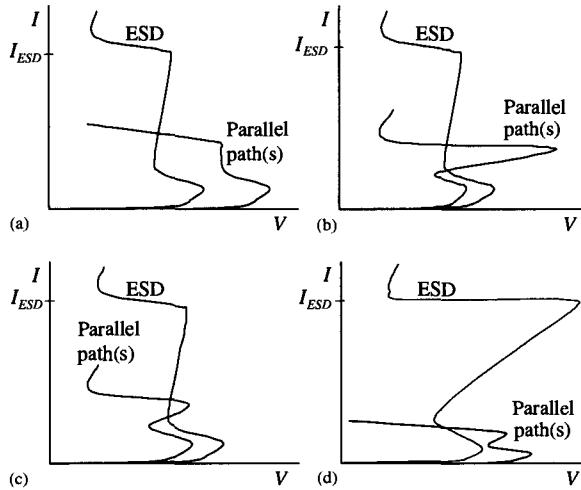


**Figure 18.9** The general configuration of the ESD protection in a bidirectional I/O circuit. The secondary clamp is usually a scaled-down version of the primary clamp. The input resistor  $R_{in}$  is usually much larger than the output series resistor  $R_s$ .

### 18.10.1 The Qualities of Good ESD Protection

The capability of the clamp network can be described in terms of its performance in four categories: robustness, effectiveness, speed, and transparency. Robustness is defined as the highest ESD level at which the clamp, taken on its own, can safely withstand an ESD event without failing. Effectiveness describes the highest ESD level where the clamp network can limit the pad voltage to a safe level such that circuits in parallel with the ESD protection do not turn on and fail. Figure 18.10 illustrates several effectiveness examples, using an  $I-V$  characteristic typical of many types of devices under ESD conditions, namely a trigger into a conduction mechanism at low current and a failure at higher current. Since ESD pulses are current driven, the current increases rapidly to its maximum, with the voltage tracking at the minimum necessary to support the current. If, at a current level  $I_{ESD}$ , the ESD clamp keeps the weakest parallel path from reaching its failure point, the ESD protection is effective.

ESD networks must activate with enough speed to clamp the ESD event safely. Although inherent in nearly all protection schemes, some clamps can trigger so slowly that the pad voltage exceeds safe levels for long enough to cause circuits in parallel to



**Figure 18.10** Illustrations of the robustness and effectiveness concepts comparing the  $I$ - $V$  characteristics of the ESD path with those of the other paths in parallel with the ESD: (a) and (b) robust and effective networks; (c) and (d) robust networks that are *not* effective.

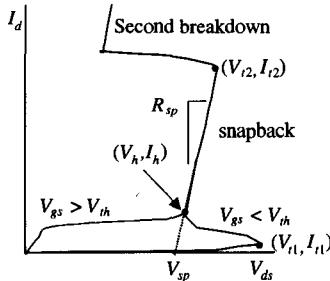
fail. Good ESD should be transparent to the normal operation of the I/O circuit. For example, the ESD clamp on an I/O should not have a capacitance above the loading limits of the signaling specification. This becomes particularly important for high-frequency signal pins.

## 18.11 ESD PROTECTION DESIGN ELEMENTS AND METHODS

### 18.11.1 Nonsilicided and Silicide-Blocked NMOS

ESD protection networks using the grounded-gate, nonsilicided NMOS device have received wide use, although many alternatives have appeared in recent years. Unfortunately, silicided junctions and lightly doped drains (LDD), both common to today's CMOS processes, reduce the ESD robustness of the grounded gate NMOS by a factor of ten [18]. To realize a grounded-gate, nonsilicided NMOS protection strategy in a silicided technology, many CMOS processes offer special silicide blocking steps and abrupt junction implants [19].

The NMOS achieves its ESD protection acclaim through the voltage-limiting ability of the device in snapback, Fig. 18.11. In snapback the NMOS can operate well above its saturation current while limiting the voltage on the drain [17], [20].



**Figure 18.11** The high-current, short-duration pulsed  $I$ - $V$  characteristics of an NMOSFET with  $V_{gs} < V_{th}$  and  $V_{gs} > V_{th}$ . The trigger point occurs at  $(V_{t1}, I_{t1})$ . Snapback takes place on the portion of the  $I$ - $V$  between the holding point at  $(V_h, I_h)$  and  $(V_{t2}, I_{t2})$ . Second breakdown takes place for currents above  $I_{t2}$ .

With  $V_{gs} < V_{th}$ , the NMOS enters snapback through the avalanche point at  $(V_{t1}, I_{t1})$ . If the gate voltage is above threshold, the NMOS enters snapback through the saturation region, without encountering a negative differential resistance region. A device conducting in snapback exhibits a resistance  $R_{sp}$ . Permanent, destructive failure, referred to as second breakdown, occurs at the point  $(V_{t2}, I_{t2})$ . A nonsilicided or silicide-blocked, abrupt junction device typically fails between 10 and 20 mA/ $\mu\text{m}$  gate width.

Figure 18.12(a) shows the typical circuit topology for a grounded-gate NMOS ESD device protecting an output driver, with example  $I-V$  characteristics shown in Fig. 18.12(b). The ESD device must be sized to sink the peak ESD current without failing. For effectiveness, the failure voltage of the output driver leg  $V_{t2OD}$  at its failure current  $I_{t2OD}$  must be higher than the trigger voltage of the ESD leg  $V_{t1ESD}$ . The effectiveness requirement ensures that the ESD device will trigger into snapback even if the output driver also triggers. To achieve this requirement, we add resistance to the output driver path, such that

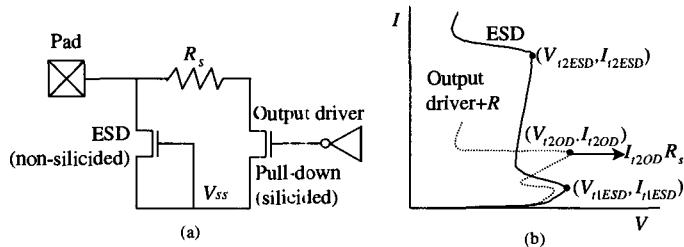
$$V_{t2OD} + I_{t2OD}R_s > V_{t1ESD} \quad (18.2)$$

or

$$R_s > \frac{(V_{t1ESD} - V_{t2OD})}{I_{t2OD}} \quad (18.3)$$

For example, if  $V_{t1ESD} = 11 \text{ V}$ ,  $V_{t2OD} = 9 \text{ V}$ , and  $I_{t2OD} = 200 \text{ mA}$ , then  $R_s \geq 10 \Omega$ . Alternatively, if signal integrity requirements specify  $R_s$ , Eq. (18.2) can be rewritten to specify a minimum output driver size. Care must be taken to account for nonlinearities and catastrophic breakdown at high currents in short resistors [21].

The resistance requirement is particularly important in silicide-blocked processes, where the output driver may be fully silicided, causing it to have snapback  $I-V$  characteristics significantly different from the ESD device. The resistance  $R_s$  may either be added explicitly or may be incorporated into the drain junction of the output driver by increasing its contact-to-gate spacing and drawing the silicide block layer around the driver.



**Figure 18.12** (a) Grounded-gate NMOS ESD protection circuit, and (b)  $I-V$  requirements for effective protection.

### 18.11.2 Silicided NMOS

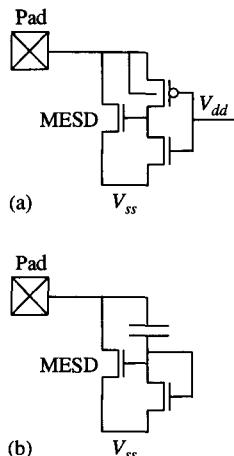
Since all high-performance CMOS processes contain silicided junctions, realizing the silicide-blocked NMOS costs an extra masking step, increasing processing cost and complexity. Therefore, an ESD protection design using a standard silicided NMOS has significant advantages. Employing this device requires uniformly triggering all of the fingers of the NMOS into snapback, which may not occur for two reasons [22]. First, the lower  $I_{t2}$

of the silicided, LDD NMOS (usually between 1 and 5 mA/ $\mu\text{m}$ ) requires a large width device to sink the total ESD current. This device must be laid out in a multifinger ladder network, often with 10 or 20 fingers. Second, nonuniformity in the silicide and its lack of ballasting resistance create variations in the grounded-gate snapback trigger voltage  $V_{t1}$  for each finger. Provided  $V_{t2} < V_{t1}$ , only the fingers with the lowest  $V_{t1}$  will trigger into snapback, and the overall device failure current, which depends randomly on process variations, will equal the  $I_{t2}$  per finger times the number of fingers that have triggered [23]. Adding fingers to increase the total device width will not improve robustness.

One solution to this problem uses gate modulation, a technique where a control circuit raises the gate bias above NMOS threshold  $V_{th}$  prior to triggering snapback during ESD [22]. All fingers smoothly enter snapback through saturation, bypassing the avalanche trigger point. Therefore, the entire device width sets the ESD failure level. However, the gate modulation circuit must set  $V_{gs} = 0$  during normal circuit operation.

A number of different gate modulation methods have been reported [20]. Power supply-referenced gate modulation uses the voltage on  $V_{dd}$  to distinguish between ESD and normal operation, Fig. 18.13(a). During normal operation, the voltage difference between  $V_{dd}$  and  $V_{ss}$  causes the gate modulation circuit to set the bias on the gate of ESD device MESD to  $V_{ss}$ . During an ESD event on the I/O, the modulation circuit detects a  $V_{dd}$  voltage near  $V_{ss}$  and sets the gate of the NMOS above  $V_{th}$ . Since  $V_{dd}$  floats during an ESD event between the I/O pin and a non- $V_{dd}$  pin, the decoupling capacitance between  $V_{dd}$  and  $V_{ss}$  keeps  $V_{dd}$  near  $V_{ss}$ .<sup>1</sup> Transient gate modulation uses the sharp rising edge of the ESD event itself to couple the NMOS gate above threshold, Fig. 18.13(b) [25], [26]. Unfortunately, as I/O signals become faster, this type of circuit will have more difficulty distinguishing between an ESD and an I/O switching event.

Synthesis of the ESD protection network with gate modulation is straightforward. If the ESD device and the output driver pull-down are constructed in the same fashion and have the same source connection, no series resistor is needed. This is because both the output driver pull-down and the ESD device have the same snapback  $I-V$



**Figure 18.13** Different methods for NMOS gate modulation: (a)  $V_{dd}$ -referenced gate modulation, and (b) gate-coupled modulation.

<sup>1</sup> This assumes that there are no current paths from the I/O pad to  $V_{dd}$ , that the decoupling capacitance between  $V_{dd}$  and  $V_{ss}$  is much larger than the capacitance between the I/O pad and  $V_{dd}$ , and that the  $V_{dd}$  to  $V_{ss}$  capacitance is large enough to absorb any parasitic transients from the ESD tester [24].

characteristics and therefore the same second breakdown voltage  $V_{t2}$ . If the source connections are different, for instance with a split supply output driver, the failure voltage of the path through the output driver must be increased, either by added series resistance or by output driver construction, to avoid failures on zaps between the I/O pad and the output driver's source supply.

### 18.11.3 Double-Diode ESD Protection

Diodes are extremely useful ESD protection devices. They are highly reliable under forward bias, with a forward breakdown current around  $50 \text{ mA}/\mu\text{m}$  device width. Their use under reverse bias is not recommended, since other devices break down below the diode avalanche voltage and since diodes in a silicided process fail at small reverse-bias currents. For ESD protection strategies based on other protection elements, diodes still participate in ESD discharges, for instance in shunting current from  $V_{ss}$  to  $V_{dd}$  through well diodes.

For diode-based I/O pad protection in a  $p$ -substrate,  $n$ -well technology, two diodes, one to the I/O driver supply rail  $V_{dd}(x)$  and one to  $V_{ss}$  are needed to provide discharge paths for all ESD polarities and pin combinations, Fig. 18.14. An  $n+$  diffusion in the substrate forms the diode to  $V_{ss}$ . This diode, which protects the output driver during a negative zap on the I/O pad, can either be drawn explicitly or can be implicit in the drain of the NMOS pull-down. In p-epi processes the highly doped substrate provides a good conduction path, and the  $n+$  diffusion diode generally conducts with minimal series resistance. Therefore, a reasonably sized diode can usually satisfy the ESD function of the  $n+$  diode. Of course, measurements on the appropriate test structures [27] should always be performed.

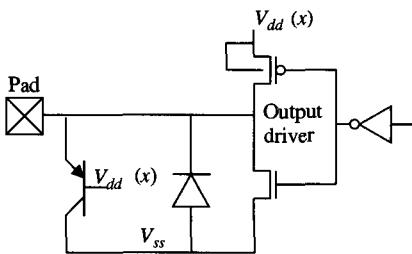
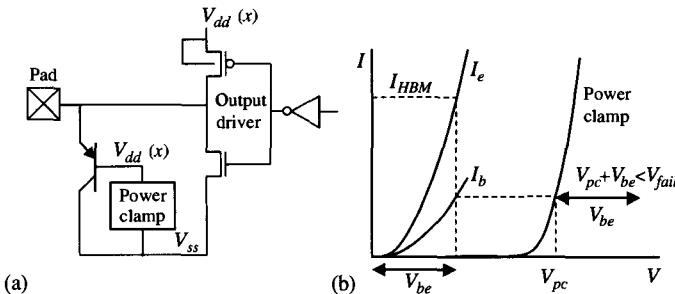


Figure 18.14 The double-diode ESD protection scheme. An  $n+$  diffusion in the substrate forms one diode. The emitter-base junction of the vertical PNP formed from a  $p+$  diffusion in an  $n$ -well composes the other.

A  $p+$  diffusion in an  $n$ -well creates the diode between the I/O pad and the power rail of the output driver pull-up (referred to here as  $V_{dd}(x)$ , since it may either be the core supply  $V_{dd}$  or an isolated I/O driver supply  $V_{dd}(x)$ ). This device is actually a vertical PNP transistor, with the  $p+$  substrate acting as a common collector. The diode forms between the emitter and base of the vertical PNP. Layout of the PNP is critical to achieve the minimum clamping voltage during ESD, since the device operates in the high-level injection regime. To minimize base resistance and provide maximum perimeter for perimeter-dominated injection mechanisms, the  $p+$  emitter diffusion should be laid out in minimum width strips, with the  $n+$  well plug surrounding the  $p+$  emitter on all sides at minimum spacing.

The vertical PNP's  $I-V$  characteristics and its scaling as a function of width should be measured with scaled-down test structures [27]. Then, device sizing can be performed, either graphically or through SPICE. As shown in Fig. 18.15(a), the desired



**Figure 18.15** The configuration of the vertical PNP in the double-diode protection scheme (a) and a graphical method for determining ESD effectiveness using the  $I$ - $V$  characteristics of the vertical PNP and the power supply clamp (b).

ESD current path through the PNP is in parallel with the output driver pull-down. To shunt ESD current through the PNP to  $V_{ss}$  requires an explicit conduction path from  $V_{dd}(x)$  to  $V_{ss}$ . A  $V_{dd}(x)$  power supply clamp, discussed further in Section 18.12, must clamp ESD events across the power rails, completing the current path from the I/O pad to  $V_{ss}$ . For effectiveness, the ESD circuit should limit the voltage across the output driver pull-down below the failure voltage  $V_{fail}$  of the NMOS. Given an emitter current  $I_e$  set by the ESD stress voltage, and the corresponding  $I_b$  flowing through a power supply clamp with a voltage drop  $V_{pc}$ , this constraint becomes

$$V_{be}(I_e) + V_{pc}(I_b) < V_{fail} \quad (18.4)$$

This process is illustrated graphically in Fig. 18.15(b), where the PNP and the power clamp's  $I$ - $V$  characteristics are overlaid. The ESD current determines the voltage  $V_{be}$  across the emitter-base junction of the PNP. At this emitter current, a base current  $I_b$  flows out of the base junction into the power clamp, which drops a corresponding voltage  $V_{pc}$ . The sum of these two voltages should be less than  $V_{fail}$ .

One can trade a smaller PNP for a larger power supply clamp to achieve the same clamping limit. Since one power clamp may serve the power rail tied to many I/O pads, weighting this trade-off strongly toward the power clamp will often result in the smallest area for ESD. To avoid the metal bus resistance contributing significantly to the voltage in the ESD path, it is often advantageous to distribute multiple power clamps along the power bus, rather than simply increasing the size of one [24]. Other constraints may influence the size of the diodes. The ESD diodes can often double as clamps for limiting I/O overshoot and undershoot from reflections in the bus environment.

## 18.12 POWER SUPPLY CLAMPS

Power supply clamps are required both for direct discharges to the power rail itself and for discharges shunted to the power rail through diodes. If the I/O protection uses the grounded-gate NMOS or the NMOS with gate modulation, these protection elements often make good power clamps. To determine effectiveness, one can construct a model

for the ESD and the parallel paths analogous to the I/O case. Dedicated supplies, particularly ones powering large devices, should be handled with care [28].

Three general types of nonsnapback clamps are in widespread use: capacitive clamps, transient-timed clamps, and power-supply referenced clamps. The capacitive clamp relies on the on-chip capacitance of the power supply itself to limit the voltage during ESD. A capacitive ESD source  $C_{ESD}$  charged to  $V_{ESD}$  and discharged onto a supply with capacitance  $C_{vdd}$  will cause the power supply to charge to

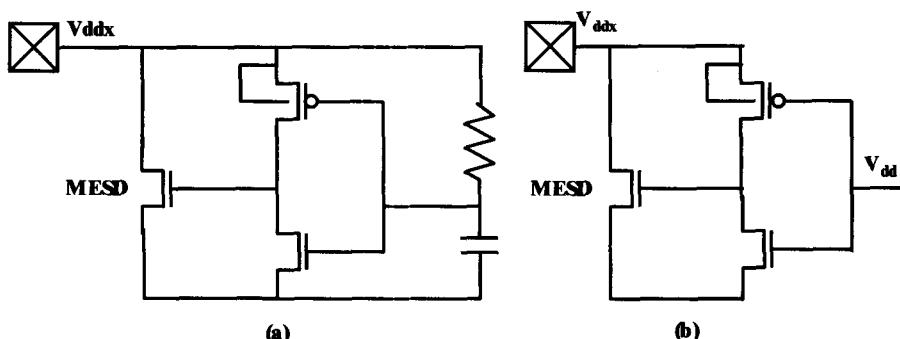
$$V := \frac{V_{ESD} C_{ESD}}{C_{ESD} + C_{vdd}} \quad (18.5)$$

Transient-timed clamps use the transient rise of the supply along with a timing circuit, usually an  $RC$  network, to clamp the ESD event [29], [30]. In the example of Fig. 18.16(a), the  $RC$  timer drives the gate of the NMOS clamping device MESD through an inverter. The device MESD can either operate in snapback or can be sized to handle the ESD current through channel conduction alone.

The power-supply referenced clamp of Fig. 18.16(b) works in the same fashion as the  $V_{dd}$ -referenced clamp described in Section 18.11.2. It uses the voltage level on an independent source to distinguish between ESD events and normal operation. During ESD, capacitance between the reference supply ( $V_{dd}$ ) and  $V_{ss}$  keeps the reference supply near  $V_{ss}$ , allowing the clamp to turn on. During normal operation, the voltage difference between the reference supply and  $V_{ss}$  keeps the clamp off. For multiple supplies, clamps can be placed on each and cross-referenced to one another [24]. For any protection scheme using supply-referenced clamps, it is important to ensure that the pin under protection and its reference are truly independent by checking for parasitic leakage paths between them. There should also be enough capacitance on the reference supply to overcome tester-induced coupling [24].

### 18.13 CDM CONSIDERATIONS

The high-current, short-duration CDM event requires additional design considerations. CDM usually creates a failure at the gate oxide of the input receiver when it is directly



**Figure 18.16** Power supply clamps: (a) a transient power supply clamp using an  $RC$  timer; (b) a power-supply referenced clamp.

connected to the pad. The ESD protection network must limit the voltage across the input receiver during a CDM event. At CDM currents, the voltage at the pad may rise well above the level expected from HBM. Since the duration of the CDM pulse is short, the large clamp at the pad can often tolerate this high current without failing. However, the input receiver's gate oxide may not. The secondary clamp, constructed as a scaled-down version of the primary clamp and shown in Fig. 18.9, limits the voltage at the input receiver. The resistor  $R_{in}$  drops the high voltage from the pad. As long as the voltage across the secondary clamp is smaller than the receiver's breakdown voltage on the CDM time scale, the receiver's protection will be effective. With enough understanding of the package and the protection structures, one can simulate CDM events to verify the ESD network's performance [31].

CDM's complexity demands particular care in the I/O design and its layout. The large CDM current can generate large voltage drops across the resistance of a metal bus. Cases when the primary and secondary clamp cannot be placed near one another and share the same power bus are extremely vulnerable to CDM failure [32], [33]. Furthermore, other gate oxides at the pad may be sensitive to CDM damage [33].

## REFERENCES

- [1] R. Senthinathan, et al., "Application Specific CMOS Output Driver Circuit Design Techniques to Reduce Simultaneous Switching Noise," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 12, pp. 1383–1388, Dec. 1993.
- [2] Hussein Hanafi, et al., "Design and Characterization of a CMOS Off-Chip Driver/Receiver with Reduced Power-Supply Disturbance," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 5, pp. 783–790, May 1992.
- [3] Will C. Gubbels, et al., "A 40-ns/100pF Low-Power Full-CMOS 256 K ( $32 \times 8$ ) SRAM," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 5, pp. 741–746, Oct. 1987.
- [4] Howard L. Kalter, et al., "A 50-ns 16Mb DRAM with a 10-ns Data Rate and On-Chip ECC," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1118–1125, Oct. 1990.
- [5] Ernestina Chioffi, et al., "High-Speed, Low-Switching Noise CMOS Memory Data Output Buffer," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 11, pp. 1359–1364, Nov. 1994.
- [6] D. Dobberpuhl, et al., "A 200-MHz 64b Dual-Issue Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, p. 1555, Nov. 1992.
- [7] B. Gieseke, et al., "A 600 Mhz Superscalar RISC Microprocessor with Out-of-Order Execution," *ISSCC Dig. Tech. Papers*, pp. 176–177, Feb. 1997.
- [8] J. Montanaro, et al., "A 160-MHz, 32-b, 0.5 W CMOS Risc Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, p. 1703, Nov. 1996.
- [9] Hector Sanchez, et al., "A Versatile 3.3V/2.5V/1.8V CMOS I/O Driver Built in a  $0.2\mu m$  3.5 nm Tox 1.8V Technology," *Digest of Technical Papers, 1999 IEEE International Solid-State Circuits Conference*, ISSCC99, Paper TP 16.2, p. 276.
- [10] P. Magnusson, et al., *Transmission Lines and Wave Propagation*, 3rd. Edition. CRC Press, New York, 1992.
- [11] W. Dally and J. Poulton, *Digital Systems Engineering*. Cambridge University Press, New York, 1998.
- [12] Thomas F. Knight and Alexander Krymn, "A Self Terminating Low-Voltage Swing CMOS Output Driver," *IEEE Journal of Solid-State Circuits*, vol. SC-23, no. 2, pp. 457–463, April 1988.

- [13] Thaddeus J. Gabaraa and Scott C. Knauer, "Digitally Adjustable Resistors in CMOS for High-Performance Applications," *IEEE Journal of Solid-State Circuits*, vol. SC-27, no. 8, pp. 457–463, Aug. 1992.
- [14] Mark Horowitz, et al., "High-Speed Electrical Signaling: Overview and Limitations," *IEEE Micro*, pp. 12–24. Jan.–Feb. 1998.
- [15] A. Fiedler, et al., "A 1.0625 Gbps Transceiver with 2 $\times$  , Oversampling and Transmit Signal Pre-Emphasis," *Proceedings of the IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 238–239, 1997.
- [16] W. J. Dalley et al., "Transmitter Equalization for 4-Gbps Signaling," *IEEE Micro*, vol. 17, no. 1, pp. 48–56. Jan.–Feb. 1997.
- [17] A. Amerasekera and C. Duvvury, *ESD in Silicon Integrated Circuits*. John Wiley and Sons, New York, 1995.
- [18] D. Wilson, H. Domingos, and M. Hassan, "Electrical Overstress in NMOS Silicided Devices," in *EOS/ESD Symp. Proc.*, pp. 265–273, 1987.
- [19] D. Krakauer and K. Mistry, "ESD Protection in a 3.3 V Sub-micron Silicided CMOS Technology," in *EOS/ESD Symp. Proc.*, pp. 250–257, 1992.
- [20] W. R. Anderson, "Circuit and Process Design Considerations for ESD Protection in Advanced CMOS Processes," *Microelectronics and Reliability*, vol. 37, pp. 1087–1103, 1997.
- [21] G. Krieger and P. Niles, "Diffused Resistors Characteristics at High Current Density Levels—Analysis and Applications," *IEEE Trans. Electron Devices*, vol. 36, pp. 416–423, 1989.
- [22] T. L. Polgreen and A. Chatterjee, "Improving the ESD Failure Threshold of Silicided n-MOS Output Transistors by Ensuring Uniform Current Flow," in *EOS/ESD Symp. Proc.*, pp. 167–174, 1989.
- [23] K. -L. Chen, "The Effects of Interconnect Process and Snapback Voltage on the ESD Failure Threshold of NMOS Transistors," *IEEE Trans. Electron Devices*, vol. 35, pp. 2140–2150, 1988.
- [24] W. R. Anderson, J. J. Montanaro, and N. J. Howorth, "Cross-referenced ESD Protection for Power Supplies," in *EOS/ESD Symp. Proc.*, pp. 86–95, 1998.
- [25] C. Duvvury and C. Diaz, "Dynamic Gate-coupled NMOS for Efficient Output ESD Protection," in *Proc. Int. Reliability Phys. Symp.*, pp. 141–150, 1992.
- [26] S. Ramaswamy, C. Duvvury, and S. -M. Kang, "EOS/ESD Reliability of Deep Sub-micron NMOS Protection Devices," in *Proc. Int. Reliability Phys. Symp.*, pp. 284–291, 1995.
- [27] "Test Structures for Benchmarking the Electrostatic Discharge (ESD) Robustness of CMOS Technologies," *SEMATECH Technology Transfer Document 98013452A-TR*, May 1998.
- [28] C. Duvvury, R. Rountree, and O. Adams, "Internal Chip ESD Phenomena Beyond the Protection Circuit," in *Proc. Int. Reliability Phys. Symp.*, pp. 19–25, 1988.
- [29] R. Merril and E. Issaq, "ESD Design Methodology," in *EOS/ESD Symp. Proc.*, pp. 233–237, 1993.
- [30] E. Worley, R. Gupta, B. Jones R. Kjar, C. Nguyen, and M. Tennyson, "Sub-micron Chip ESD Protection Schemes Which Avoid Avalanche Junctions," in *EOS/ESD Symp. Proc.*, pp. 13–20, 1995.
- [31] S. G. Beebe, "Simulation of Complete CMOS I/O Circuit Response to CDM Stress," in *EOS/ESD Symp. Proc.*, pp. 259–270, 1998.
- [32] T. Maloney, "Designing MOS Inputs and Outputs to Avoid Oxide Failure in the Charged Device Model," in *EOS/ESD Symp. Proc.*, pp. 220–227, 1988.
- [33] S. Dabral and T. J. Maloney, *Basic ESD and I/O Design*. Wiley-Interscience, New York, 1999.

# HIGH-SPEED INTER-CHIP SIGNALING

Stefanos Sidiropoulos

*Rambus, Inc.*

Chih-Kong Ken Yang

*UCLA*

Mark Horowitz

*Stanford*

Advances in IC fabrication technology coupled with aggressive circuit design have led to an exponential growth in the speed and integration levels of digital ICs. This has in turn created a great demand for high-speed links and busses to interconnect these chips. Gbit/s-scale I/Os are being adopted for memory busses [22], peripheral connections [31], and multiprocessor interconnection networks [12]. This chapter looks at these high-speed links and discusses important design issues and current design techniques.

The design of a link would be very simple if only the speed of light were infinite,<sup>1</sup> since then an inverter would be a perfect driver and receiver. Finite speed of light, however, makes the design much more difficult. The link electronics must deal with the time delay of the information traveling down the wire and the signal energy stored on the wire. (The power driven into the wire takes time to reach the load.) To deal with these issues, a high-speed link has four parts: a transmitter that converts the bits to an electrical signal capable of propagating on a long wire, the long wire itself, a receiver that converts the signal energy of the wire back to bits, and a timing-recovery circuit that compensates for the time delay of the wire. Figure 19.1 illustrates these components.

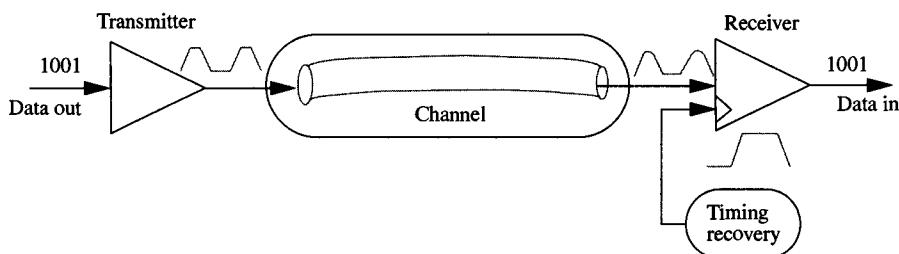


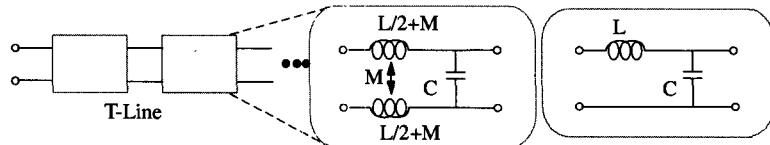
Figure 19.1 Signaling system components.

<sup>1</sup> For most IC circuit design (and most of the designs in this book), this assumption is in fact made. Short wires are modeled as a ‘node’ and only have lumped capacitance, and long wires are modeled by a distributed RC network. This approximation is valid, since the time of flight of the signal on the wire is very small compared to the frequency of interest (i.e., the circuit is much smaller than the wavelengths of interest). To model finite signal velocity, the wire must be a distributed RLC network, as described in the next section.

The next section briefly reviews the electrical issues associated with transmission lines, the common name for these long wires, and describes how different wire configurations effect the noise in the system. Section 19.2 discusses metrics often used to assess the performance of I/O interfaces and describes how the timing and voltage margins are used to measure the robustness of a link. Using these metrics, we will then examine each link component, beginning with the transmitter (Section 19.3), continuing with the receiver (Section 19.4) and the timing-recovery circuits (Section 19.5), and ending with the limitations caused by the transmission line, and what can be done about them in Section 19.6.

## 19.1 TRANSMISSION LINES

When the transmitter starts driving a transmission line, it launches an electromagnetic wave. This wave can only ‘see’ the initial part of the transmission line; it has no information about the final load or even other segments of the wire. So, the geometry and materials of this local segment set the ratio of the signal’s voltage to its current. This ratio is called the line impedance since it has the same dimensions ( $V/I$ ) as resistance. An ideal transmission line can be modeled as a distributed LC circuit, where the inductance and capacitance model the stored magnetic and electric energy of the wave. The impedance of the line is set by  $z = \sqrt{L/C}$  and signals propagate with a velocity of  $v = 1/\sqrt{LC}$  where  $L$  and  $C$  are the inductance and capacitance per unit length. A simple model of a transmission line is shown in Fig. 19.2 where the distributed line is approximated by a number of LC sections.



**Figure 19.2** Simple transmission-line model. The first model shows the inductance in both the signal and return paths with strong mutual coupling between the two. Since the current in the return is always equal and opposite to the signal current, the rightmost model is equivalent for modeling the signal difference between the terminals.

### 19.1.1 Image Currents

A critical property of transmission lines is that they are intrinsically differential structures, i.e., each port really has two terminals, the signal and its return. When one launches a current into the transmission line, an equal and opposite current, called the image current, starts to flow out the return. The rightmost circuit in Fig. 19.2 only models the differential signal between the pins of each port and not any common-mode signal; the two return pins are not actually shorted together. Since the return is often connected to an AC ground, many designers forget about the image current and the return path, but return path errors are the most common causes of link noise.

Some transmission line environments, such as coaxial cables, provide good signal returns. Printed circuit boards (PCBs) can also be built to provide a good transmission-line environment by placing reference planes (ground or  $V_{DD}$ ) under each wiring layer, forming microstrip lines.<sup>2</sup> Absence of an explicit signal return in the vicinity of the signal trace will result in the image current flowing on other signal traces. This coupling between signals gives rise to crosstalk and is modeled by coupling capacitance and mutual inductance between the wires. The worst transmission-line environment often exists in IC packages. Since common packages do not employ ground planes and the connections for the image currents are far from the signals, excessive coupling (crosstalk) can arise between the signal lines.<sup>3</sup>

Crosstalk creates noise on the victim wire that is proportional to the signal on the aggressor wire. Since coupling is through a frequency-dependent reactance ( $\omega L$  and  $1/\omega C$ ), the proportionality constant depends on the bandwidth of the signals. As will be discussed in Sect. 19.3.2, signals with energy at high frequencies (i.e., fast edge rates) experience greater amounts of coupling, creating the need to limit the transmitted signal bandwidth.

### 19.1.2 Reflections

Another source of proportional noise is impedance mismatch in the transmission line which causes reflections. The long wires that the transmitter must drive often consist of a number of different sections—a wire on the package connecting the transmitter chip and the printed circuit board (PCB), a trace on the PCB, possibly connectors and a coaxial or twisted-pair cable between boards, a trace on the receiving PCB, and another package wire to connect to the receiver chip. Each connection between transmission line segments must satisfy both constraints of conservation of energy and continuity of voltage: (1) power flowing into the connection has to be equal to the power flowing out of it, and (2) the voltages at the two sides of the connection have to be equal. If the connected impedances are the same, the solution is trivial, and the signal flow continues with the same voltage. If the second segment has a lower impedance, the voltage in this segment has to be lower (equal or higher voltage would mean that the connection generates power). So, in order to reduce the voltage and increase the current flowing toward the second segment a negative wave is created and sent back to the transmitter. This way the superposition of the forward and backward waves in the first segment match the current and voltage of the forward traveling wave in the second segment. Conversely, if the impedance of the second segment is larger, the voltage in this segment is larger, and to balance the voltage and reduce the current, the reflected wave is positive.<sup>4</sup> These constraints lead to the familiar equation for the reflection

<sup>2</sup> However, some thought about the image currents must be given in PCB layout, since when a signal line changes wiring levels through a via the image current must change levels. If a high-speed bus changes levels, the two image planes must be connected or well bypassed to allow the image current to change levels as well. If the image planes are not bypassed, AC noise will appear between the two return planes. This noise will then change the voltage difference between the signal and the plane on the second level, and will look like noise at the other end of the signal trace.

<sup>3</sup> Even with advanced flip-chip, ball-grid packages, traces are needed to fan out the bumps and balls to vias on the package and board. Coupling in these traces can still be an issue.

<sup>4</sup> Even in cases where the reflected waves are of reasonable amplitude, e.g., 20%, most of the signal power continues down the line (reflected power is only 4% in this case). Thus even if the segments are not at the right impedance, the effect on the signal amplitude at the end of line is usually small.

size

$$V_{\text{reflect}} = \Gamma V_{\text{in}} = \frac{Z_{\text{out}} - Z_{\text{in}}}{Z_{\text{out}} + Z_{\text{in}}} V_{\text{in}}$$

where  $\Gamma$  is the reflection coefficient.

Reflections can be viewed as feedback signals indicating that the current-to-voltage ratio established at the transmitter is incorrect and needs to be adjusted. However, since no information about the transmitter impedance is available at the reflection-generating connection, a few round trips might occur before the system settles. At that point the voltage and current in all segments of the transmission line will be uniform, and the ratio ( $V/I$ ) is set by the final load impedance. If the wire delay is very small compared to the rise time of the signal, this settling happens so quickly that the driver essentially sees the load directly, and reflections can be ignored because they are very brief compared to the signal transition.

Reflections in high-speed links can degrade performance. They are versions of the transmitted signal that arrive at the receiver after spending some time traveling in the wrong direction on the transmission line and hence have a different delay. Since they arrive out of sync with the main signal, the reflections appear as noise at the receiver.<sup>5</sup>

Connecting a transmission line to a driver or receiver is just like connecting it to another segment of a transmission line. If the impedances do not match, a reflection will occur. Sometimes systems are intentionally designed with a mismatched impedance at either the driver or receiver. Systems are built with low-impedance drivers that guarantee a certain voltage swing is driven on the wire. In these systems, the receiver is built to match the impedance of the wire to prevent reflections. Similarly, many systems are built where the transmitter's impedance is matched to the line, and the receiver is left unterminated (high impedance). In these systems, a wave of half the desired signal is launched in the wire (with the driver and transmission line forming a  $Z_0/(Z_{\text{drv}} + Z_0)$  voltage divider). This voltage doubles when it reflects at the open end of the transmission line, presenting a full-swing signal at the receiver. The reflected wave is then absorbed when it reaches the driver. The advantage of this source-terminated system is that it does not dissipate any static power. Once the wave reflects back at the driver, the net current in the line goes to zero.

Systems where either the driver or receiver is intentionally mismatched to the transmission-line impedance require better impedance control of the transmission-line segments than a system where both the transmitter and receiver are terminated. For a reflection to interfere with the received signal, a signal must reflect an even number of times (each reflection changes the signal direction). Thus, the reflection noise interfering with the received signal will be proportional to the product of the two worst reflection coefficients in the system, including the reflections at the driver and the receiver. For reasonable impedance control ( $\pm 10\%$  reflections), double termination reduces the reflection noise at the receiver to around  $\pm 1\%$  from around  $\pm 10\%$  for a system with either end unterminated.

### 19.1.3 Attenuation

An ideal transmission-line segment simply delays the input signal without inducing any distortion of its spectral characteristics. However, real transmission media

<sup>5</sup> Actually, reflected signals are not true noise, because they are predictable and can be compensated out. Systems that 'equalize' the channel do exactly that, by computing the extra delay of the reflection, and subtracting it from the received signal.

attenuate the signal because of the line conductor's resistance and the loss in the dielectric between the conductors [30]. Both loss mechanisms increase with increasing signal frequency, causing the wires to behave as low-pass filters. Longer wires result in more severe filtering. The filtering of the line reduces the amplitude of a bit and spreads the signal out in time, adding a tail that interferes with subsequent bits.

While high-frequency loss has recently become an issue in high-speed link design, the sections on link components will focus on performance limits of the silicon circuits rather than the wires. After discussing the capabilities of the silicon, Section 19.6 describes high-frequency wire loss in more detail and proposes different ways to deal with the problem. Many of these approaches are system solutions that leverage the information presented in the previous sections.

#### 19.1.4 Methods of Signaling

The characteristics of transmission lines, described in the previous section, make it easier to understand the trade-offs between different signaling topologies. In general, three largely independent binary choices characterize most signaling systems: point-to-point link or multi-drop bus, unidirectional or bidirectional link, and single-ended or differential signaling. The latter two choices are a trade-off between wires and proportional noise in the system.

In applications that require low-latency between multiple ports such as DRAMs and backplanes, multidrop busses have been preferred over point-to-point connections. Point-to-point structures require either more routing in a fully connected network or induce larger latency in a daisy-chain connection. However, the electrical characteristics of busses are considerably worse than point-to-point links. Each drop of the bus appears as a short line splitting off from the transmission line, causing the propagating energy to divide and then reflect off the unterminated end. Unless the stub is very short, e.g., in a Rambus system [22], making busses work at high speeds is extremely difficult. So in the subsequent sections of this chapter the discussion on high-performance I/O circuits will be directed toward point-to-point links.

Another cost/performance trade-off is whether the system is unidirectional or bidirectional. For bidirectional links, each pin is attached to both a transmitter and a receiver. Typically, only one transmitter-receiver pair in a link is operating at one time (half duplex), halving the system bandwidth. An alternative is simultaneous bidirectional links (full duplex) which can achieve low pin count and high aggregate bandwidth but requires much more careful design. The receiver must cancel out its own transmitted signal, and a single reflection of the transmitted signal will act as noise seen by its own receiver. Thus, simultaneous bidirectional signaling decreases the number of signaling wires at the cost of an increase in self-generated noise.

A similar trade-off between pin count and performance is present in differential versus single-ended signaling. Differential signaling requires two connections. In the ideal case, each signal serves as the other's return path, and all of the signal energy is stored between the two wires. This situation is the case for a perfect differential medium, such as a twisted pair suspended in space. In actuality, many implementations (e.g., in a PC-board environment) couple the two signals to a common ground plane. The effect is that the net image current in the ground plane is zero. While single-ended signals also use two connections, the return path is often connected to a supply and is shared to reduce pins. The sharing of the return current path increases coupling between

the signals, which increases the noise on these signals. Sections 19.3 and 19.4 cover how the different signaling methods interact with the transmitter and receiver circuit design. Before starting that discussion, we will briefly digress and talk about different methods of measuring both CMOS circuit performance and overall link performance.

## 19.2 LINK PERFORMANCE METRICS

To understand the basic frequency limitations in high-speed links and how they will scale with technology, this section presents a delay metric that, to the first order, uncouples circuit performance from fabrication technology. This metric helps identify the clock frequency as one of the major limitations to simple CMOS link speeds. Next, it presents the ‘eye’ diagram, which overlays the transmitted waveforms of a long series of bits on top of each other to better illustrate the link margins. Using this diagram, it is easy to see how coupling, reflections, and filtering reduce the margin of the link. In addition to the deterministic noise that we have discussed so far, there is also true statistical noise in the system. If the margin after the deterministic noise is not large enough, the true noise will cause random bit errors, leading to a measurable bit error rate (BER) for the link.

### 19.2.1 FO-4 Delay Metric

As technology scales, basic circuit speed improves. Fortunately, the delays of all CMOS digital circuits scale roughly the same way, causing the ratio of a circuit’s delay to the delay of a reference circuit to remain roughly constant. We take advantage of this fact by creating a metric called a fanout-of-four (FO-4) delay. An “FO-4 delay” is the delay of one stage in a chain of inverters, where each of the inverters in the chain drives a capacitive load (fan-out) that is  $4 \times$  larger than its input capacitance. Figure 19.3(a) illustrates the normalized delay of various circuit structures versus technology and voltage scaling, demonstrating that the normalized delay of a circuit has less than a 20% variation for a relatively complex circuit structure. Figure 19.3(b) shows the actual FO-4 inverter delay for various technologies. Interestingly, the FO-4 delay for these processes is directly proportional to the channel length. The proportionality constant can be roughly approximated by  $500 \text{ ps}/\mu\text{m}$  at worst-case operating conditions (typical process, 90%  $V_{dd}$ , 125°C).

We can use this metric to look at one important component in any link design: the clock. A clock must be buffered and distributed across the chip. Figure 19.4 shows the effect of driving different pulse widths (expressed in the FO-4 delay metric) through a clock buffer chain with a fan-out of four per stage. As the pulse width shrinks, the amplitude of the clock pulses becomes significantly reduced. The minimum pulse that can propagate through the clock chain is around 3 FO-4 delays. Since a clock is actually a rising pulse followed by a falling pulse, the minimum clock cycle time will be 6 FO-4. Accounting for margins, clocks cycles faster than 8 FO-4 (which is roughly  $2 \times$  the frequency of the clock in Alpha microprocessors) are unlikely. However, this clock limitation leads to relatively low data rates (500 Mb/s in a 0.5  $\mu\text{m}$  technology). High-speed transmitters and receivers will need to transmit more than one bit per clock cycle by using parallel-to-serial and serial-to-parallel converters.

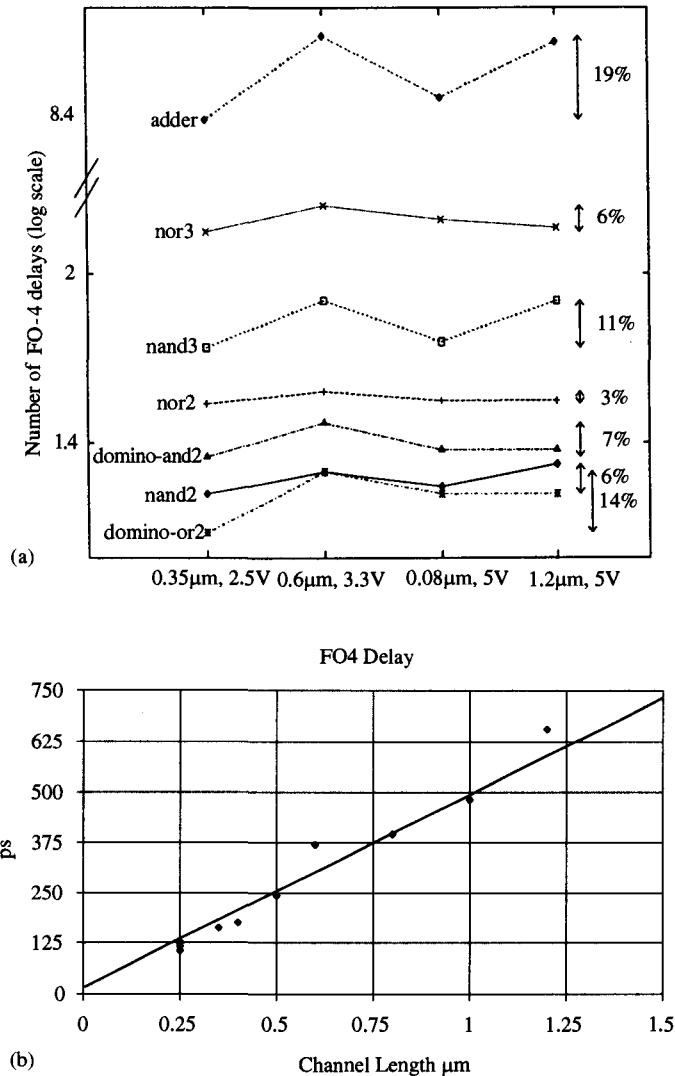
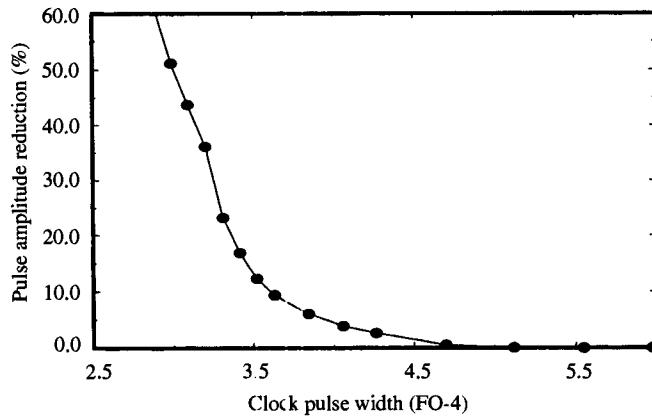


Figure 19.3 Fanout-4 metric performance.

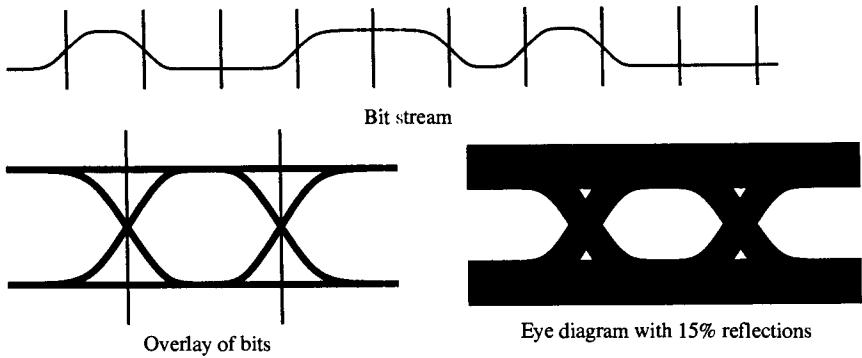
### 19.2.2 Link Margins

To understand how many bits per cycle can be transmitted, we can create an ‘eye’ diagram, a plot that quickly shows the voltage and timing margins of a link. An example of this plot is illustrated in Fig. 19.5. If reflections or other proportional noise are present, the voltage of a “1” and “0” becomes uncertain, since its value depends on the noise. The voltage margin, the difference between the highest “0” and the lowest “1”, becomes smaller as can be seen by the reduction of the eye height. Voltage noise also decreases the eye width because of the signal’s finite slope.

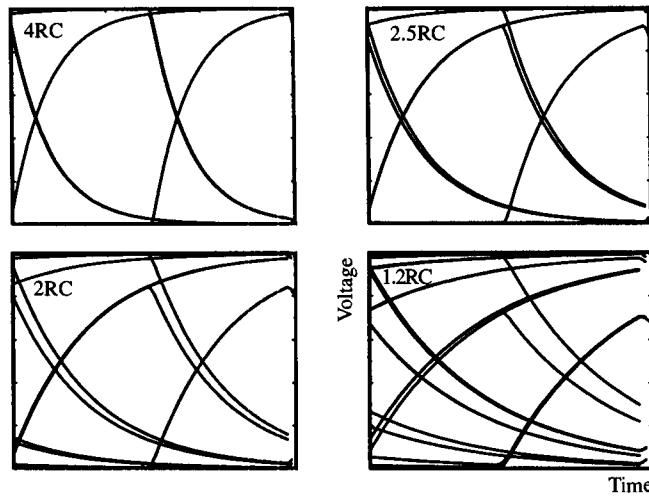
Eye diagrams are also a good way to look at the effect of low-pass filtering on a link. As shown in Fig. 19.6, due to filtering, a reduction in bit time results in a pulse amplitude reduction. Low-pass filtering also causes intersymbol interference (ISI), since the tail of



**Figure 19.4** Pulse amplitude at the end of a chain of four FO-4 inverters as a function of pulse width.



**Figure 19.5** Eye diagram of transmitter output.



**Figure 19.6** Reduction of data eye due to a simple RC low-pass filter. The bit width is measured in number of time constants. The ISI becomes large for bits smaller than 2 RC.

the previous bit affects the starting voltage of the current bit thus introducing uncertainty in the amplitude and reducing the timing margins. The eye closes completely and the signal is undetectable when an isolated pulse is attenuated by a factor of 2 (6 dB).

Generally, eye diagrams are shown for signals at the receiver's input. Since the receiver has internal noise sources, a designer measures the true link margins by adding both timing and voltage offsets at the receiver and then measuring how much can be added before the link starts to fail. Rather than always tracing the complete eye diagram, often one estimates the eye by measuring the voltage and timing margins at the center of the eye.

So far we have described deterministic noise sources which are due to the ISI and reflections of the transmitted signal or coupling from other signals in the system. In addition to this self-generated noise, there is true Gaussian noise as well. While techniques exist to correct for self-generated noise (e.g., equalization that will be discussed in Section 19.6), the random noise is not correctable and sets a limit on system performance. The intrinsic sources of this noise are the random fluctuations due to the inherent thermal and shot noise of the system components. For links operating with small voltage or timing margins, the random noise will occasionally cause the link to fail. To account for this, links often specify a bit error rate (BER) which is the probability that any bit will be in error. For example, a 1 Gbps link with a BER of  $10^{-11}$  will fail on average every 100 seconds or once in  $10^{11}$  bits. The probability of failure is an exponential function of the excess signal margin (signal minus proportional noise) divided by the RMS value of the noise. Since the RMS noise of most links is small (on the order of millivolts) changing the true signal margins by 10–20 mV can change the BER by many orders of magnitude.

The next sections will use these metrics to look at circuit techniques for building high-speed, low-noise transmitters and receivers.

## 19.3 TRANSMITTERS

The main function of the transmitter is to convert bits into a voltage waveform that can be propagated down the channel. Generating a high data-rate signal that has accurate levels and induces small system noise is the key issue in transmitter design. Additional goals in many parallel links is to minimize the time delay through the driver and the dissipated power. To accomplish these goals, we use two techniques: parallelism and configurable hardware with closed loop control. Parallelism overcomes the intrinsic speed limitations of CMOS circuits, and closed loop control overcomes the wide variability of CMOS device characteristics. As we will see later, these same two techniques are used in the receiver and the clock-recovery circuits.

A block diagram of a transmitter is shown in Fig. 19.7. The data bits are serialized in a parallel-to-serial converter, and then driven onto the channel. This section starts by describing the design trade-offs of different drivers. Then, it discusses control loops that can reduce reflections and signal coupling, and maintain accurate output levels. The last part of this section talks about using parallelism to improve the bit rate of the link.

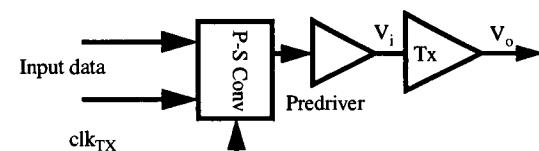


Figure 19.7 Transmitter block diagram.

### 19.3.1 Output Drivers

There is not much variability in the design of transmitter output drivers. They consist of one or two transistors and possibly one resistor. The two main variables are the impedance of the driver and the operating region of the MOS device. A voltage-mode driver consists of two transistors which connect to supplies that set the signal swing. The transistors are sized so they operate in the linear region of their I-V curve. This type of driver can be used as a low-impedance driver. By correctly sizing transistors and possibly adding a series resistor, the driver's source impedance can be designed to match the impedance of the line [Fig. 19.8(a) and (b)]. A special low-voltage supply is often used with this type of driver to reduce the voltage swing and hence the power needed to drive the transmission line. Signal swings under a volt are common, and for these low voltages the pMOS pull-up transistor is replaced by an nMOS device. A current-mode driver [Fig. 19.8(c)] generally contains a single transistor that operates in the saturation region of the I-V curve. This driver can act as a high-impedance driver, or by adding a parallel termination resistance, it can be a matched-impedance driver as well.

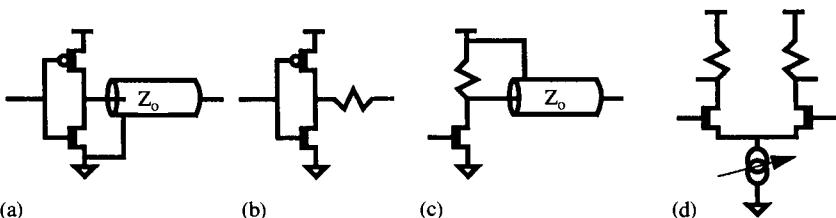


Figure 19.8 Output driver configurations.

Very low-impedance voltage-mode drivers are sometimes built [9] but they are usually not needed. The reason is that they require large transistors to get the low ( $\sim 5\Omega$ ) desired resistance, and this low-impedance drive causes reflections. Most voltage-mode drivers are matched to the line impedance to minimize reflections. This can be done by adjusting the size of the devices to match the line impedance, or by making the impedance of the devices sufficiently smaller and adding a series resistor (e.g.,  $20\Omega$  transistors with a  $30\Omega$  resistor for a  $50\Omega$  line is common practice). Adding the series resistance increases the required transistor size, but improves the impedance matching. This type of design minimizes the voltage across the device which reduces the effects of the nonlinear transistor resistance, and it lowers the sensitivity of the overall impedance to the transistor resistance. Without the series resistance, the nonlinearity of the transistor's resistance makes building a matched driver difficult.

In voltage-mode signaling, noise on the power supply is directly transmitted onto the signal wire. If the signal return also connects to the power supply on the chip, then the noise is common mode and is not seen at the receiver. Unfortunately, this is almost never done. The signal return connection goes to board-level ground and the supply noise gets injected as part of the signal. This noise is caused by the image current flowing through the ground network that connects the chip to the board. To mitigate this problem, designers must carefully avoid large inductive loops in the current return path of the signal or use differential drivers. In addition, since the signal return current flows through either the ground or  $V_{DD}$  of the driver, these two supplies must be very

well bypassed especially near the package. Fortunately the large on-chip bypass capacitance between  $V_{DD}$  and ground helps with this decoupling.

The operation of a current-mode driver is similar, except in this case the voltage across the transistor is larger, and the transistor operates in its saturated current region. For a given current (i.e., swing) a current-mode driver will have a smaller device size than a voltage driver, since the voltage driver must source the current at a much smaller  $V_{DS}$ . The larger voltage across the device results in higher power dissipation. However, the high impedance of the driver isolates the output signal from ground noise. In addition, system design is made easier by referencing the signal only to the positive supply, because the positive supply can be used as the transmission line signal return.

Both voltage-mode and current-mode single-ended drivers suffer from noise injection into the power supply due to changing supply currents and self-induced  $di/dt$  noise. Use of differential drivers can eliminate this problem. Two complementary single-ended drivers are often used. This configuration can minimize noise by keeping the ground current roughly constant over time. A common alternative for current-mode drivers is to use an open-drain differential pair as shown in Fig. 19.8(d). The downside of using a differential pair is the decreased  $V_{GS}$  on the output devices. Larger devices are needed for the same output swing, resulting in higher output capacitance.

### 19.3.2 Impedance, Current, and Slew-Rate Control

Current-mode drivers should have a well-controlled swing, and voltage-mode drivers should have a well-controlled output impedance. Both swing and impedance depend on the device transconductance and operating voltage. Thus, the output swing or impedance will vary with process and operating conditions. In order to achieve robust system operation, a high-speed design must accurately set the swing, impedance, and slew rate of the output drivers. Most links use a control loop to compensate for variations in process and operating conditions. Figure 19.9 illustrates the block diagram of a loop that controls the current of a current-mode driver. The output driver contains a set of

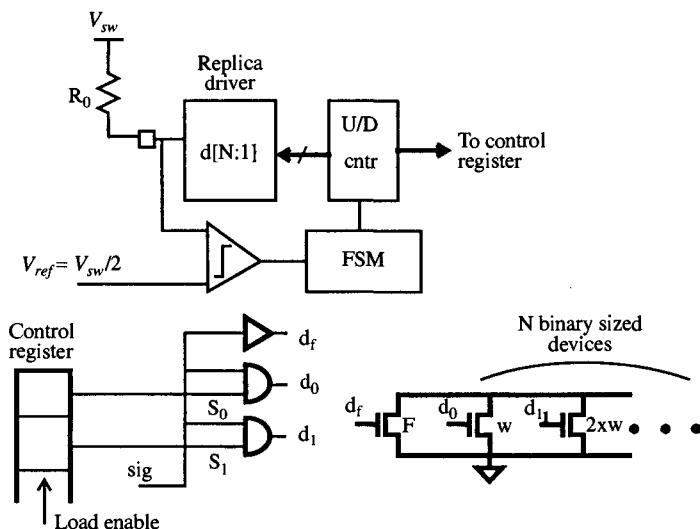
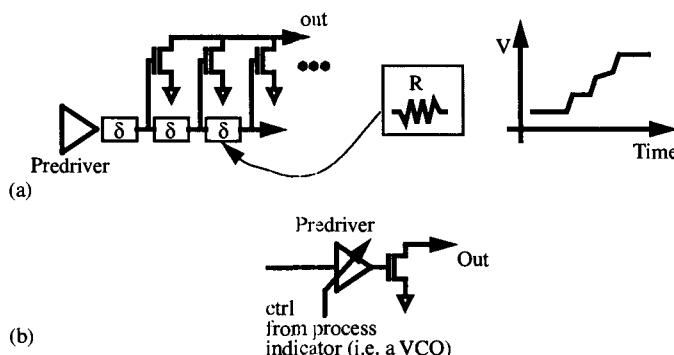


Figure 19.9 Current-control feedback loop.

binary-weighted transistors. A digital control word, stored in a register, sets the number of transistors used by the driver. The control word value is set to the appropriate value by using a negative feedback loop. The loop contains a counter, a dummy output driver terminated to a resistance and a comparator. The dummy output driver and resistance are scaled so that when the output of the dummy driver is  $V_{sw}/2$  the output of a real driver is  $V_{sw}$ . The comparator compares a reference voltage (i.e., half of the voltage swing) to the output of the dummy driver. The counter value is appropriately decreased or increased according to the comparator output. This way the current of the dummy driver is correctly set. The counter output is periodically latched into the global control register, thus ensuring that the swing remains constant during operation. An impedance control loop works in a similar way. In such a system, a current is injected into a dummy output and an external resistance, and a feedback loop adjusts the transistor sizes to make the voltage drop (and hence the resistance) of the two branches the same.<sup>6</sup>

As mentioned earlier, to reduce both coupling and reflection noise, all drivers need to control the high-frequency spectral content of their output. This is commonly done by limiting the slew rate of the driver itself. Variations in process and operating conditions make slew-rate control difficult. If the desired slew rate is achieved for the slowest possible fabrication and operating conditions, then the fastest case can exceed the maximum slew rate. On the other hand, if the fast case meets the slew-rate constraint, then the output bandwidth becomes too slow for the desired data rate, causing ISI in the slow case. Early attempts at slew-rate control tried to find parameters that were inversely correlated with transistor speed. One method divides the driver into multiple segments, and then turn on each segment sequentially as shown in Fig. 19.10(a). Commodity SRAMs use polysilicon resistors between the driver segments to generate the delay. Thin poly or gate oxide increases the RC delay and compensates for faster transistor speeds hence decreasing the variation in output slew rate. More recent methods use an adjustable transition time predriver with explicit feedback as depicted in Fig. 19.10(b). In one implementation, the speed of the process is measured, and the result of this



**Figure 19.10** Slew-rate control using segmented driver (a), or using controlled predriver (b).

<sup>6</sup> A number of people have proposed to directly measure the transmission-line impedance and to adjust the drivers to this impedance [6]. In these systems, a voltage-mode source-terminated driver is used, and the line is open at the receiver. The feedback loop looks at the voltage on the line before the reflection returns. If the voltage is larger than  $V_{sw}/2$ , the impedance is increased; if it is smaller, the impedance of the driver is decreased.

measurement sets the speed of the predriver [24]. A more precise implementation uses a controllable delay element as the predriver. The control voltage of the delay elements is generated by a PLL, whose VCO contains identical delay elements and is locked to the bit-rate clock. In this way, the delay and thus the transition time of the predriver is actively controlled to be a constant fraction of the bit time [44].

The impedance, current, and slew-rate control circuits rely on the ability to measure the I/O performance indirectly using some dummy circuits. The measurement results adjust controls of the real circuits to achieve the desired performance. With abundant transistors and reasonable matching between devices, these techniques work quite well and are commonly used to solve a number of problems in high-speed link design.

### 19.3.3 Maximizing Bit Rate

Having described methods for delivering well-shaped bits onto the transmission line, we now look at what limits the bit rate. The simplest and most commonly used solution to overcome the 6–8 FO-4 clock cycle limitation is to use a 2 : 1 multiplexor that drives a new bit on each phase. Figure 19.11 shows the effect of bit-time on the pulse width of the multiplexor output signal while assuming that an aggressive FO-2 buffer chain drives the clock for the 2 : 1 multiplexor. Although the multiplexor is fast enough for a bit time of 2 FO-4's, the minimum clock period of 8 FO-4 will limit the system to bit widths of 4 FO-4. Since this performance is fast enough for many applications (roughly 2 Gbps in 0.25  $\mu$ m process technology), the 2 : 1 multiplexing architecture is used in most parallel links [22], [31], and [41]. The latency of these systems is quite modest, consisting of about a clock cycle plus the multiplexor and driver latency. Similarly modest is their power dissipation:  $CV^2f$  clock power for the parallel-to-serial converter plus the IV power of the driver.

Achieving shorter bit time creates a dilemma. To get around the clock limitation requires a large fan-in multiplexor, but the speed of a 4 : 1 multiplexor limits the bit time to 3–4 FO-4. To create a faster multiplexor, we need to reduce its output swing and trade gain for increased bandwidth. The highest possible bandwidth is achieved when the multiplexing occurs directly at the output pin where both a low time-constant ( $25\text{--}50\Omega$  impedance) and small swings are present. Figure 19.12 shows the basic idea behind this technique. To achieve parallelism through multiplexing, additional driver legs are connected in parallel. Each driver leg is enabled sequentially by well-timed pulses of a width equal to a bit time. Since in this scheme the current pulses at the output are completely generated by on-chip voltage pulses, the minimum output pulse

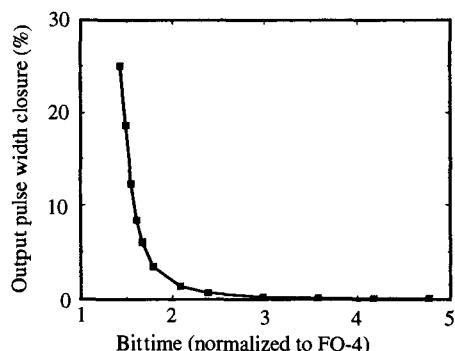
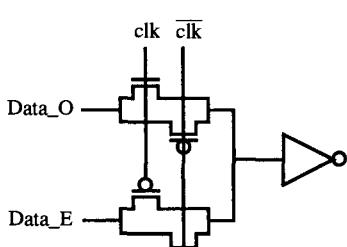


Figure 19.11 Effect of bit time on output pulse width of a simple 2:1 multiplexor.

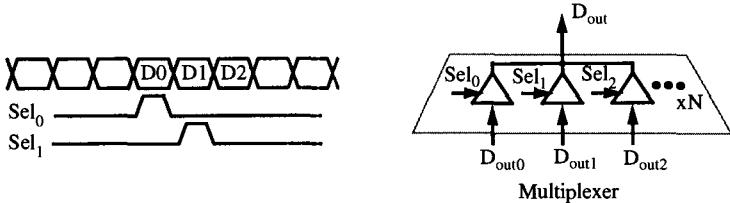


Figure 19.12 Higher bandwidth multiplexing at the output pin.

width is limited again by the minimum pulse width that can be generated on-chip [26]. An improved scheme, illustrated in Fig. 19.13(b) using a grounded-source current-mode output driver, eases that limitation [48]. In this configuration, two control signals, both at the slower on-chip clock rate, are used to generate the output current pulse. In this design, either the transition time of the predriver output or the output bandwidth determine the maximum data rate. Figure 19.13(c) illustrates the amount of pulse time closure with decreasing bit width for an 8:1 multiplexor utilizing this technique. The minimum bit time achievable for a given technology is less than a single FO-4 inverter delay. The cost of this scheme is a more complex clock source, since precise phase-shifted clocks are required, and an increase in latency (measured in bit times). In addition, since the parallel-to-serial conversion is done at the final buffer stage, this design has larger area and clock power than a simple 2:1 multiplexor design. The highest speed links use this technique, and achieve bit times that are on the order of 1 FO-4.

All of these approaches create transmitters with bit rates that track technology. The rising frequency of these systems will place increasingly difficult constraints on the design of the packaging to ensure that coupling, reflection, and attenuation does not limit performance.

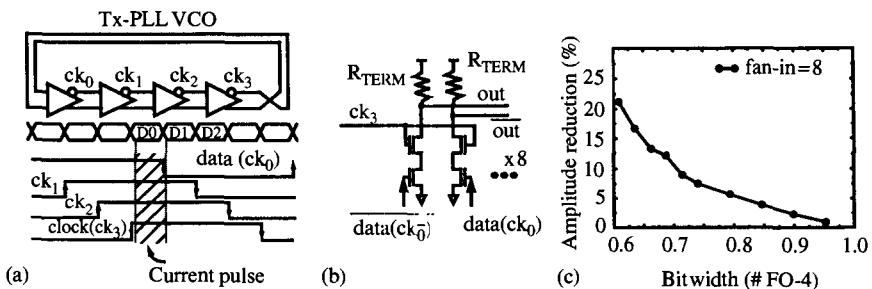


Figure 19.13 High-bandwidth output-driver implementation using overlapping pulses and grounded source driver. Part (a) shows the overall timing, with the shaded portion marking the desired bit time. Part (b) shows the circuit diagram of the output driver, but the clock qualification of the data signals is not shown. Part (c) plots the performance of this system.

## 19.4 RECEIVERS

The design of the receiver in many ways parallels the design of the transmitter. One of the critical issues is reducing received noise, which, like the transmitter, requires atten-

tion to return currents and ensuring that the bandwidth of the receiver matches the bit rate. In addition, input demultiplexing is used to allow these systems to overcome clock and amplifier delay constraints. Finally, in recent designs closed loop control is being used to nullify offsets in the input amplifier or the sample time, and to adjust the bandwidth of the input receiver. Since the performance of the receiver really depends on the quality of the input signal, we start there.

### 19.4.1 Input Noise

The main sources of noise for a receiver are mutual coupling between inputs and differences between chip ground and board ground. Differences between these voltages arise because of the large image currents that flow through the supply pins to support the output drivers.<sup>7</sup> Unless care is taken, some of this ground noise also appears at the input to the receiver.

Differential signaling is a good way to minimize the effects of this noise. If both signals are routed together, ground noise will couple similarly into both signals, changing the common mode but not the differential signal. Differential signals also reduce the effect of signal coupling. Since the signal and its complement are always next to each other, any mutual coupling from one signal is at least partially compensated by coupling from its complement. As a result, with differential signaling the required signal amplitude can be quite low, and links can operate successfully with signals on the order of tens of mV [44].

Conceptually, a transmission line is a differential system, and the same performance can be achieved in single-ended links. The key is that the signal return path needs to be brought onto the chip next to the signal, and connected to the reference supply, e.g., chip  $V_{DD}$ . With this connection the difference between chip  $V_{DD}$  and board  $V_{DD}$  is dropped across the large common-mode inductance of the link, and there is little noise injected between the signal and the on-chip  $V_{DD}$ . This connection makes reference generation easier since the signal is now referenced to the on-chip supply, but is rarely done in real single-ended link design since it takes as many package pins as a differential design, and it requires the signal returns to be somewhat isolated from each other.

In a PCB, the returns will be shorted together on the board and connected to the board  $V_{DD}$  making the ideal connection impossible to achieve. In this case, we need to generate a reference based on the board  $V_{DD}$ . This external reference generation is used in most single-ended link designs, but has a number of difficulties. It would work well if its coupling to the chip  $V_{DD}$  and ground matched the coupling on an input pin. Unfortunately, in nearly all systems, the reference pin is shared among a large number of inputs, and has a larger capacitance coupling to the chip supplies than a normal input [22], [31], [36], [41]. This difference in coupling means that a range of high-frequency on-chip ground noise will couple into the reference line more than the input pins, causing high-frequency noise to appear at the input of each receiver. The noise manifests itself in the time domain in the form of reference voltage “spikes.” A high-bandwidth receiver might sample the input during one of the spikes, causing an error. A receiving amplifier with limited bandwidth can effectively filter some of this noise, increasing voltage margins. Optimal noise rejection can be achieved if the amplifier front end is replaced

<sup>7</sup> On-chip bypass capacitance only reduces chip  $V_{DD}$  to chip ground noise, and has no effect on the noise between chip ground and board ground.

by an integrator which averages the polarity of the input signal during the valid-bit time [37].<sup>8</sup> Even with filtering, this additional noise coupling means that in real systems single-ended signals need much larger swings than differential signals to work correctly.

The noise situation is made even worse for simultaneous bidirectional signaling where the system needs to subtract its transmitted waveform to generate the received waveform. The subtraction is done by generating a local reference that tracks the value being transmitted. The local reference needs to match the timing and the voltage of the transmitter. Neither is easy, since the transmitted waveform depends on the effective impedance through the package, which is the least-controlled section of the transmission line. Mismatches between the reference and the output appear as noise at the input to the receiver, making this signaling scheme require the largest swings, and benefit the most from a bandwidth-limited receiver.

### 19.4.2 Receiver Design

Almost all receivers use at least a two-stage design, where the first stage does the signal conditioning and usually the sampling function, and the second stage provides the gain and latching function. To condition the signal, the first stage generally provides the necessary filtering, translates the input levels from anywhere in the allowable input common-mode range to a fixed output common-mode voltage, and converts the single-ended input into a differential signal for the second stage. Since almost all high-speed links use a demultiplexed receiver architecture, the sampling is done either at the first stage or at the input to the following stage. Given all these tasks, it is not surprising that high voltage gain is not required, and is often close to unity.

To get good common-mode noise rejection, the input stage is generally a differential pair connected to some load devices. The gates of the MOS devices are connected to the input pin and the reference, or to the differential inputs. The common-mode range sets whether the input pair is nMOS (inputs near  $V_{DD}$ ) or pMOS (inputs near ground). Since the stage is being used in a demultiplexing receiver, one can eliminate all ISI issues introduced by the receiver by shorting the differential pair outputs before each sample period, thus eliminating any effect of the previous bit. In most systems the load devices are either resistors or MOS devices operated in their linear region, so the stage forms a simple differential amplifier. The bandwidth of the amplifier is set by the RC time constant of the loads.

To build an integrating receiver, the load devices are replaced with switches and capacitors, and the differential-pair current is integrated over the bit time to see whether the input was mostly ‘one’ or mostly ‘zero’ [37]. Another approach to noise filtering is to change the current in the differential pair along with the effective resistance of the load device to make the bandwidth of the amplifier track the bit time. Similar to transmitter slew-rate control, one can leverage the fact that buffers in the clock generator are adjusted to have a bandwidth related to the bit rate [44].

The gain required by the system is provided by the second stage, which is a clocked regenerative amplifier. This topology has the best speed/power/area trade-off. A common design for this stage is a StrongARM latch [7]. Given the low gain of the first-stage amplifier and the high offsets of regenerative amplifiers, the input-referred offset of this

<sup>8</sup> In case the input signal exhibits increased timing uncertainty, the signals near the edge of a bit are very noisy. A “windowed” integrator can be employed to integrate the bit value only around a portion of the bit time when the signal is least likely to change.

topology can be tens of millivolts, especially if small input devices are used. Again, closed-loop feedback can be used to reduce this problem. Devices are added that can create an offset in either the first or second amplifier. The controls to these devices can be set during an initial open-loop calibration step or continuously adjusted via a feedback loop that averages the received data [15].

Note that in the design of the receiver not much attention is paid to the control of the effective sample point (set-up time) of the circuit. Instead, the timing-recovery loop uses a receiver in its feedback to compensate for the set-up time. It is critical that for any given process corner the set-up-to-hold window be made as small as possible, since this represents the uncertainty in the sampling point due to input parameters (edge rate, signal swing, common-mode voltage) for which the control loop cannot correct.

The latency of the receiver is really the delay through the two-stage receiver plus the resynchronization delay which realigns the data with the system clock of the receiver chip. The delay of the amplifiers varies, but is roughly one bit time for the first stage, plus 4 FO-4 for the second stage. The 4 FO-4 delay is roughly the time needed for a regenerative amplifier to take a small-swing input to a full-swing output. Power dissipation in the receiver is primarily due to the DC current of the first-stage amplifier and the clock power for the second stage.

#### 19.4.3 Demultiplexing Issues

Using multiple sampler amplifier pairs in parallel greatly reduces the bandwidth requirements on the amplifier [25]. The most common form of this demultiplexing employed in low-latency parallel systems [22] is 1:2 demultiplexing. Two receivers are used on each input, one of them triggered by the positive and one by the negative edge of the clock. Each receiver has a half-cycle for the first stage to sample (while resetting the amplifier) while another half-cycle (one full bit time) can be allocated for the regenerative amplifier to resolve the sampled value. The bit width achievable by this simple demultiplexing structure is determined by the minimum operating cycle time of the regenerative amplifier and is on the order of 4 FO-4 delays, matching the clock limitations [22], [36].

A higher degree of demultiplexing can be employed by using well-controlled clock spacing as shown in Fig. 19.14(a). In this system, the demultiplexing occurs

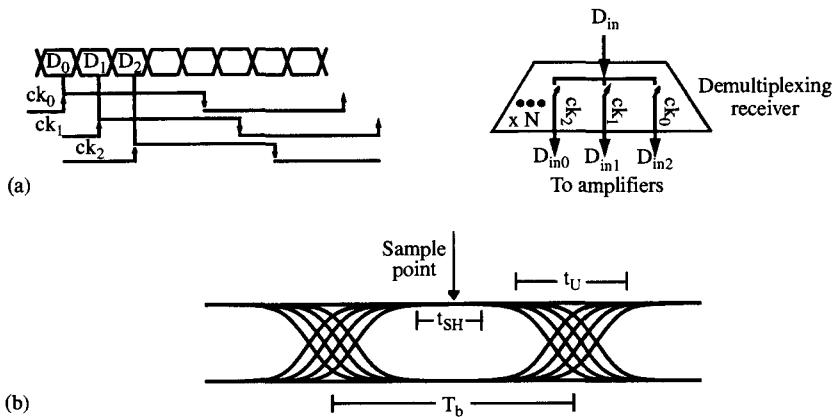


Figure 19.14 Higher-order demultiplexing (a) and timing margins (b).

at the input sampling switches [20], [21], [25], [47], which are often simple nMOS pass gates. The fundamental bandwidth limitation on this parallel architecture is imposed by the ability to generate the finely spaced clock phases and the sampling bandwidth. In most links, we intentionally decrease the sampling bandwidth by adding a bandwidth-limited stage before the samplers to minimize the effects of noise. The fundamental limitation of CMOS switches is less than 0.1 FO-4 [18]. Even with simple nMOS sampler design, the aperture is less than 0.5 FO-4 delays and will not limit the performance of these links. What does limit the performance of these high-speed links is the quality of the sampling clocks needed to maintain reasonable timing margins in the link. The precision and control of these clocks are described in the next section.

## 19.5 CLOCK GENERATION

Almost all high-speed links use phase-locked loops (PLLs) to generate their internal clocks. Generating clocks for the transmitter is often easy, since there is no need to precisely align the phase of the clock(s) to an external reference clock. However, generating clocks for the receiver is more difficult, since the chip must adjust the placement of the clock(s) so that the receiver samples each bit in the middle of the data eye. Consequently, the receive clocks must track the transmitter clock's phase and frequency. There are two independent variables that set the timing relationship between the transmitter and receiver: one is whether there is a single global frequency source that is distributed to all the link components, and the other is whether a clock is sent in-phase with the data. The easiest link designs are the ones where a single clock acts as a frequency source for the entire system. The disadvantage of this approach is that the global clock is a single point of failure for the entire system, and if the links are long, one still needs to deliver the clock to all parts in the system. The alternative is to use separate but nominally equal frequency sources for each end of the link. In reality, the frequencies in these systems match to a few parts per thousand or better. The timing-recovery circuit must not only adjust phase but also compensate for the small frequency difference.<sup>9</sup>

The two key metrics for judging the quality of the clock generation are phase offset, the DC error in the placement of the clock, and jitter, the AC component of this error. After exploring basic PLL operation and the causes of offset and jitter, this section looks at a dual-loop PLL, a promising architecture that allows a designer to more easily align the receive clock in a wide variety of system environments.

### 19.5.1 PLLs and Phase Noise

Figure 19.15 shows the two basic approaches to PLLs: VCO-based phase-locked loops (PLLs), and delay-line-based phase-locked loops or delay-locked loops (DLLs). The basic idea behind the operation of these two circuits is quite similar: using a control voltage ( $V_C$ ), they both try to drive the phase of their periodic output signal (clk) to have a fixed relationship with the phase of their input signal (ref-clk). VCO-based PLLs are more flexible, but are more complicated than those based on delay lines. Since a VCO changes the phase of its output clock by integrating the controllable VCO frequency, a VCO-based PLL is inherently a higher-order control system that creates some design

<sup>9</sup> Additional logic must handle the difference in data rate between the two chips.

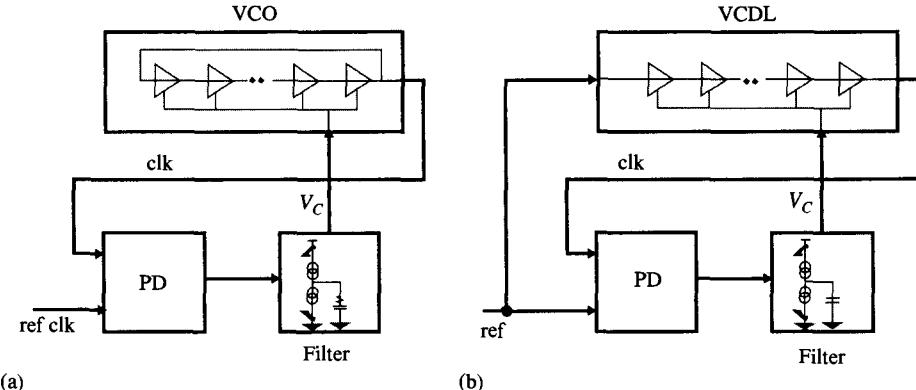


Figure 19.15 PLL and DLL loop topologies.

challenges [13], [34]. On the other hand, since many applications supply a correct frequency clock [19], [27], DLLs directly control the output phase by using a voltage-controlled delay line (VCDL) which delays its input clock by a controllable time delay to generate the output clock. The VCDL in this system is simply a delay gain element, resulting in a first-order (unconditionally stable) control system that is much easier to design.

Both the VCO and VCDL use buffer elements with an additional input that controls their delay. The feedback loop uses this control to adjust the phase of the outputs. While much effort helps to reduce the effect that  $V_{DD}$  or  $V_{substrate}$  has on the delay, some sensitivity remains. Thus noise on these supplies will cause variations in the delay of the buffers and needs to be corrected by the loop. Unfortunately, like all feedback systems, the control loop has finite bandwidth and can only correct for errors below the control bandwidth. High-frequency errors are uncorrected and cause high-frequency phase noise, or jitter, on the output clocks.

In many systems, the jitter from the core PLL is small compared to the jitter caused by the final clock buffer chain. The final clock buffers are generally CMOS inverters, with no special noise isolation. For these structures, a 1% change in supply yields roughly a 1% change in delay, which is more than an order of magnitude larger than the supply sensitivity of a well-designed buffer. Unless the delay of the core delay line is  $10 \times$  the clock buffer delay, the clock buffer chain will dominate the on-chip jitter. For a total buffer delay of about 6 FO-4, and 5% high-frequency noise, the peak-to-peak jitter of the clock will be around 0.3 FO-4 delays. The jitter for a well-designed clock with no supply noise, arising from more fundamental noise sources, is roughly 1.5% of the cycle time, or 0.12 FO-4 for a clock cycle of 8 FO-4 delays.

For the high-order multiplexing and demultiplexing systems described earlier, precisely spaced clock phases are used to determine the bit-width. Several techniques can be used to generate these phases. The simplest is to use a ring oscillator (or a delay line with its input phase locked to its output phase) and tap the output of each of its stages. For example, a four-stage oscillator employing differential stages can generate eight edges evenly spaced in the  $0\text{--}360^\circ$  interval. If the oscillator operates with a period of 8 FO-4 delay, the phase spacing is 1 FO-4. For even finer phase spacing than a single buffer delay, phase interpolators can be used to generate a clock edge occurring halfway between any two output phases [46], [47].

In addition to the high-frequency noise, there are DC errors in the positioning of the clock edges. These errors arise from mismatches between balanced paths or elements and are a concern in systems that transmit more than one bit per cycle. To address these matching issues many link designers use closed loop control. In a 2:1 multiplexing system where the bit time is a half-cycle, the bit time is controlled by the time between the rising and falling edges of the clock. To match the time of each half-cycle, duty-cycle correctors can be added to compensate for small errors caused by fabrication mismatches [27].

In order to align the clock to the proper position, the timing of the data can be extracted either from a clock that is bundled with the data or from the transitions in the data signal itself. If the data signal is used, some coding is necessary to guarantee that enough transitions exist to maintain phase lock. Different phase-detector designs can lock the PLL output either directly to the middle of the bit time, the optimal sample point, or to the data transitions [1]. If the loop locks to the middle of the data transition region, the true sample clock is generated by shifting the clock phase by half a bit time.

A common challenge in phase detection for both DLL- and PLL-based timing recovery is that the circuit has to cancel out the set-up time of the input receiver. As discussed in Section 19.4.2, this usually implies using a receiver replica as the phase detector, resulting in a single binary output, rather than an analog value of the phase error. Using this phase detector creates a nonlinear, or bang-bang control system that locks the phase to the middle of the data transitions and causes difficulties for both VCO and VCDL-based PLLs. For a VCO-based PLL, shifting the clock by half a bit time is easily done by tapping the appropriate phase from the ring oscillator. However, VCO-based PLLs are second-order control loops that are hard to stabilize with the binary outputs of a digital phase detector. Generally these systems will achieve lock only if the initial condition is very close to the desired final lock point, which means they need a linear phase detector that initially locks the loop before the nonlinear detector is enabled. For VCDL-based designs, bang-bang control does not cause any difficult issues. However, these designs have their own limitations when used in links. One is generating a phase shift of half a bit time to center the sample clock.<sup>10</sup> Another is the limited range of the delay line which requires the designer to ensure that the loop never tries to adjust the phase to a point beyond the range of the delay line. In addition to added design complexity, for systems where the clock frequencies of the transmitter and receiver differ slightly, a standard DLL cannot be used as it will run out of range.

### 19.5.2 Dual-Loop Architectures

A dual-loop PLL architecture offers a simple solution to using an input receiver as a phase detector [14], [38]. Figure 19.16 illustrates a block diagram of a dual-loop architecture. By combining a core loop to generate a set of coarsely spaced clocks with a peripheral loop that interpolates between these clocks, these systems can decouple frequency and phase acquisition and enable the construction of DLLs with infinite phase range. The core loop can be constructed as a DLL [38] or PLL [14], [23] which can be either analog or digitally controlled. The peripheral loop only needs to acquire phase under the control of a digital phase detector and is implemented as a bang-bang

<sup>10</sup> The problem is that the control loop generally adjusts the delay to align the END of the clock buffer chain to be equal to the input clock. So unless the delay of the buffer is known (which is not the case), the loop does not have an easy way to shift the clock by one-half a bit time.

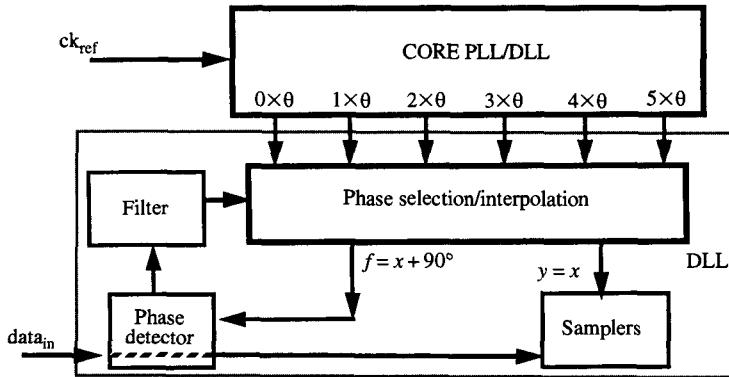


Figure 19.16 Dual-loop PLL/DLL architecture.

loop with digital control. The separate control of the length of a delay element (the core loop) and the adjustment of the phase means this architecture can be used to recover clocks in many different system environments.

Systems with either a single global clock or different local clocks can use a dual-loop architecture. If a global clock is used, the peripheral loop will select a constant phase shift that compensates for the internal clock's buffer delay and the time of flight of the signals. If the local clocks have slightly different frequencies, this loop will slowly rotate the phase to keep the clock aligned with the incoming data—the phase rotation of the peripheral loop compensates for the small frequency difference in the clocks. This rotation is possible because the peripheral loop does not have any boundary conditions and can rotate through a full  $360^\circ$ .

To center the sample clock, two matched peripheral loops are connected to a single core loop; one loop drives the sample clock, and the other drives the clock to the receiver acting as a phase detector. The clocks are shifted  $90^\circ$  in phase from each other (for a system that transmits 2 bits per cycle). The peripheral loop feedback sets the phase-detector clock to be in the middle of the transition region and causes the normal clock to be aligned in the center of the bit eye. The phase offset of this centering is set by the precision of the  $90^\circ$  phase shift and the matching of the two clock paths. To eliminate the offset caused by the mismatch in the clock paths, some recent links use the data receiver directly as the phase detector. These systems periodically send calibration packets, and during this time either the transmit clock or receiver clock is shifted in phase. The resulting offset can be quite low ( $< 10\text{ ps}$  in a  $2\text{ Gbps}$  link) [3].

The dual-loop clocking architecture is an example of how higher levels of integration can be used to deal with both silicon and system limitations. However, increasing a link's data rate still depends on achieving higher on-chip clock rates. As discussed in the following section, basic technology scaling will assist in the realization of that goal.

### 19.5.3 Timing Circuit Scaling

Achieving smaller symbol times requires proportional shrinking of both the AC and DC timing errors. Jitter (which depends on the amount of supply/substrate noise, the sensitivity to noise, and the bandwidth of the noise tracking) should scale with technology, if certain constraints are met. Supply/substrate noise is commonly due to

digital switching and, hence, is a constant percentage of the supply voltage. As supply voltage scales with technology, so does the expected noise. The supply/substrate sensitivity of a buffer is typically a constant percentage of its delay, on the order of 0.1–0.3% delay/% supply for differential designs using replica feedback biasing [28]. For a DLL, this implies that the supply- and substrate-induced jitter scales with operating frequency—the shorter delay in the chain, the smaller the phase noise. This argument also holds for the jitter caused by the buffer chain that follows the DLL. As technology scales, the delay of the buffer chain scales, and so does the resulting jitter. For a VCO-based PLL, because noise is integrated by the VCO, the jitter scaling additionally relies on whether the loop bandwidth (and thus the input clock frequency) scales. This depends on the system costs of bringing a higher-frequency clock onto the chip.

Scaling of static timing errors is more challenging. The fundamental source of these errors is random device mismatch, which affects both the position of clock edges and the absolute position of a receiver's sampling aperture. For example, mismatch between a timing loop's phase detector and the actual receiver becomes an increasing component of static timing offset. Similarly, in oversampling links with multiphase clocking, phase spacing errors as a percentage of the bit width generally increase with decreasing transistor feature sizes. The main reason is that the device threshold voltage is becoming a larger fraction of the clock's scaled voltage swing. Fortunately, the static nature of these errors enables their cancellation through static calibration schemes. For example, the interpolating clock-generation architectures described in [29], [47] can be augmented with digitally controlled phase interpolators [38] and digital control logic to effectively cancel device-induced phase errors. Also, active offset-cancellation circuitry [43] in comparators will become necessary to mitigate voltage offsets and their effect on timing aperture position.

Applications of static-timing calibration techniques for very high data rate links will become more practical as shrinking device dimensions will enable the greater complexity associated with integrating more support circuits on-chip. Similar circuit complexity can be added to deal with imperfections of transmitters and receivers. The next section discusses how increasing degrees of integration can be used to deal with one more obstacle: wire bandwidth.

## 19.6 FUTURE TRENDS

The previous sections have shown that the bit rates of the CMOS interface circuits will continue to scale with technology, with 2:1 multiplexor interfaces providing 4–5 Gbps signaling in 0.1 μm technology. Unfortunately as bit rates increase, the low-pass filtering of the wire will become more important, constituting the key obstacle to overcome. The 4 Gbps rate possible today using high-degree multiplexing is already higher than the bandwidth of copper cables longer than a few meters.

While designs must account for finite wire bandwidth, it will not fundamentally limit signaling rate, at least not for a while. The question of how to maximize the number of bits communicated through a finite bandwidth channel is an old one. It is the basis of modem technology which transfers 30–50 Kbps through the limited bandwidth (4 KHz) of phone lines [2], [4], [16], [17]. To counteract the limited bandwidth, systems make better use of the available signal-to-noise ratio (SNR) to transmit multiple bits per symbol; see [33] for a good description of these methods. This section will focus on the issues in

applying some of these techniques to systems with very high symbol rates, where it is difficult to achieve a high-resolution representation of the received waveform.

### 19.6.1 Equalization

The bandwidth limitation of the cable depends on the frequency dependence of the conductor dissipation and dielectric loss. The transfer function of a 6 m and 12 m RG55U cable, shown in Fig. 19.17, illustrates increasing attenuation with frequency. Figure 19.18 illustrates the effects of that frequency-dependent attenuation in the time domain. When a single 4 Gbps bit is injected onto a 12 m cable, the filter causes the pulse amplitude to be reduced by more than 40% and creates a long settling tail which significantly closes the resulting data eye.

The simplest way to achieve a higher data rate, even with the lossy channels, is to actively compensate for the uneven transfer function. This can be accomplished either through an equalizing receiver or a predistorting transmitter.

Receiver equalization utilizes a receiver with increased high-frequency gain in order to flatten the system response [39]. This filtering can be implemented as an amplifier with the appropriate frequency response. Alternatively the required high-pass filter can be implemented in the digital domain by feeding the input to an analog-to-digital converter (ADC) and postprocessing the ADC's output. Using an ADC at the input and building the filters digitally is the usual technique, because it also allows one to implement more complex nonlinear receive filters. While this approach works well when the data input is bandwidth limited to several MHz, it becomes more difficult if not impossible with GHz signals because of the required GSamples/sec converters. Instead of a digital implementation for these high bit rates, the filter was implemented as a one-tap finite impulse response (FIR) ( $1\alpha D$ ) switched-current filter [42].

Most high-speed links use transmitter predistortion to equalize the channel [45], [5], [11], [32]. This technique utilizes a filter that precedes the actual line driver. The filter is driven by the bit stream that is to be transmitted, and the coefficients of the filter are chosen to decrease the low-frequency energy, thus equalizing the channel response. In

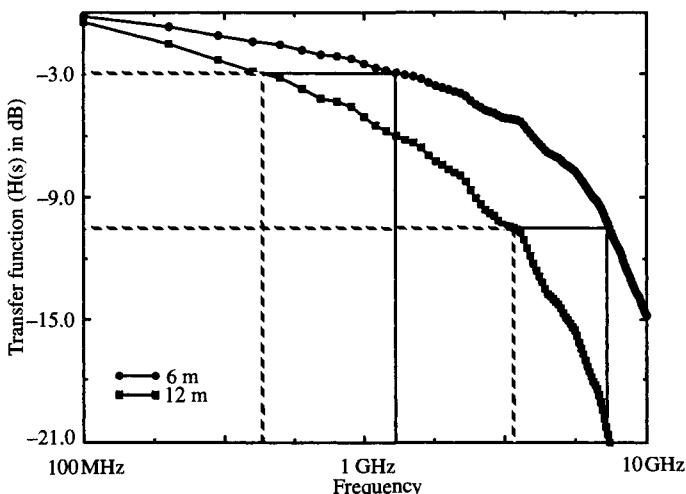
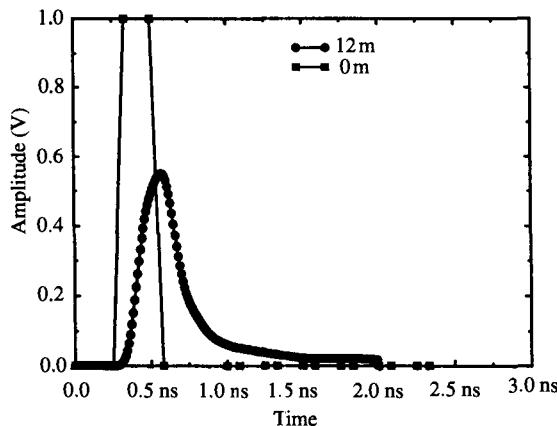


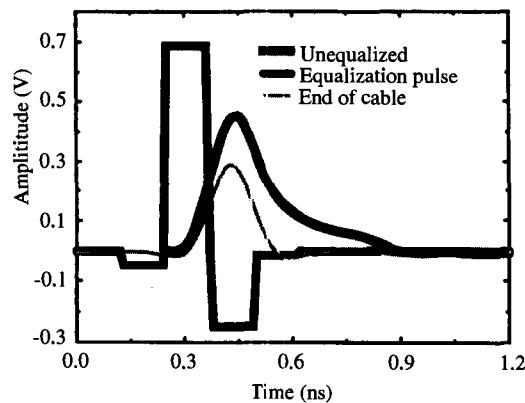
Figure 19.17 Transfer function of a 12 m RG55U cable.



**Figure 19.18** Time-domain response of a 0.25 ns pulse into a 12 m RG55U cable.

the time domain, this reduces the amplitude of continuous strings of same-value data on the line. The length of the optimal filter depends on the tail of the single-pulse response. It is essentially the number of bits that can affect the currently transmitted bit. For cable attenuation, one or two taps is sufficient [5], [10], [11]. A sufficiently long filter can compensate even for reflections. The effect of transmitter equalization is shown in Fig. 19.19. The equalizer transmits small negative pulses before and after the original pulse to eliminate the tail of the pulse response. The output of this FIR filter is a quantized representation of the desired output voltage, so the output driver becomes a high-speed DAC. This DAC already exists in most current-mode output drivers, since it is used to adjust the output current. The only new requirement is the need for linearity between the control word and the output current.

The downside of transmitter equalization is that the transmitter does not have any information about the signal shape at the receiver end of the channel. Obtaining the appropriate FIR filter coefficients can be challenging. So far, predistorting high-speed transmitters utilize either static filter coefficients or a software loop to program the FIR.



**Figure 19.19** Pulse response from a predistorted transmitter.

It will not be difficult for future systems to use periodic training that requires back-channel information to ensure robust operation under varying channel conditions.

Equalization, in either the transmitter or receiver, is the easiest and most cost-effective technique for extending link bit rate. However, equalization does not take full advantage of the channel capacity, since it simply attenuates the low-frequency components of the signal so that they match the high-frequency channel-attenuated components (shown in the horizontal lines in Fig. 19.17). This attenuation reduces the overall signal power, and hence degrades the system SNR. Degrading SNR is wasteful, at least in theory. Shannon showed that the maximum capacity (in bits per second) per hertz of channel bandwidth depends on SNR and is given by the relationship  $\text{Capacity}/f_{bw} = \log(1 + \text{SNR})$ . The larger the signal-to-noise ratio, the more information can be transferred for a given bandwidth by transmitting multilevel signals. The key is the signal-to-noise ratio. To get multiple bits/Hz, the system needs large SNR, and this will not happen if it has significant amounts of proportional noise (uncorrected reflections or crosstalk). If all of the large proportional noise sources, including reflections and noise coupling, can be canceled through filtering, the resulting SNR of the link will be high, and it will be possible to send multilevel symbols on the channel to increase the data rate.

## 19.6.2 Multilevel Signaling

By transmitting multiple bits in each symbol time, the required bandwidth of the channel for a given bit rate is decreased. The simplest multilevel transmission scheme is pulse amplitude modulation. This requires a digital-to-analog converter (DAC) for the transmitter and an analog-to-digital converter (ADC) for the receiver. During each symbol,  $N$  analog levels are transmitted, communicating  $\log_2(N)$  bits. An example is four-level pulse amplitude modulation (4-PAM) where each symbol time comprises 2 bits of information. A 5 GSym/sec, 2 bit dataeye is shown in Fig. 19.20 [10]. As was mentioned earlier, the key to the operation of these systems is the SNR at the input receiver. While a 10% reflection noise is not an issue for a binary link (voltage margin starts at  $\pm 50\%$ ), it is nearly the entire voltage margin in a 4-PAM system (voltage margin starts at  $\pm 17\%$  of full swing).

For these systems to succeed, all proportional noise sources need to be either small

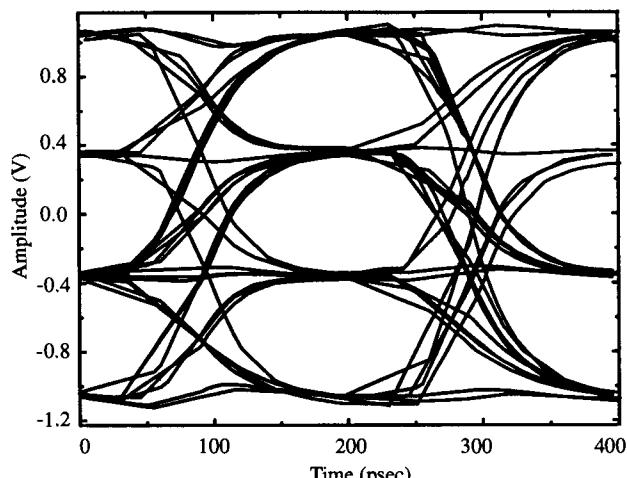


Figure 19.20 Simulated 4-PAM data eye.

or accounted for by the link. For systems with good quality channels, multilevel signaling and equalization might do better than just equalization alone. To really gain the ultimate benefit of Shannon's theorem, the system must model and compensate for all sources of proportional noise. Low-speed links match the analog input waveforms with complex (100 tap) adaptive FIR models of the channel. For high-speed links, the limitations will come from the quality of the ADC and DAC, and the amount of signal processing power available.

### 19.6.3 DAC and ADCs

The realization of more advanced signaling techniques requires higher analog resolution both for the transmitter and the receiver. While equalization can be implemented solely with DACs, all the other techniques require both high-speed DACs and ADCs. The intrinsic noise floor for these converters is thermal noise,  $v = \sqrt{4kT\Delta f}$ . In a  $50\Omega$  environment, the noise is roughly  $1\text{nV}/\sqrt{\text{Hz}}$ , which is negligible for 2–5 bits resolution even at GHz bandwidth (@ 4 GHz,  $3\sigma < 200\mu\text{V}$ ).

Implementing an output DAC is not foreign to link designers—DACs are already embedded in output drivers for impedance or current control. Furthermore, implementing high-speed DACs with less than 8 bit resolution is less challenging than implementing similar speed and resolution ADCs, since transmitter device sizes are large and hence inherently less sensitive to device mismatches due to process variations. Instead, these DACs are limited by circuit issues—primarily transmit clock jitter, and the settling of output transition-induced glitches. Both quantities are related to the intrinsic technology speed and therefore can be predicted to scale with reduced transistor feature size. High-speed, low-resolution DACs are rarely used as products; the nearest comparable product is found in video display RAMDACs achieving 8 bit linearity operating at 320 MHz on a  $0.8\mu\text{m}$  process [40]. In the research domain, a 2 bit driver achieving 5 GS/s in a  $0.3\mu\text{m}$  technology has been demonstrated [10].

The receive-side ADC is more difficult because it requires accurate sampling and amplification of signals on the order of 30 mV (5 bits of resolution for a 1 V signal) at a very high rate. Examples of high-speed ADCs can be found in disk-drive read-channels, where a 6-bit flash-ADC operating at 200 MHz has been demonstrated in a  $0.6\mu\text{m}$  process technology [35]. Recent research demonstrated a multi-GS/s 4 bit ADC [8] which indicates that high data rate conversion is possible.

## 19.7 CONCLUSION

Technology improvements have enabled the development of higher bit-rate links by simply scaling device speeds. This scaling was reflected by the fact that bit rates of links are scaling along with the FO-4 technology speed metric. Figure 19.21 shows the published performance for both serial and parallel links fabricated in different technologies. Low-latency parallel links have maintained bit times of 3–4 FO-4 inverter delays. Meanwhile, by applying high degrees of parallelism, the off-chip data rate of serial links has achieved bit times of one FO-4 inverter delays.

Recent research and development results indicate that the fundamental limitation of bit-rate scaling stems from the communication channel, rather than semiconductor technology and circuit design. Frequency-dependent channel attenuation limits the bandwidth by introducing intersymbol interference. As long as sufficient signal

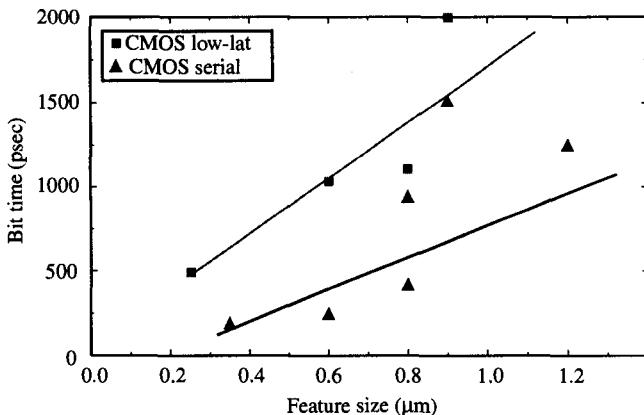


Figure 19.21 Scaling of link bit time with transistor feature size.

power exists, the simplest technique to extend the bit rate beyond the cable bandwidth is transmitter predistortion.

Current research is focused on finding methods to extend the performance of links beyond what can be done with equalization, by exploiting the large SNR of the channel. This interest is leading to very high rate ADC and DAC components. While some groups have demonstrated encouraging results, the practicality of these techniques remains an interesting, but open, question.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the students in the high-speed links research group at Stanford University and the technical staff at Rambus for providing valuable data and suggestions. This work is partially supported by grants from HP, LSI Logic, and DARPA.

## REFERENCES

- [1] R. Best, *Phase-Locked Loops*, 3rd ed. McGraw-Hill, New York, 1997.
- [2] J. Boxho, et al., "An Analog Front End for Multi-standard Power Line Carrier Modem," *1997 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 84–85.
- [3] K. Y. Chang, et al., "A 2 Gb/s Asymmetric Link for High-Bandwidth Packet Switches," *Hot Interconnects V Symposium Record*, Stanford, 171–179, August 1997.
- [4] M. Combe, et al., "A Second-Generation Modem Analog Front-End," *1992 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*. pp. 224–225, 290.
- [5] W. J. Dally, et al., "Transmitter Equalization for 4-Gbps Signaling," *IEEE Micro*, vol. 17, no. 1, pp. 48–56, Jan.–Feb. 1997.
- [6] A. DeHon, et al., "Automatic Impedance Control," *1993 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 164–165, Feb. 1993.

- [7] K. S. Donnelly, et al., "A 660 MB/s Interface Megacell Portable Circuit in 0.3  $\mu\text{m}$ –0.7  $\mu\text{m}$  CMOS ASIC," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 12, pp. 1995–2003, Dec. 1996.
- [8] W. Ellersick, et al., "GAD: A 12-GS/s CMOS 4-bit A/D Converter for an Equalized Multi-Level Link," *Proceedings of 1999 IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, pp. 49–52.
- [9] G. L. Esch, Jr., et al., "Theory and Design of CMOS HSTL I/O Pads," *Hewlett-Packard Journal*, vol. 49, no. 3, pp. 46–52, Aug. 1998.
- [10] R. Farjad-Rad, et al., "A 0.3- $\mu\text{m}$  CMOS 8-GS/s 4-PAM Serial Link Transceiver," *Proceedings of 1999 IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, pp. 41–44.
- [11] A. Fiedler, et al., "A 1.0625 Gbps Transceiver with 2x-oversampling and Transmit Signal Pre-emphasis," *1997 IEEE International Solids-State Circuits Conference, Digest of Technical Papers*, pp. 238–239.
- [12] M. Galles, et al., "Spider: A High-Speed Network Interconnect," *IEEE Micro J*, vol. 17, no. 1, pp. 34–39, Jan.–Feb. 1997.
- [13] F. Gardner, *Phase Lock Techniques*. John Wiley and Sons, New York, 1979.
- [14] M. Horowitz, et al., "PLL design for a 500 MB/s Interface," *1993 IEEE International Solids-State Circuits Conference, Digest of Technical Papers*, pp. 160–161.
- [15] T. H. Hu, et al., "A Monolithic 480 Mb/s Parallel AGC/decision/clock-recovery Circuit in 1.2- $\mu\text{m}$  CMOS," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 12, pp. 1314–1320, Dec. 1993.
- [16] H. Ishida, et al., "A Single-Chip V.32 bis Modem," *1994 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 66–67.
- [17] International C.C.I.T.T. Recommendations, "V.29–V.33," Geneva.
- [18] H. O. Johansson, et al., "Time Resolution of NMOS Sampling Switches Used on Low-swing Signals," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 2, pp. 237–245, Feb. 1998.
- [19] M. G. Johnson, et al., "A Variable Delay Line PLL for CPU-Coprocessor Synchronization," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 5, pp. 1218–1223, Oct. 1988.
- [20] B. Kim, et al., "A 30-MHz Hybrid Analog/Digital Clock Recovery Circuit in 2- $\mu\text{m}$  CMOS," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 6, pp. 1385–1394, Dec. 1990.
- [21] S. Kim, et al., "An 800 Mbps Multi-Channel CMOS Serial Link with 3x Oversampling," *IEEE 1995 CICC Proceedings*, p. 451, Feb. 1995.
- [22] N. Kushiyama, et al., "A 500-megabyte/s Data Rate 4.5M DRAM," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 4, pp. 490–498, April 1993.
- [23] P. Larsson, et al., "A 2-1600 MHz 1.2–2.5 V CMOS Clock-recovery PLL With Feedback Phase-selection and Averaging Phase-interpolation for Jitter Reduction," *1999 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 356–357, Feb. 1999.
- [24] B. Lau, et al., "A 2.6 GB/s Multi-Purpose Chip to Chip Interface," *1998 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 162–163.
- [25] K. Lee, et al., "A Jitter-tolerant 4.5 Gb/s CMOS Interconnect for Digital Display," *1998 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 310–311, Feb. 1998.
- [26] K. Lee, et al., "A CMOS Serial Link for Fully Duplexed Data Communication," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 4, pp. 353–364, April 1995.
- [27] T. H. Lee, et al., "A 2.5 V CMOS Delay-locked Loop for 18 Mbit, 500 megabyte/s DRAM," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 12, pp. 1491–1496, Dec. 1994.
- [28] J. G. Maneatis, et al., "Low-jitter Process-independent DLL and PLL Based on Self-biased Techniques," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1723–1732, Nov. 1996.

- [29] J. G. Maneatis, et al., "Precise Delay Generation Using Coupled Oscillators," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 12, pp. 1273–1282, Dec. 1993.
- [30] R. Matick, *Transmission Lines for Digital and Communication Networks*, 3rd ed. IEEE Press, New York, 1997.
- [31] R. Mooney, et al., "A 900 Mb/s bidirectional signaling scheme," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 12, pp. 1538–1543, Dec. 1995.
- [32] J. Poulton, et al., "A Tracking Clock Recovery Receiver for 4Gb/s Signaling," *Hot Interconnects V Symposium Record*, Stanford, pp. 157–169, August 1997.
- [33] J. Proakis and M. Salehi, *Communication Systems Engineering*. Prentice Hall, Engelwood Cliffs, NJ, 1994.
- [34] B. Razavi (Ed.), *Monolithic Phase Locked Loops and Clock Recovery Circuits*. IEEE Press, New York, 1996.
- [35] D. Reynolds, "A 320 MHz CMOS Triple 8 bit DAC with PLL," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 12, p. 154, Dec. 1994.
- [36] S. Sidiropoulos, et al., "A CMOS 500-Mbps/pin Synchronous Point to Point Link Interface," *Proceedings of 1994 IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, pp. 43–44.
- [37] S. Sidiropoulos, et al., "A 700-Mb/s/pin CMOS Signaling Interface Using Current Integrating Receivers," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 681–690, May 1997.
- [38] S. Sidiropoulos, et al., "A Semi-digital DLL With Unlimited Phase Shift Capability and 0.08–400 MHz Operating Range," *1997 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 332–333.
- [39] B. S. Song, et al., "NRZ Timing Recovery Technique for Band Limited Channels," *1996 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 194–195.
- [40] J. Spalding, et al., "A 200 M Sample/s 6b Flash ADC in 0.6  $\mu$ m CMOS," *1996 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 320–321.
- [41] T. Takahashi, et al., "A CMOS Gate Array With 600 Mb/s Simultaneous Bidirectional I/O Circuits," *IEEE Journal of Solid State Circuits*, vol. 30, no. 12, Dec 1995.
- [42] H. Tamura, et al., "Partial Response Detection Technique for Driver Power Reduction in High Speed Memory-to-processor Communications," *1997 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 241–248.
- [43] R. Van de Plassche, *Integrated Analog-to-Digital and Digital-to-Analog Converters*. Kluwer Academic Publishers, 1994.
- [44] G. Wei, et al., "A Variable Frequency Parallel I/O Interface with Adaptive Supply Regulation," *2000 IEEE Solid State Circuits Conference, Digest of Technical Papers*, p. 298.
- [45] A. X. Widmer, et al., "Single-chip 4\*500-MBd CMOS Transceiver," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 12, pp. 2004–2014, Dec. 1996.
- [46] D. Weinlader, et al., "An Eight Channel, 36GS/s CMOS Logic Analyzer," *2000 IEEE Solid State Circuits Conference, Digest of Technical Papers*, p. 170.
- [47] C. K. K. Yang, et al., "A 0.8- $\mu$ m CMOS 2.5 Gb/s Oversampling Receiver and Transmitter for Serial Links," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 12, pp. 2015–2023, 1996.
- [48] C.K.K. Yang, et al., "A 0.5- $\mu$ m CMOS 4-Gbps Serial Link Transceiver with Data Recovery using Oversampling," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, pp. 713–722, May 1998.

PART  
**VII**

**RELIABILITY**

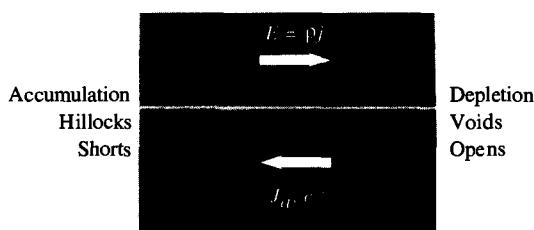
J. Joseph Clement  
*Sandia National Laboratories*

## 20.1 INTRODUCTION

Thin-film interconnects on an integrated circuit can be required to handle current densities in excess of  $10^5 \text{ A/cm}^2$ . This is on the order of 1000 times that allowed for house wiring. These relatively high current densities can be accommodated because of the excellent heat sinking provided through the thin insulating dielectric layers and the underlying silicon substrate. In addition, the small cross-sectional area of an integrated circuit interconnect means Joule heating is greatly reduced compared to conventional wires.

The use of higher current densities in integrated circuit design is precluded by concerns over long-term reliability due to the phenomenon of electromigration. Electromigration is the atomic transport arising from a momentum transfer from the current conduction electrons to the constituent metal atoms. This “electron wind” driving force creates a net flux of metal atoms in the direction of the electron flow. As the atoms migrate, there will be a depletion of material “upstream” and an accumulation “downstream” at sites of atomic flux divergence (Fig. 20.1). This process may lead to void formation and growth at sites of material depletion resulting in a large increase in electrical resistance, or to dielectric cracking and the formation of extrusions at sites of material accumulation resulting in a short between adjacent lines, thereby ultimately causing the circuit to fail.

Of course, without a divergence in the atomic flux, there would be no electromigration damage. There are many possible causes of a flux divergence in thin-film interconnects, including microstructural inhomogeneities, temperature gradients, and contact between dissimilar materials, such as aluminum contacting a tungsten-filled via. At sites of atomic flux divergence, the electromigration-induced material accumulation and depletion leads to the development of localized compressive and tensile stresses, respectively. A backflow flux due to the resulting stress gradient is then created,



**Figure 20.1** SEM micrograph of electromigration damage in an interconnect test line showing the formation of a hillock “upstream” and a void “downstream” of the electron flow.

which opposes the electromigration flux. Therefore, it is essential to consider mechanical stress effects in evaluating interconnect reliability. This includes electromigration-induced stresses, as well as other sources, such as the stresses created due to differences in the thermal expansion rates of materials.

## 20.2 MATERIALS AND PROCESS EFFECTS ON ELECTROMIGRATION

The materials, structures, and techniques used in reliability studies to characterize electromigration have evolved over the years, in conjunction with the changes in the materials and processing methods used to fabricate integrated circuit interconnects.

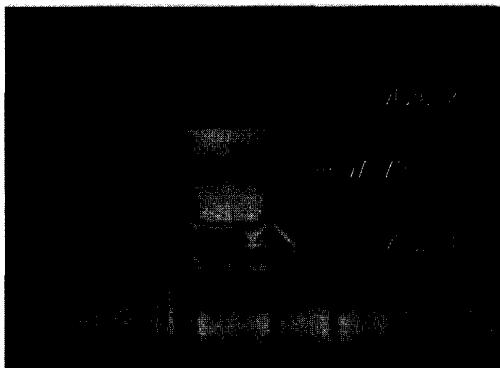
### 20.2.1 Interconnect Technology Evolution

Since being first identified as a failure mechanism in integrated circuits in the late 1960s [1], [2], electromigration in thin-film metals, in particular aluminum and aluminum alloys, has been extensively studied. Aluminum became the interconnect material of choice for a variety of important reasons. It is inexpensive, has a fairly low resistivity (only gold, copper, and silver are lower), adheres well to silicon dioxide, and forms a self-passivating oxide. It can be easily deposited and etched, and does not introduce deep-level traps in silicon, which kill minority carrier lifetime (unlike gold, silver, and copper).

It was demonstrated fairly early on that small amounts of solutes added to the aluminum could significantly improve electromigration reliability, the best-known example being alloys with Cu [3]. In the 1970s, semiconductor manufacturers often added a small amount of silicon (1–2 wt%) to the aluminum in order to help prevent shallow junction shorting or “spiking” [4]. In the 1980s, thin layers of refractory materials were introduced as diffusion barriers between the aluminum and the silicon junction. Processes using CVD tungsten to fill contacts and vias with high-aspect ratios were introduced in the late 1980s. These layers of refractory materials interposed between the aluminum and the silicon junction eliminated the need for silicon doping of the aluminum, and the silicon was removed. At about the same time, thin layers of refractory materials on top of the aluminum were introduced as antireflective coatings to improve the photolithographic patterning of very fine interconnect line widths.

A state-of-the-art manufacturing process with aluminum-based interconnects, as shown in Fig. 20.2, will typically use an aluminum–copper alloy (usually 0.5–1 wt% Cu) sandwiched between much thinner layers of refractory conductors, such as TiN,  $TiAl_3$ , Ti, or some combination of these. Electrical contact from the interconnect to the underlying transistors, as well as between the multiple levels of interconnect is made through tungsten-filled vias, which do not electromigrate. These form perfectly blocking boundaries for the aluminum atomic flux and, therefore, are preferred sites for void formation.

The thin refractory layers over and under the aluminum are highly impervious to electromigration, but unfortunately have much higher resistivity than aluminum. However, they do provide a redundant electrical path, and some measure of protection for the circuit, if a void develops in the aluminum. Whereas earlier, the formation of a void spanning the line would cause an open circuit, in modern multilevel metallization



**Figure 20.2** This SEM micrograph shows a FIB cross section of a tungsten-filled via connecting two levels of aluminum–copper wiring. Thin TiN refractory layers, above and below the thick aluminum–copper, in each metal level can be seen. The void at the bottom of the blocking tungsten plug in Metal-1 formed during accelerated electromigration stressing.

with redundant refractory layers voiding will result in a resistance increase. Therefore, *limited* void growth may not significantly affect interconnect resistance or circuit performance.

Early on, thin aluminum and aluminum-alloy films were typically deposited by evaporation. Later, sputter deposition became the method of choice because it gave better control of the aluminum alloy composition, and could also be used to deposit the thin refractory metal layers at the top and bottom of the metal stack. It is widely expected that copper will eventually replace aluminum alloys as the interconnect material in high-speed circuit applications because of its lower resistivity. Presently, the most popular technique for depositing copper interconnects in a damascene process is electroplating. This metallization technology typically uses a thin refractory metal film, such as Ta, Ti, TaN, or TiN, as a copper diffusion barrier to line the oxide trench prior to copper deposition. This thin refractory metal liner should offer similar protection from voiding in copper lines.

### 20.2.2 Effect of Interlevel Dielectrics

Silicon dioxide is presently the material that is commonly used as the dielectric to electrically isolate the various levels of interconnect on the integrated circuit. It was recognized very early that encapsulating the aluminum line with a protective glass layer improves electromigration reliability [4], [5]. This rigid encapsulation allows local electromigration-induced compressive stresses to build up in the interconnect, much larger than the yield stress for an aluminum line without glassification. The resulting stress gradient in turn will create a backflow flux that opposes the electromigration flux, prolonging the interconnect lifetime.

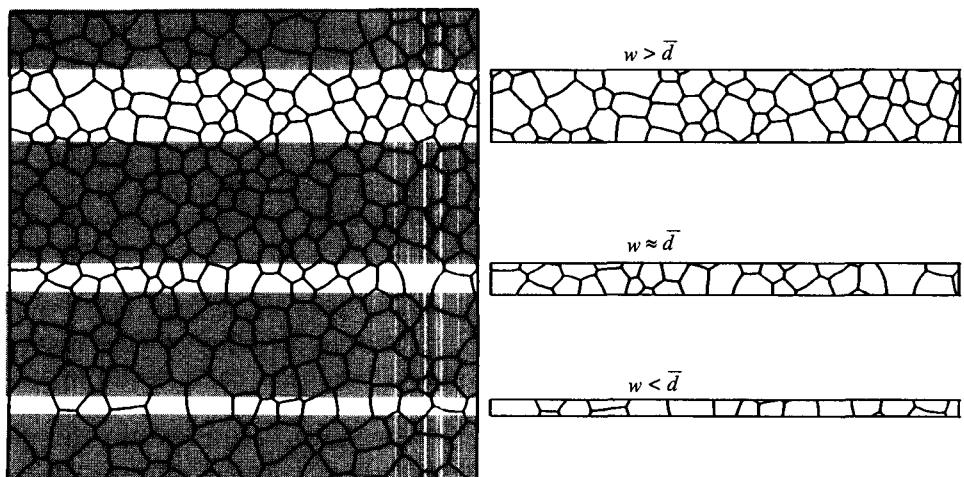
However, confining the interconnect in a rigid dielectric also has the undesirable consequence of introducing tensile stresses due to differential thermal expansion. As the silicon wafer is cooled from the silicon dioxide deposition temperature, the aluminum wants to shrink faster than the silicon substrate, but is prevented by the rigid, well-adhering dielectric. The stresses are exacerbated in narrow lines and with thicker dielectrics. The resulting tensile stress is the driving force behind the phenomenon called “stress voiding” [6], which is closely related to electromigration. This again points out that mechanical stress and electromigration forces are inextricably linked in determining interconnect reliability.

There is considerable interest in finding materials with a lower dielectric constant than silicon dioxide for use as interlevel dielectrics. These “low-k” dielectrics should reduce RC delay and capacitive coupling, thereby making possible faster chips. A few studies have shown improved electromigration performance using polyimide or other less rigid dielectric films compared to silicon dioxide [7], [8].

In evaluating potential alternatives to the ubiquitous silicon dioxide as an inter-level dielectric, another important factor to consider from a reliability perspective is the thermal conductivity [7]. To first order the temperature rise due to Joule heating in the interconnect is inversely proportional to the thermal conductivity of the underlying dielectric. Since electromigration lifetime has an exponential temperature dependence, this can have serious deleterious consequences on interconnect reliability. The thermal conductivity of silicon dioxide is quite good compared to alternative low-k dielectrics.

### 20.2.3 Microstructure, Line Width, and Line Length

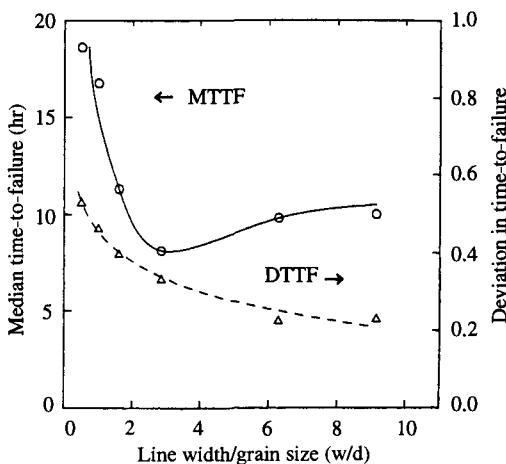
The metallic thin films used to form interconnects on integrated circuits are polycrystalline, composed of many small individual crystals called grains. Aluminum and aluminum-alloy thin films sputter-deposited on planar substrates have columnar grains, that is, the grain boundaries are roughly perpendicular to the surface and span the entire thickness of the film. Moreover, the grain sizes are found to be lognormally distributed. The grain size is influenced by a number of factors, including the type of substrate, surface roughness, deposition conditions, and temperature. The diagram in Fig. 20.3(a) illustrates the grain boundary microstructure in a continuous polycrystalline film with a lognormal grain size distribution.



**Figure 20.3** Simulated thin-film grain structure: (a) continuous polycrystalline film, and (b) three lines of various widths patterned from the continuous film. For a line width  $w$  greater than the average grain size  $\bar{d}$ , there will be a continuous grain boundary path for electromigration. For  $w \approx \bar{d}$ , the grain boundary path is interrupted by spanning, bamboo grains. For  $w < \bar{d}$ , a large portion of the line is made up of bamboo grains and grain boundary paths are very short.

The electromigration lifetime of interconnects is inherently statistical as a result of intrinsic variations in the grain microstructure. Inhomogeneities in the local grain structure can cause a divergence in the electromigration atomic flux. Failure is more likely to occur when there is an abrupt change in grain size [9]. A region with smaller, fine grains will have more grain boundary paths for material transport than a region of larger, coarse grains.

The dependence of electromigration lifetime on line width is rather complex [10], [11]. Figure 20.3(b) shows schematically examples of three different line widths patterned from a continuous polycrystalline film. If the line width is significantly larger than the average grain size, the line is likely to have a continuous grain boundary path along its length. When the line width approaches the average grain size, the grain boundary path is interrupted by “bamboo” grains, which span the width of the line. Electromigration transport across a bamboo grain must occur through the lattice or along the interface. Grain boundary diffusivity is much higher than diffusivity through the lattice or along the interface, so electromigration will occur much more rapidly along a polygranular cluster than it will at a spanning grain. Spanning grains, therefore, are sites of electromigration flux divergence. The lifetimes of lines with a few spanning grains should be lower than wider lines with no spanning grains. This effect is illustrated in the data shown in Fig. 20.4.



**Figure 20.4** Median time-to-failure  $t_{50}$  vs. line width/grain size ratio. As the line width is reduced,  $t_{50}$  slowly decreases, then rises rapidly for very narrow lines with bamboo microstructure. (From [11].)

However, as the line width is reduced still further, such that it is much smaller than the average grain size, the occurrence of spanning grains becomes much more frequent, and the lengths of the polygranular clusters become shorter. The electromigration damage is lower in shorter polygranular segments, so that the electromigration lifetime again begins to increase at very narrow line widths, as shown in Fig. 20.4.

In addition to the effect of line width on electromigration lifetime, there is a dependence on line length, as well. Short lines are more reliable than long lines [10]. This is understandable given the statistical nature of microstructural discontinuities. In wide lines, a longer line will have a greater chance than a short line to have variation in local grain structure somewhere along its length, thereby reducing its expected lifetime. For near-bamboo lines, a longer line will have a higher probability to have a long polygranular cluster. If the lifetime is determined by the most severe microstructural defect in the metal interconnect, then a decrease in lifetime with increasing line length is to be expected.

## 20.3 ELECTROMIGRATION LIFETIME

To correctly assess the reliability of interconnect metallization, it is essential to understand the impact of environmental and operating conditions on the electromigration failure mechanism. Further, in order to confidently extrapolate time-to-failure from accelerated testing to actual circuit use conditions, it is critically important to develop accurate electromigration failure models.

### 20.3.1 Electromigration Transport

The electromigration atomic flux resulting from an applied electric field  $E$  can be written

$$J_a = -C_a \frac{D_a}{kT} q^* E, \quad (20.1)$$

where  $C_a$  is the atomic concentration,  $D_a$  is the atomic diffusivity,  $kT$  is the thermal energy, and  $q^*$  is the effective charge. The effective charge  $q^* = |Z^*|e$ , where  $e$  is the elementary charge and  $Z^*$  is the effective charge number. The electric field can be expressed as the product of the metal resistivity  $\rho$  and the current density  $j$ , that is  $E = \rho j$ . The diffusivity has an Arrhenius temperature dependence, which we can write explicitly as  $D_a = D_0 \exp(-Q/kT)$ , where  $Q$  is the activation energy. With this, Eq. (20.1) becomes

$$J_a = -C_a \frac{q^* \rho j}{kT} D_0 \exp\left(-\frac{Q}{kT}\right) \quad (20.2)$$

Written in this form it is easier to identify the important parameters that influence the rate of electromigration degradation. The parameters  $C_a$ ,  $\rho$ , and  $q^*$  are determined by material properties and cannot be easily changed. The diffusion coefficient  $D_0$  and the activation energy  $Q$  are also both material-dependent parameters, but can vary within a certain range of values depending on processing. For aluminum the activation energy can range from as low as 0.5 eV for grain boundary diffusion, up to 1.3–1.4 eV for diffusion through the lattice. Since the diffusivity depends exponentially on  $Q$ , small changes in the activation energy can have a significant impact on the electromigration flux, and consequently on the lifetime. The electromigration reliability of copper interconnects may potentially be better than aluminum-based wiring at service temperatures, primarily because most of the diffusive transport pathways should have higher activation energies [13].

The activation energy is low for transport along grain boundaries in wide, pure aluminum lines. As mentioned in Section 20.2.1, solute addition, in particular the addition of Cu, can effectively improve electromigration reliability. The solute tends to segregate to the grain boundary, and it seems to modify the grain boundary diffusivity to retard the flux of atomic aluminum, thereby increasing the activation energy for transport along grain boundaries. As the line width narrows to become comparable to the average grain size, electromigration along high-diffusivity grain boundaries is reduced, and atomic transport via other paths, such as along interfaces or through the lattice, becomes important, as described in Section 20.2.3. Thus, the effective activation energy increases as the line width decreases, since the activation energies for these other transport mechanisms are higher [12].

As evident in Eq. (20.2), the temperature  $T$  has a significant impact on the electromigration flux. However, the chip temperature is usually fixed by system environmental constraints, including packaging and cooling issues. The current density  $j$  is the single parameter left to our control during the design process in order to meet electromigration reliability goals. Note that current density is important not only as it directly affects the electromigration driving force in Eq. (20.2), but it can also indirectly have a significant impact on the atomic flux by increasing the line temperature through Joule heating. Care must be taken to limit the current to prevent excessive self-heating. Nonuniform Joule heating can create temperature gradients in the interconnect. Since the atomic diffusivity has an Arrhenius temperature dependence, a large temperature gradient itself can cause a flux divergence and lead to failure.

### 20.3.2 Accelerated Lifetime Characterization

To evaluate electromigration reliability, testing of interconnect structures is typically performed at temperatures and currents much higher than the intended use conditions in order to observe failures in a reasonable time frame. Traditionally, electromigration reliability characterization is usually performed using packaged test devices, heated in an oven, under a constant applied current stress while monitoring the voltage *in situ*. Due to random microstructural variations, nominally identical interconnects will not all fail at the same time. Typically, a group of identical structures are tested together under the same stress conditions, yielding a distribution of failure times.

Electromigration failure times are generally represented as a lognormal distribution, whereby a plot of the logarithms of the failure times relative to the cumulative failure percentage on a normal probability scale fits a straight line. The lognormal distribution is characterized by two parameters: the median time-to-failure  $t_{50}$  and the slope or logarithmic standard deviation  $\sigma$ . The lognormal cumulative probability function is given by

$$F(t) = \Phi\left(\frac{\ln(t/t_{50})}{\sigma}\right) \quad (20.3)$$

where  $\Phi$  is the standard normal cumulative distribution function  $\Phi(x) = \frac{1}{2}(1 + \text{erf}(x/\sqrt{2}))$ . The logarithmic standard deviation can be expressed as

$$\sigma = \ln\left(\frac{t_{50}}{t_{16}}\right) \quad (20.4)$$

where  $t_{16}$  is the time at which approximately 16% of the devices fail (more precisely, 15.866%).

To extrapolate the data measured under the accelerated test conditions to operating conditions the following semi-empirical model first proposed by Black [2] is widely used:

$$t_{50} = A j^{-n} \exp(Q/kT) \quad (20.5)$$

where  $A$  and  $n$  are empirically determined constants. The value of  $n$  has been the subject of considerable research and debate. Experimental determination of  $n$  is complicated by the statistical nature of electromigration failure. It is fairly well established that values of  $n > 2$  are typically artifacts of Joule heating effects at high current densities, and that  $1 \leq n \leq 2$  when self-heating is properly taken into account.

### 20.3.3 Modeling Electromigration

Including the driving force due to mechanical stress gradients in addition to the electromigration driving force, the net atomic flux can be written

$$J_a = C_a \frac{D_a}{kT} \left( \Omega \frac{\partial \sigma}{\partial x} - q^* E \right) \quad (20.6)$$

where  $\Omega$  is the atomic volume, and  $\sigma$  is the hydrostatic stress, which is taken to be positive in tension. This formulation assumes that across any transverse section of the interconnect line the stress relaxes to a hydrostatic state on a time scale that is short compared to that for long-range diffusion along the line length [14]. The atomic concentration as a function of position along the line  $x$  and time  $t$  can be obtained by solving the continuity equation

$$\frac{\partial C_a}{\partial t} + \frac{\partial J_a}{\partial x} = 0 \quad (20.7)$$

The details of the solution are presented elsewhere [14], [15].

One important modeling result is the observation that at long times the electromigration-induced stress buildup and void growth will saturate at a steady-state value, which depends on the electromigration driving force. In steady state  $\partial C_a / \partial t = 0$ , and from Eq. (20.6) the steady-state stress gradient is

$$\frac{\partial \sigma}{\partial x} = \frac{q^* \rho j}{\Omega} \quad (20.8)$$

In steady state, then, the electromigration flux due to the electric current will be counterbalanced by the backflow flux due to the stress gradient. This effect was first noted by Blech [16] in drift velocity experiments studying the edge displacement of aluminum islands due to electromigration. He observed that there is a critical threshold for the current density  $j$  and line length  $l$  below which the edge displacement ceases. X-ray topography [17] showed that the stress at the anode edge, where hillocks were seen to form, was at the compressive yield stress  $\sigma_y$ , while the stress at the cathode edge was near zero. The steady-state stress gradient is  $\sigma_y / l$ , and the critical threshold product is given by

$$(jl)_{c, stress} = \frac{\Omega \sigma_y}{q^* \rho} \quad (20.9)$$

Similarly, void growth has been seen to saturate due to the effect of the backflow flux resulting from the electromigration-induced stress gradient [18]. The steady-state void size is well-approximated by [19], [20]

$$\frac{\Delta l}{l} = \left( \frac{q^* \rho}{2B\Omega} \right) jl \quad (20.10)$$

where  $B$  is the appropriate elastic modulus relating the strain to stress in the line. The resistance increase will be proportional to the void size, that is,

$$\frac{\Delta R}{R} = K \frac{\Delta l}{l} \quad (20.11)$$

where the proportionality constant  $K$  depends on the ratio of the sheet resistance of the redundant refractory layer(s) to that of the aluminum alloy, as well as the details of the void morphology. If circuit failure is defined by a percentage increase in resistance  $(\Delta R/R)_{crit}$  or by an absolute change in resistance  $\Delta R_{crit}$ , this will correspond to a

critical void size  $\Delta l_{crit}$ . Therefore, the critical threshold product in this case is given by

$$(jl)_{c, void} = \left( \frac{2B\Omega}{q^* \rho} \right) \left( \frac{\Delta l_{crit}}{l} \right) \quad (20.12)$$

Thus, if the steady-state electromigration-induced stress and void growth saturate at a level below the critical stress or the critical void size required for failure, then the interconnect can be considered to be virtually immortal. The level at which the steady-state stress and void size saturate is proportional to the electromigration driving force, that is, proportional to the  $jl$  product.

Another important result of modeling work is what it reveals about the current density exponent  $n$  in the acceleration model of Eq. (20.5). The electromigration failure mechanism may be controlled by the buildup of stress to a critical level  $\sigma_{crit}$ , for example, a critical compressive stress leading to dielectric cracking. In this case modeling predicts that the time-to-failure will be proportional to  $j^{-2}$ , that is,  $n = 2$ . Alternatively, the lifetime may be determined by void growth to a critical size. In this case modeling yields a failure time that is proportional to  $j^{-1}$ , that is,  $n = 1$ .

Therefore, model results predict that  $n = 1$  when the time-to-fail is controlled by a void growth mechanism, and that  $n = 2$  in the case that damage nucleation is the predominate mechanism determining the failure time. If electromigration failure is a serial process of defect nucleation followed by void growth, such that one mechanism does not dominate, then one might expect a value of  $n$  between 1 and 2. Only in the case that  $jl$  approaches the critical product  $(jl)_c$  as the current density is reduced would we expect values of  $n > 2$ .

Further modeling [15] shows that all of these results obtained for dc stress conditions also apply under pulsed dc stressing where  $j$  becomes the average current density  $j_{avg}$ . This agrees with the empirically derived average current lifetime model discussed next.

### 20.3.4 Pulsed dc and ac Operation

Testing to evaluate electromigration reliability is typically performed using dc stresses, while in most CMOS circuit applications devices operate under pulsed current conditions. In the power bus lines the current is unidirectional (pulsed dc), while the signal lines carry bidirectional current (ac) pulses. It is critical, therefore, that we understand how electromigration reliability under pulsed dc and ac conditions correlates with dc test results.

Consider a repetitive pulsed dc waveform that is rectangular with a period  $P$  and pulse width  $t_{on}$ , so that the duty ratio  $r = t_{on}/P$ . For the case that the peak pulse current is equal to the current applied in a dc stress test, one might expect that the ratio of the lifetime under a pulsed dc stress to the dc lifetime is simply the inverse of the duty ratio, that is,

$$\frac{t_{f,pdc}}{t_{f,dc}} = \frac{1}{r} \quad (20.13)$$

This model assumes that the electromigration damage accumulates when the current is “on,” and the time that the current is “off” has no effect other than to add to the total time-to-failure. It turns out that this model is too conservative and underestimates the time-to-failure under pulsed dc stress. Many experiments [21], [22], [23] have shown

$$\frac{t_{f,pdc}}{t_{f,dc}} > \frac{1}{r} \quad (20.14)$$

This indicates that during the “off” portion of the current waveform there is some relaxation or “healing” of electromigration damage [24]. Several careful experiments [21], [22], [23] have established that, in the absence of significant Joule heating effects, the pulsed dc lifetime depends on the average current. Using the average current density in the electromigration lifetime model given in Eq. (20.5), the average current model [21] is given by

$$t_{50,pdc} = A j_{avg}^{-n} \exp(Q/kT) \quad (20.15)$$

Since  $j_{avg} = rj_p$ , where  $j_p$  is the peak current density,

$$\frac{t_{j,pdc}}{t_{f,dc}} = \frac{1}{r^n} \quad (20.16)$$

Since it has been repeatedly demonstrated and widely accepted that electromigration lifetime depends on the average current for a unidirectional pulse waveform, the question now posed is whether the average current model is also effective in predicting electromigration behavior under bidirectional pulsing. Intuitively this makes sense because reversing the direction of the current reverses the mass transport and presumably cancels out electromigration damage. This effect was qualitatively demonstrated very early on [25] in the study of electromigration phenomena in thin-film metals. However, there is still little quantitative data characterizing the electromigration lifetime under ac pulses.

Electromigration lifetimes can be very long under ac stress [22], [23]. This is particularly true when the average current for the ac waveform is zero. In this case, Eq. (20.15) yields an infinite lifetime. This is the problem faced by experimenters: lifetimes are so long under ac stressing that it is difficult to find failures in a reasonable time. Thus the temptation is to increase the magnitude of the peak current to further accelerate the test. This can complicate the interpretation of the test results by introducing Joule heating effects. The best available data indicate that in the absence of severe self-heating the experimental results with dual-polarity pulses are consistent with the average current model.

A generalized version of the average current model has been proposed for bidirectional pulsed current, the average current recovery model [23]:

$$t_{50,ac} = A j_{eff}^{-n} \exp(Q/kT) \quad (20.17)$$

The effective current density is defined as

$$j_{eff} := j_{max} - \lambda j_{min} \quad (20.18)$$

where  $j_{max} = \max(|j_+|, |j_-|)$ ,  $j_{min} = \min(|j_+|, |j_-|)$ , and  $j_+$  and  $j_-$  are the time-averaged current density including only the positive or negative current, respectively. The single parameter  $\lambda$  heuristically accounts for the degree of damage recovery obtained with ac signals, where  $0 \leq \lambda \leq 1$ . For  $\lambda = 0$ , any annealing or damage recovery is ignored, and  $j_{eff} = j_{max}$ . This is the most conservative choice for reliability evaluation. For  $\lambda = 1$ , perfect annealing occurs, and Eq. (20.17) reduces to the average current model of Eq. (20.15) with the singularity in  $t_{50}$  for pure ac, that is, for  $|j_+| = |j_-|$ . For  $\lambda < 1$ , this singularity goes away.

## 20.4 DESIGNING FOR ELECTROMIGRATION RELIABILITY

In the design and layout of integrated circuits, concerns about electromigration reliability have traditionally been addressed by requiring that the average current per unit width in interconnects be kept below design rule limits or guidelines [26]. With each successive technology generation, as minimum interconnect line widths decrease and the average current increases with higher performance and decreasing cycle times, electromigration design rule limits become more of a problem for designers.

### 20.4.1 Reliability Budgeting

We may view the chip as a system made up of many components, and that the failure of any individual component can cause the system to malfunction. The probability that the system will fail at time  $t$  then is given by

$$P(t) = 1 - \prod_i [1 - p_i(t)] \quad (20.19)$$

where  $p_i(t)$  is the probability of failure for the  $i$ th component. It is not difficult to show statistically that a chip-level system with a few interconnects above the traditional electromigration limit can be just as reliable as a chip with more interconnects running currents at the limit. This concept has been called statistical electromigration budgeting (SEB) [27].

The design rule limits are typically conservative estimates meant to ensure that an overall chip-level reliability goal is achieved. Meeting the design rule limits does not necessarily guarantee that the chip reliability goal is met. It is conceivable that a chip with many lines with currents at the design rule limit might have an unacceptable chip-level reliability risk.

It is much more likely, however, that the electromigration design rule limits are so conservative that the design is adversely affected, for instance, by requiring wider interconnects which take up valuable chip real estate. This points out that the traditional design approach lacks flexibility. With SEB, designers would have the option to accept some interconnects with higher risk, as long as the chip-level electromigration reliability goal is not compromised.

Moreover, large circuit designs have become so complex that designers cannot fully know how a chip will operate. Clearly what is needed is a CAD tool to determine the current and evaluate the electromigration reliability for each interconnect on the chip. This information could then be used to calculate an overall chip-level reliability risk. This would effectively allow the chip reliability goal to be budgeted among the various classes of interconnect (e.g., Metal-1, Metal-2, etc.) or among the chip subcircuits to maximize overall chip performance.

### 20.4.2 CAD Tools

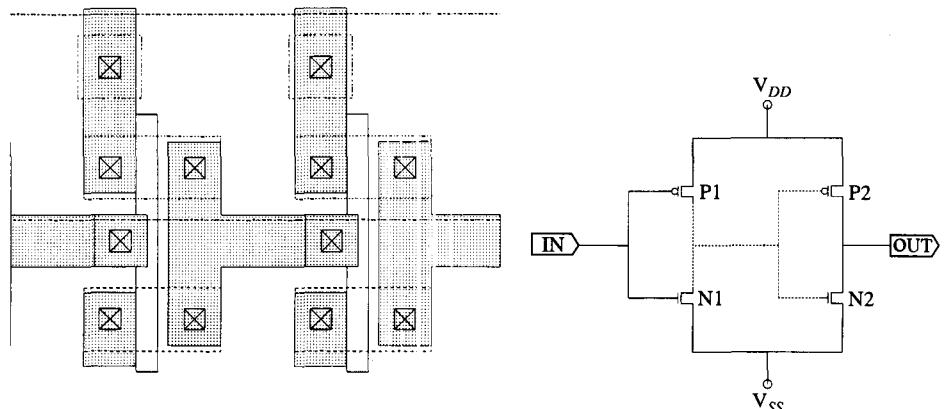
To address this need, several circuit-level electromigration design tools have been developed [27], [28], [29], [30]. These tools basically incorporate electromigration failure models into general-purpose circuit analyzers to evaluate interconnect reliability. Those based on SPICE-like simulators [28] or other timing simulators [29] have been demonstrated on relatively small circuits or only on sections of circuits. Due to computational complexity, these techniques do not appear to be suitable for reliability analysis of very large circuit designs.

Modern ULSI circuits may include tens of millions of interconnect segments in each of the power/ground grids and many thousands of signal networks, each made up of many elements. CAD tools and analysis methodologies have been developed that allow full-chip electromigration reliability evaluation to be carried out on state-of-the-art designs [27], [30]. This section gives a general outline of some of the techniques that can be used to analyze electromigration risk of large ULSI designs.

Interconnect reliability analysis starts with extracting from the layout database the parasitic resistance and capacitance associated with the signal and power networks. In order to support the electromigration analysis, the extraction tool must also provide additional data, such as the interconnect segment width and layout layer (i.e., type of interconnect) associated with each element from the layout database, along with the parasitic resistance and capacitance. The attachment of each transistor to the power/ground grid and to each signal network also is noted in the circuit database.

Analyses of the signal networks and the power grids are done separately, using somewhat different methodologies. The current in power lines is unidirectional, so we need only determine the average current in each interconnect segment for electromigration analysis. Since the current in signal lines may be bidirectional, the evaluation of electromigration reliability is more complicated in that both the average and rms currents need to be checked. We first illustrate a technique for determining the average current in each interconnect segment in a signal line with a simple example. Evaluation of the rms current is discussed in Section 20.4.4.

The layout of a simple two-stage inverter chain is shown in Fig. 20.5(a), and the circuit diagram is shown in Fig. 20.5(b). In this example, we will focus on the signal path from the output of the first inverter, indicated by the dashed lines in the circuit diagram. In particular, we examine the case that this signal path is discharged through the NMOS device N1.



**Figure 20.5** Two-stage inverter example: (a) layout, and (b) circuit diagram with the signal path from the output of the first inverter indicated by the dashed line.

In Fig. 20.6 the signal path is shown as an RC switching network. The resistors are extracted for interconnect wiring segments, each of a particular length, width, type, and thickness, e.g., first-level metal (M1), and for vias of a particular type and size, e.g., M1

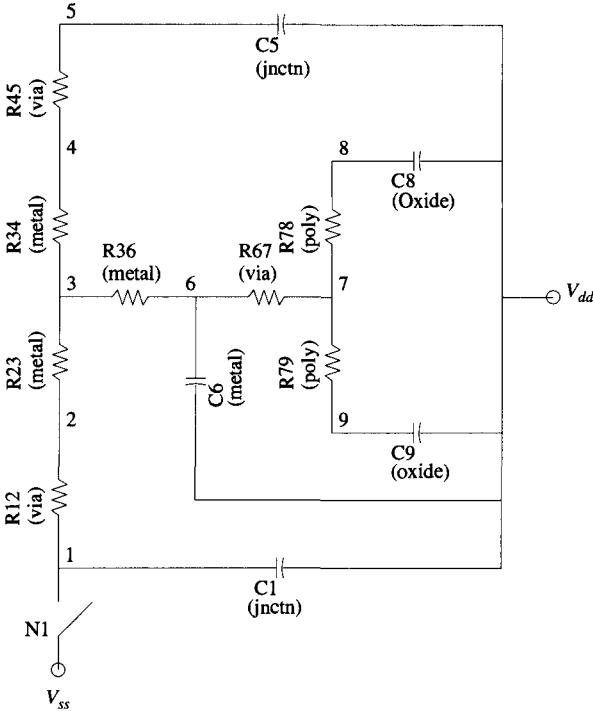


Figure 20.6 Signal path as an RC network.

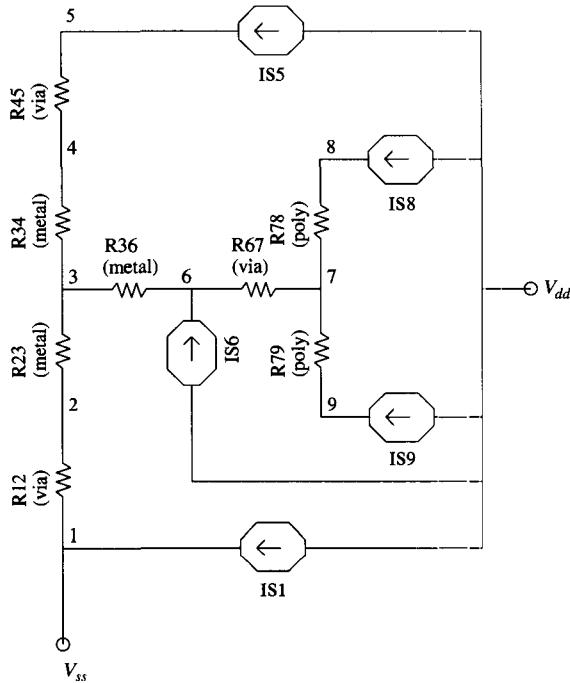
to polysilicon and  $n^+/p^+$  silicon (M1C). The capacitors are defined by capacitive loading of the junctions and gates, as well as interconnect self-loading. The RC network in this example is simplified for our purpose. The network actually extracted from layout using CAD tools (e.g., HILEX, REX, and CUP [31]) would be somewhat more complicated with more components.

In Fig. 20.7 the capacitors are replaced by current sources. The current of each source is set to the capacitance of the capacitor it replaces times the voltage swing ( $V_{dd}$ ) divided by the cycle time ( $t_{cycle}$ ). It is then straightforward to solve Kirchhoff's laws for the node voltages and branch currents using a computationally efficient sparse matrix solver.

A similar analysis can be performed for the case where the signal node is charged through the PMOS device. For more complicated circuits, there may be multiple paths for charging and discharging the signal line, and the currents can be calculated for each of these cases. The effective average current  $I_{eff}$  can then be determined based on Eq. (20.18).

The analysis of the power and ground grids is carried out a little differently. The goal is to determine the current in each interconnect segment in the network path between a transistor driver and some perfect voltage reference point. Ideally this reference would be the  $V_{dd}$  and  $V_{ss}$  pads, but because of the sheer size of the power grids in large chips, this reference might sometimes be taken to be the point at which the network contacts a very wide, thick upper-level metal layer.

At the places where they connect to the resistor ladder network, the transistors are replaced by current sources. The current of each source is set to the total capacitance of the node that the transistor must drive  $C_{load}$  times  $V_{dd}$  divided by  $t_{cycle}$  times a node activity factor  $f$  ( $0 \leq f \leq 1$ ). This factor represents the average typical frequency at



**Figure 20.7** Signal path resistor network with the capacitors in Fig. 20.6 replaced by current sources.

which the driver will switch relative to the operating frequency (clock), and can be determined by logic simulators (e.g., CHANGO [31]). Connecting each of the sources to the power grid individually, the large linear system can be solved for the node voltages and branch currents in each interconnect segment. The total current can then be calculated by superposition.

Detailed analysis as described above of all of the signal networks on a chip is not practical or necessary. Therefore, a filtering step is performed to identify those networks that will not be susceptible to electromigration failure. These low-risk networks can then be screened from further analysis. For example, let  $I_{lim}$  be defined as that value of current that will yield an acceptably low level of electromigration risk even for the most susceptible interconnect element, e.g., the thinnest metal or smallest via. One simple screening algorithm, then, is to eliminate all networks with a total load capacitance given by [32]

$$C_{load} \leq \frac{I_{lim} t_{cycle}}{f V_{dd}} \quad (20.20)$$

### 20.4.3 CAD Reliability Analysis

For reliability analysis, statistical modeling parameters describing the electromigration lifetime distribution for each class of interconnect needs to be specified. The extraction tool classifies each interconnect component based on the layout information. Each component might naturally be grouped by layout layer and type of interconnect, for example, Metal-1, Via-1, Metal-2, Via-2. Further, each metal layer might be binned according to line width with the corresponding statistical parameters reflecting the effect of line width on electromigration lifetime.

We assume that the failure distribution for a single interconnect element, that is, a single via or a line of a certain unit length, is well represented by lognormal statistics. Then the cumulative probability of failure at time  $t$  for a single element in the  $i$ th structure class is

$$F_i(t) = \Phi\left(\frac{\ln(t/\theta_i)}{\sigma_i}\right) \quad (20.21)$$

where  $\theta_i$  is the estimated median time-to-failure given by Eq. (20.5) normalized to the design standard effective current  $I_{std}$  and temperature  $T_{std}$ .

For each structure class the estimated composite electromigration risk  $R_i$  over the chip product life  $\tau$  is computed based on a series reliability model for statistically independent failure. This implies that any interconnect failure results in chip failure. Therefore,

$$R_i = 1 - \prod_{j=1}^{m_i} \left[ 1 - \Phi\left(\frac{\ln(S_{ij}^{n_j} Y_{ij} \tau / \theta_i)}{\sigma_i}\right)\right]^{N_{ij}} \quad (20.22)$$

The  $S_{ij}$  and  $Y_{ij}$  are simply the stress scaling factors due to current and temperature, respectively, given by

$$S_{ij} = \frac{I_j}{I_{std}} \quad \text{and} \quad Y_{ij} = \exp\left(\frac{Q_i}{k} \left[\frac{1}{T_{std}} - \frac{1}{T_j}\right]\right) \quad (20.23)$$

$N_{ij}$  is the number of elemental units (i.e., the number of unit lengths or number of contacts) present on the chip at the stress combination  $S_{ij}$  and  $Y_{ij}$ . There are  $m_i$  stress combinations in the  $i$ th structure class. Figure 20.8 shows examples of the distribution in the  $S_{ij}$  for three classes of interconnect on a large microprocessor chip. The effective average currents for determining the  $S_{ij}$  were computed using the techniques described in Section 20.4.2.

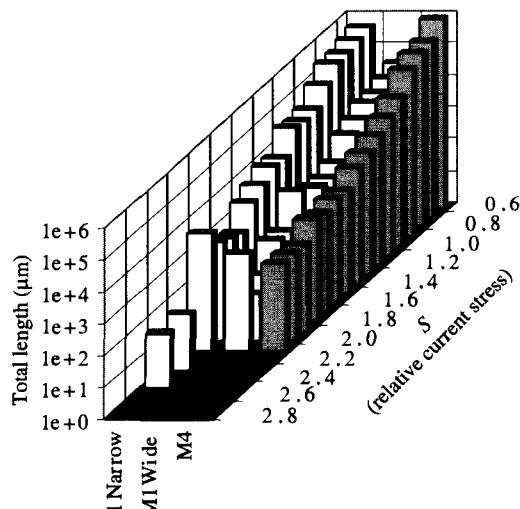


Figure 20.8 Distribution of  $S_{ij}$  for the Alpha 21164 at 333 MHz, 100°C. (From [27].)

The total chip electromigration reliability risk calculated for all  $k$  interconnect classes is

$$R = 1 - \prod_{i=1}^k [1 - R_i] \quad (20.24)$$

In Fig. 20.9 the electromigration reliability risk associated with each of the three classes of interconnect shown in Fig. 20.8 has been calculated and is given along with the total reliability risk for all three classes combined.

The results of this type of reliability assessment can identify those interconnect elements with the greatest risk, and this information can then be used by the designers to modify the circuit layout as necessary to meet the chip reliability goals. Figure 20.10 demonstrates a graphical interface for reporting the results of the electromigration reliability analysis. In this figure the  $V_{dd}$  grid for the chip is shown in various gray tones corresponding to the various levels of interconnect. The electromigration violations are superimposed on the grid as bright white spots.

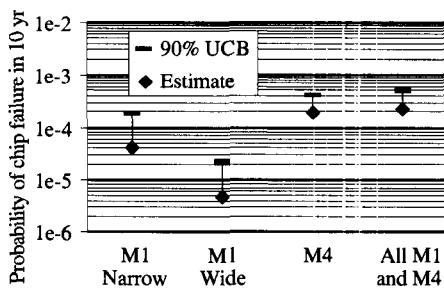


Figure 20.9 Electromigration risk assessment for the Alpha 21164 at 333 MHz, 100°C based on the current stress in the interconnect classes given in Fig. 20.8. (From [27].)

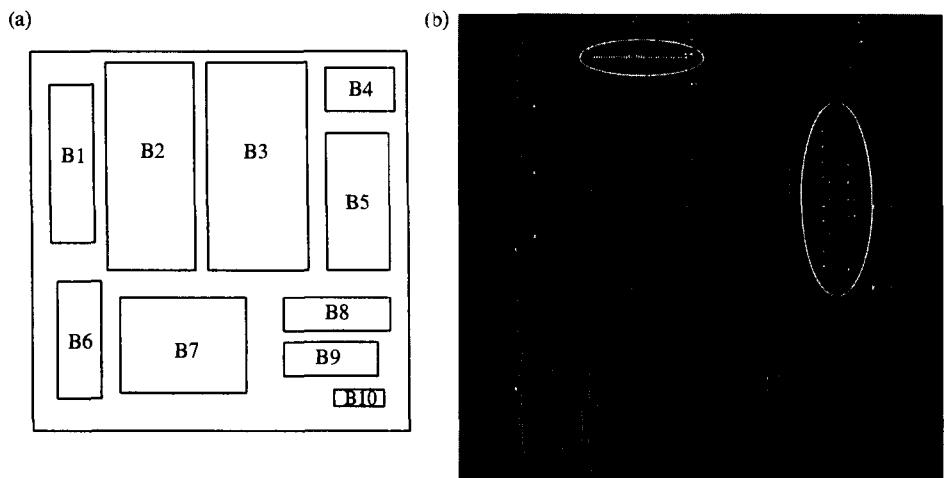


Figure 20.10 A 1.7 million transistor multimedia chip example: (a) block diagram, and (b)  $V_{dd}$  grid with electromigration violations shown as bright spots. The errors at the upper part of block B2 and in block B5 on the right side of the chip are due to block B3 not being correctly connected to the power grid. Since B3 is not connected to the power rail directly, power to block B3 must be routed through blocks B2 and B5, creating electromigration design violations in interconnects not designed to handle the resulting high currents. (From [30].)

The risk assessment methodology described above is based on a generalized form of the Black electromigration lifetime acceleration model, namely the average current recovery model. However, there is another electromigration failure criterion that can be used—the Blech critical threshold described in Section 20.3.3. A methodology whereby this failure criterion might be implemented in CAD tools to evaluate complex circuit layout was recently described [33]. Interconnects that do not meet the critical threshold criterion can be filtered from further reliability analysis, thereby simplifying the task. Further, this can potentially give designers added flexibility to use larger currents in very short lengths of metal.

#### 20.4.4 RMS Current Limits

The effective average current may be very small in signal lines with bidirectional current waveforms, resulting in very little incremental electromigration risk in the SEB calculation. However, this does not mean that we can allow the peak currents to be arbitrarily large even though the average current approaches zero, because of the risk of temperature-gradient failures due to excessive Joule heating. Since the temperature increase is proportional to the square of the rms current, it is therefore necessary to limit the rms current.

In this regard, the SPICE-like circuit simulation tools have an advantage in that they are able to accurately determine the rms current through the interconnect. However, it is possible to make a conservative estimate of the rms current in each interconnect element on the entire chip using the average currents calculated from the RC network analysis described above. Let  $I_+$  and  $I_-$  denote the maximum positive and negative time-averaged currents, respectively, obtained from the signal network analysis under charging and discharging situations. The rms current is then given by

$$I_{rms}^2 = \gamma(I_+^2 + I_-^2) \frac{t_{cycle}}{t_{rise}} \quad (20.25)$$

where  $t_{rise}$  is taken to be the minimum voltage rise time for a chain of wide-channel, self-loaded gates, and  $t_{rise} \approx t_{fall}$ .  $\gamma$  is a shape factor, which is determined by the assumed shape of the current pulse. For a triangular current pulse,  $\gamma = 1.33$ .

If this conservative estimate of  $I_{rms}$  per unit width exceeds the specified design safe operating limit, a more detailed circuit simulation may need to be performed to obtain a more accurate calculation of the current waveform and rms current. Typically only a very small percentage of the interconnects in signal networks is found to violate rms current limits. Ultimately, layout modifications may be required to meet reliability design limits.

### 20.5 CONCLUSION

In this chapter, many of the issues surrounding electromigration reliability in state-of-the-art chip design and fabrication were addressed. The first part of the article reviewed the physical mechanisms underlying the electromigration phenomenon in thin-film interconnects on integrated circuits. Process and materials effects on electromigration were considered.

The two environmental parameters that control electromigration are temperature and current density. The chip temperature is usually fixed by system application constraints, which leaves the current density as the single parameter left for designers to control. While accelerated lifetime testing is typically carried out using dc current stress in order to minimize test times, the interconnects in most circuits are subjected to pulsed current in operation. Data indicate that the electromigration driving force is simply the average current density.

Once the interconnect metallization process is defined and characterized, it is up to the designers to lay out the circuits and interconnects in such a way that the resulting chip meets the product reliability specifications. CAD tools can facilitate electromigration analysis and risk assessment of large, complex ULSI circuits. In the last part of this chapter, some basic techniques that may be implemented in CAD tools for this purpose were outlined.

Analyses of power grids and signal networks to determine the average current in each interconnect element were discussed. The average current can then be used to calculate a risk for each interconnect element, and the individual risks combined to assess the overall chip reliability risk. Since the effective average current in signal lines with bidirectional current can be vanishingly small, it is necessary to check the rms current as well, to limit Joule heating effects. The results of this type of reliability assessment can be used by designers to identify interconnect elements with the greatest risk, and the circuit layout modified as necessary to meet the chip reliability goals.

By far, the workhorse materials used in integrated circuit metallization have been, and continue to be, aluminum-based interconnects and silicon dioxide interlevel dielectrics. The need for improving high-speed ULSI circuit performance is driving the development of new interconnect metallization processes utilizing copper and low-k dielectrics. The knowledge gained from over three decades of work investigating electromigration issues in thin-film aluminum and aluminum alloys will be extremely valuable in evaluating the reliability of new copper-based metallizations. With the continuing evolution of integrated circuit technology, there will always be new and challenging problems in interconnect reliability assurance.

## ACKNOWLEDGMENTS

I would like to thank Professor Carl Thompson (MIT) and Jim Lloyd (JPL) for many helpful discussions. I am also grateful to John Kitchin (Compaq) for providing Figs. 20.8 and 20.9 and to Steffen Rochel (Simplex) for Fig. 20.10.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL8500.

## REFERENCES

- [1] I. A. Blech and H. Sello, "The Failure of Thin Aluminum Current-carrying Strips on Oxidized Silicon," *Physics of Failure in Electronics*, vol. 5, USAF-RADC Series, p. 496–505, 1966.
- [2] J. R. Black, "Mass Transport of Aluminum by Momentum Exchange with Conducting Electrons," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 148–159, 1967.

- [3] I. Ames, F. M. d'Heurle, and R. E. Horstmann, "Reduction of Electromigration in Aluminum Films by Copper Doping," *IBM J. Res. Develop.*, vol. 14, pp. 461–463, July 1970.
- [4] J. R. Black, "Electromigration Failure Modes in Aluminum Metallization for Semiconductor Devices," *Proc. IEEE*, vol. 57, pp. 1587–1594, Sept. 1969.
- [5] J. R. Lloyd and P. M. Smith, "The Effect of Passivation Thickness on the Electromigration Lifetime of Al/Cu Thin Film Conductors," *J. Vac. Sci. Technol.*, vol. A1, pp. 455–458, 1983.
- [6] F. G. Yost, D. E. Amos, and A. D. Romig, Jr., "Stress-Driven Diffusive Voiding of Aluminum Conductor Lines," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 193–201, 1989.
- [7] J. R. Lloyd, "Electromigration in Al-Cu Thin Films with Polyimide Passivation," *Thin Solid Films*, vol. 91, pp. 175–182, 1982.
- [8] T. Usui, T. Watanabe, S. Ito, M. Hasunuma, M. Kawai, and H. Kaneko, "Significant Improvement in Electromigration of Reflow-sputtered Al-0.5wt%Cu/Nb-liner Dual Damascene Interconnects with Low-k Organic SOG Dielectric," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 221–226, 1999.
- [9] M. J. Attardo and R. Rosenberg, "Electromigration Damage in Aluminum Film Conductors," *J. Appl. Phys.*, vol. 41, pp. 2381–2386, May 1970.
- [10] B. N. Argarwala, M. J. Attardo, and A. J. Ingraham, "Dependence of Electromigration-induced Failure Time on Length and Width of Aluminum Thin-film Conductors," *J. Appl. Phys.*, vol. 41, pp. 3954–3960, Sep. 1970.
- [11] J. Cho and C. V. Thompson, "Grain Size Dependence of Electromigration-induced Failures in Narrow Interconnects," *Appl. Phys. Lett.*, vol. 54, pp. 2577–2579, June 19, 1989.
- [12] M. L. Dreyer, K. Y. Fu, and C. J. Varker, "The Effects of Temperature and Microstructure on the Components of Electromigration and Mass Transport," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 304–310, 1993.
- [13] J. R. Lloyd and J. J. Clement, "Electromigration in Copper Conductors," *Thin Solid Films*, vol. 262, pp. 135–141, 1995.
- [14] M. A. Korhonen, P. Børgesen, K. N. Tu, and C. -Y. Li, "Stress Evolution Due to Electromigration in Confined Metal Lines," *J. Appl. Phys.*, vol. 73, pp. 3790–3799, April 15, 1993.
- [15] J. J. Clement, "Reliability Analysis for Encapsulated Interconnect Lines under dc and Pulsed dc Current Using a Continuum Electromigration Transport Model," *J. Appl. Phys.*, vol. 82, pp. 5991–6000, Dec. 15, 1997.
- [16] I. A. Blech and C. Herring, "Stress Generation by Electromigration," *Appl. Phys. Lett.*, vol. 29, pp. 131–133, Aug. 1, 1976.
- [17] I. A. Blech and K. L. Tai, "Measurement of Stress Gradients Generated by Electromigration," *Appl. Phys. Lett.*, vol. 30, pp. 387–389, April 15, 1977.
- [18] R. G. Filippi, R. A. Wachnik, H. Aochi, J. R. Lloyd, and M. A. Korhonen, "The Effect of Current Density and Stripe Length on Resistance Saturation During Electromigration Testing," *Appl. Phys. Lett.*, vol. 69, pp. 2350–2352, Oct. 14, 1996.
- [19] P. Børgesen, M. A. Korhonen, D. D. Brown, C. -Y. Li, H. S. Rathore, and P. A. Totta, "Stress Evolution During Stress Migration and Electromigration in Passivated Interconnect Lines," *AIP Conf. Proceedings*, vol. 305, pp. 231–253, 1994.
- [20] J. J. Clement, J. R. Lloyd, and C. V. Thompson, "Failure in Tungsten-filled via Structures," *Mat. Res. Soc. Symp. Proc.*, vol. 391, pp. 423–428, 1995.
- [21] J. M. Towner and E. P. van de Ven, "Aluminum Electromigration under Pulsed dc Conditions," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 36–39, 1983.
- [22] J. A. Maiz, "Characterization of Electromigration under Bidirectional (BC) and Pulsed Unidirectional (PDC) Currents," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 220–228, 1989.
- [23] L. M. Ting, J. S. May, W. R. Hunter, and J. W. McPherson, "AC Electromigration Characterization and Modeling of Multilayered Interconnects," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 311–316, 1993.

- [24] J. R. Lloyd and R. H. Koch, "Study of Electromigration-induced Resistance and Resistance Decay in Al Thin Film Conductors," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 161–168, 1987.
- [25] I. A. Blech and E. S. Meieran, "Direct Transmission Electron Microscope Observation of Electrotransport in Aluminum Thin Films," *Appl. Phys. Lett.*, vol. 11, pp. 263–266, Oct. 15, 1967; "Electromigration in Thin Al Films," *J. Appl. Phys.*, vol. 40, pp. 485–491, Feb. 1969.
- [26] P. B. Ghate, "Electromigration-induced Failures in VLSI Interconnects," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 292–299, 1982.
- [27] J. Kitchin, "Statistical Electromigration Budgeting for Reliable Design and Verification in a 300-MHz Microprocessor," *Proc. Symp. on VLSI Circuits*, pp. 115–116, 1995.
- [28] R. H. Tu, E. Rosenbaum, W. Y. Chan, C. C. Li, E. Minami, K. Quader, P. K. Ko, and C. Hu, "Berkeley Reliability Tools---BERT," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1524–1534, Oct. 1993.
- [29] C. -C. Teng, Y. -K. Cheng, E. Rosenbaum, and S. -M. Kang, "iTEM: A Temperature-Dependent Electromigration Reliability Diagnosis Tool," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 882–892, Aug. 1997.
- [30] S. Rochel, G. Steele, J. R. Lloyd, S. Z. Hussain, and D. Overhauser, "Full-Chip Reliability Analysis," *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 356–362, 1998.
- [31] V. Peng, D. R. Donchin, and Y. T. Yen, "Design Methodology and CAD Tools for the NVAX Microprocessor," *Proc. IEEE Int. Conf. on Computer Design: VLSI in Computers and Microprocessors*, pp. 310–313, 1992.
- [32] J. Wright and C. Edmondson, "Metal Migration Considerations in Signal Lines," *Proc. IEEE 1987 Custom Integrated Circuits Conf.*, pp. 594–597, 1987.
- [33] J. J. Clement, S. P. Riege, R. Cvijetic, and C. V. Thompson, "Methodology for Critical Threshold Design Rule Evaluation," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 576–581, May 1998.

Kaizad Mistry  
Intel Corporation

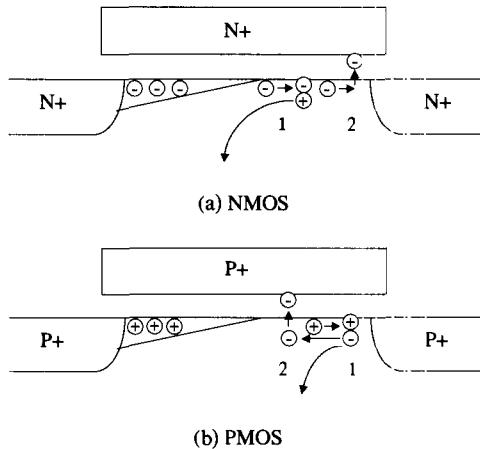
Hot carrier degradation is one of the reliability mechanisms that constrains high-performance microprocessor design. Hot carrier reliability considerations limit the voltages applied to transistors, and add to the complexity of the circuit design task. However, these constraints must be met in order to achieve a reliable high-performance design that is viable in the marketplace. In this chapter, we will review the origin and physics of the hot carrier degradation phenomenon, discuss the effects of the degradation on circuit performance as well as methods of modeling the degradation. Finally, we will explore circuit design guidelines and design tools to guarantee a reliable design, and describe an example analysis of a section of a microprocessor.

## 21.1 WHAT ARE HOT CARRIERS?

When an MOS transistor is ON, charge carriers (electrons in n-MOSFETs and holes in p-MOSFETs) are accelerated from the source toward the drain due to the electric field. The average energy of carriers is thus increased. As a result of scattering processes, some carriers will have energy higher than average and some lower. These high-energy carriers are often referred to as *hot carriers*; for a good description of the carrier heating process, see Frey [1].

Why are these hot carriers a concern? When they have sufficient energy (about 3 eV for electrons and 4 eV for holes), hot carriers can be injected into the gate oxide of the transistor. Most of these injected carriers are transported across the gate oxide and are measured as gate current. However, a small fraction can cause damage to the gate oxide or to the oxide–silicon interface. This damage takes the form of electrically charged trap sites that degrade the performance of the MOSFET and hence of the circuit. When a MOSFET operates as a good switch, the drain-source current in the OFF state is small while the current in the ON state is large. The typical effect of hot carrier degradation in n-MOSFETs is to degrade the ON current, while in p-MOSFETs the OFF current is increased. Either one of these changes can affect circuit performance. In the rest of this section, we will examine carrier injection in more detail.

Figure 21.1 shows a schematic illustration of hot carrier generation in n-channel and p-channel MOSFETs. Let us consider the n-MOSFET first. As electrons are accelerated from the source toward the drain, they gain energy from the lateral electric field set up under high drain bias. For an n-MOSFET biased in saturation, the lateral field rises gradually from the source side toward the drain side of the channel, with a sharp peak near the drain junction region where the electron velocity is saturated. In this high field region, the electrons can gain a large amount of energy. A fraction of the hot carriers will



**Figure 21.1** Schematic cross section illustrating carrier heating and injection for (a) n-channel MOSFET: Electrons are accelerated by the electric field toward the drain and gain energy, resulting in impact ionization (1) and carrier injection into the gate (2). (b) p-channel MOSFET: Holes are accelerated towards the drain and gain energy, resulting in impact ionization (1). Electrons created by impact ionization are accelerated back toward the source, gaining energy, and are injected into the gate (2).

have energies exceeding one of two critical energy thresholds: the energy required for impact ionization, and the energy required for carrier injection, both discussed next.

A fraction of the electrons will gain sufficient energy to cause impact ionization, or the creation of an electron-hole pair: the energetic electron excites an electron from the valence band into the conduction band, leaving a hole in the valence band. The holes are measured as a substrate current,  $I_b$ . This substrate current, a measure of hot carrier generation, is given by [2]

$$I_b = K1 \cdot I_d \cdot \exp(\Phi_i/q \cdot \lambda \cdot E) \quad (21.1)$$

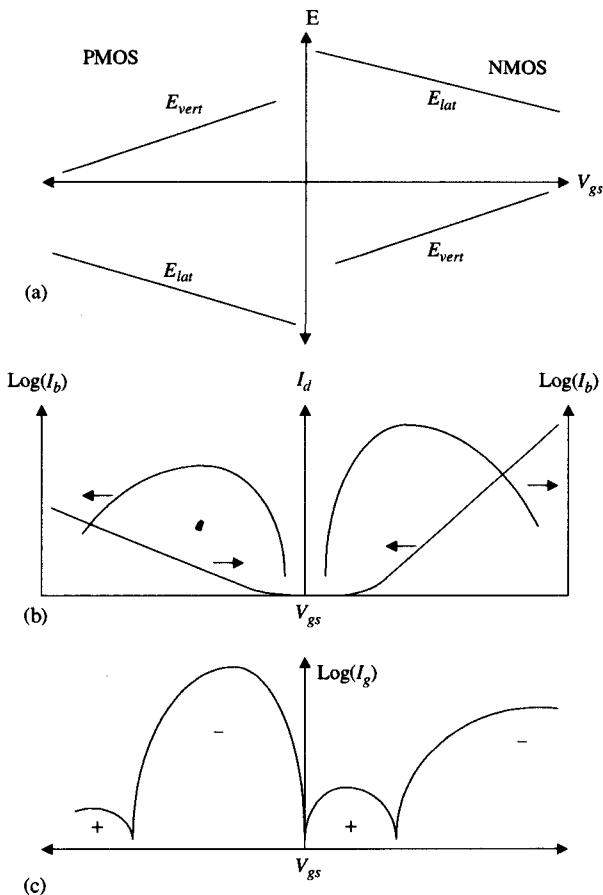
where  $I_d$  is the drain current,  $K1$  is a constant,  $\Phi_i$  is the threshold energy to cause impact ionization ( $\sim 1.3$  eV),  $\lambda$  is the mean free path,  $E$  is the peak lateral electric field, and  $q$  is the electronic charge. Figure 21.2 shows the gate bias dependence of the lateral and vertical electric fields in MOSFETs, along with the drain, substrate, and gate currents. As shown in Fig. 21.2(b), the substrate current initially rises as  $V_{gs}$  is increased above threshold, as the number of inversion electrons available for impact ionization increases. At higher  $V_{gs}$  values, the substrate current is reduced, as the lateral electric field is reduced.

When carriers have even larger energy, they may be able to surmount the oxide-silicon energy barrier and be injected into the gate oxide. The gate current characteristics of n-MOSFETs at high drain bias is shown in Fig. 21.2(c). Both the lateral and the vertical electric field play a role in defining these characteristics [Fig. 21.2(a)]. At high gate voltage, this gate current is negative; that is, electrons are injected into the gate oxide from the channel and are collected at the gate electrode. These hot electrons are injected into the gate and are only weakly repelled by the small vertical electric field. As the gate bias is reduced, the lateral field is increased, but the vertical field is larger and repels electrons from the gate, so that the gate current is reduced. The electronic gate current is given by [2]

$$I_g, e = K2 \cdot I_d \cdot \exp(\Phi_e/q \cdot \lambda \cdot E) \quad (21.2)$$

where  $K2$  is a constant and  $\Phi_e$  is the energy barrier for electron injection into the gate oxide ( $\sim 3.1$  eV).

At still lower gate voltages, the gate current changes sign as holes are injected into the oxide. Recall that holes were created by energetic electrons through the process of



**Figure 21.2** Carrier injection in MOSFETs. (a) Lateral and vertical fields vs.  $V_{gs}$  at high  $V_{ds}$ . A positive lateral electric field attracts electrons toward the drain. A positive vertical field attracts electrons toward the gate. (b) Drain and substrate currents in n-channel and p-channel MOSFETs versus  $V_{gs}$ . (c) Gate currents in n-channel and p-channel MOSFETs versus  $V_{gs}$ . “+” symbols indicate holes injected into the oxide. “-” symbols indicate electron injection.

impact ionization. These holes are themselves accelerated (back toward the source) by the high lateral electric field and can become hot. If they have sufficient energy, the holes are injected into the gate oxide. The vertical field is favorable for holes to be attracted toward the gate electrode, so that even though the number of holes in the channel is smaller, and its injection probability lower, the hole gate current dominates at low gate voltage. The hole gate current is given by

$$I_g, h = K_3 \cdot I_b \cdot \exp(\Phi_h/q \cdot \lambda \cdot E) \quad (21.3)$$

where  $K_3$  is a constant and  $\Phi_h$  is the energy barrier for hole injection into the gate oxide ( $\sim 4\text{ eV}$ ). Over a large range of gate voltages in the middle of the gate voltage range, both electrons and holes are injected, but the gate current is dominated by whichever is algebraically larger.

For p-MOSFETs, similar arguments apply, only the polarities and hence the sign of the carriers are reversed. Both holes and electrons are also injected into the gate for p-MOSFETs, but under opposite bias regimes. The lateral electric field accelerates holes toward the drain and some of these holes create electron-hole pairs through impact ionization. At high  $|V_{gs}|$ , some of these holes also have sufficient energy to be injected into the gate oxide. Because of the higher energy required for hole injection than for

electron injection, this gate current is much smaller than the electron gate current in n-MOSFETs. And, because the vertical electric field near the drain end of p-MOSFETs with  $|V_{gs}|$  less than  $|V_{ds}|$  repels these holes, the hole gate current is even smaller than that for comparable n-MOSFETs at low  $V_{gs}$ . At lower  $|V_{gs}|$ , the hole gate current is reduced by the increasing repulsive force of the vertical electric field.

At even lower  $|V_{gs}|$ , the gate current changes sign as electrons are injected into the oxide of the p-MOSFET. Electrons created by energetic holes through impact ionization are accelerated back toward the source by the high lateral electric field and become hot. If they have sufficient energy, these electrons are injected into the gate oxide. The vertical field is favorable for electrons to be attracted toward the gate electrode, and the barrier for electron injection is lower than that for hole injection, so that the electron gate current is much larger than the hole gate current. Indeed, the electron gate current in p-MOSFETs is even larger than the electron injection in comparable n-MOSFETs!

## 21.2 HOW DO HOT CARRIERS DEGRADE MOSFETS?

Since the hot carrier reliability effect was first reported [3]–[5], a debate has raged as to the mechanisms of damage [2]–[11]. Do hot holes or hot electrons create the damage? Is the damage due to traps in the oxide or at the oxide–silicon interface? While that debate was never formally laid to rest, the picture presented in this section represents one attempt at a consensus [12]. First some basics. Traps in the oxide are defects that can capture carriers and thereby alter their charge state. *Oxide electron* traps can capture electrons and for the purposes of this work can be considered neutral when empty and negatively charged when full; likewise, *oxide hole* traps are considered neutral when empty and positively charged when they capture a hole. *Interface* traps are defects at the oxide–silicon interface. These traps are distinct from oxide traps in that they respond to the fermi level in the silicon. When the silicon energy bands are bent such that the fermi level is above the trap level, the trap is occupied, and when the fermi level is below it is empty. Generally, traps in the upper half of the bandgap are neutral when empty and are negatively charged when full, affecting n-MOSFETs. Traps in the lower half are neutral when empty and positively charged when full, affecting p-MOSFETs.

The dominant damage type for n-MOSFETs is the creation of interface traps. Interface trap creation is largest at low- and mid-gate voltages when the drain voltage is high. At low-gate voltages, interface traps are created directly by energetic holes. In the mid-gate voltage range interface traps are created from the energy released when an electron and a hole recombine.

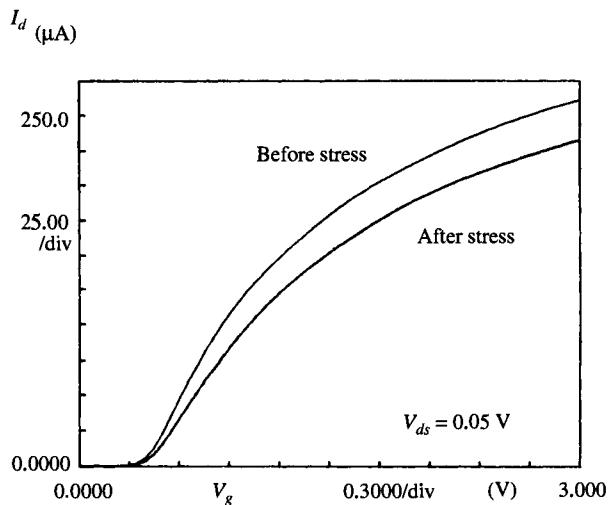
In the mid-gate voltage range both electrons and holes are injected into the oxide. A fraction of the injected holes is captured by oxide traps situated close to the oxide–silicon interface. These interfacial trapped holes are positively charged and so have a large probability of capturing electrons that are also being injected into the oxide. When an interfacial trapped hole is neutralized by electron capture, energy is released. This energy can disrupt chemical bonds at the oxide–silicon interface in such a way as to create an interface trap [13], [14]. This trap creation process requires that the injected holes have energies no greater than that needed to be injected into the oxide; the energy required to create the traps is provided by the hole-electron capture. The bond being broken at the interface is typically an Si–H bond. Thus the presence of hydrogen or water-related species (OH) can affect the rate of interface trap creation, and an excess of these species is avoided in high-quality oxides.

At low-gate voltages, most of the carriers being injected are holes. Because the lateral electric field in the silicon is larger at lower gate voltages, some of the hot holes have energies that significantly exceed the interfacial barrier height ( $\sim 4\text{ eV}$ ); these holes still have excess energy after entering the oxide. Interface traps can be generated directly by the injected hot holes when they have enough energy not only to overcome the interfacial energy barrier, but also to disrupt chemical bonds near the silicon–silicon dioxide interface.

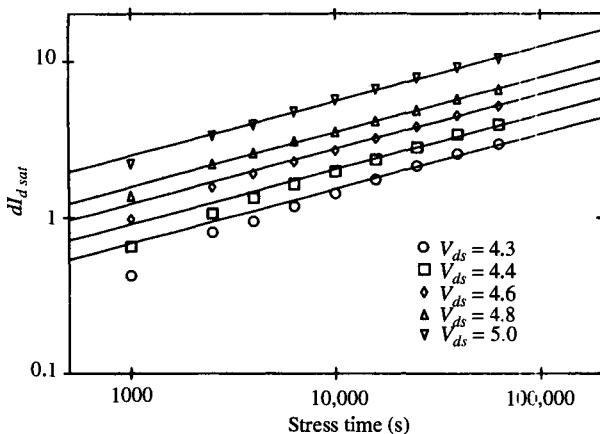
What is the effect of these traps on MOSFET characteristics? The damage created during hot carrier injection is localized near the drain junction region, because that is where the lateral electric field is at a maximum, and hence carriers are the hottest. Recall that the dominant type of damage for n-MOSFETs stressed at mid-gate voltages are interface traps. When the MOSFET is on, the silicon bands are bent so that the fermi level is above the interface trap level and the interface traps have a negative charge. One way to look at the impact of this negative charge distribution located near the drain end of the MOSFET is to divide the MOSFET into two regions: the undamaged region and the region containing the damage. The negative charge in the damaged region will result in a larger threshold voltage than in the undamaged region. Thus, when the undamaged part of the MOSFET is strongly ON, the damaged part may only be weakly ON, so that the net MOSFET current in the ON state is smaller than it was before the damage was created. Figure 21.3 shows  $I_d - V_g$  curves for a MOSFET prior to and after stress. The MOSFET threshold voltage has increased and the drive current has decreased as a result of the stress. Figure 21.4 shows the percentage decrease in drive current plotted as a function of stress time on a log-log scale. A straight line results, indicating a time power law behavior for degradation vs. stress time [15] with a power of 0.5.

$$\Delta I_d \propto t^{0.5} \quad (21.4)$$

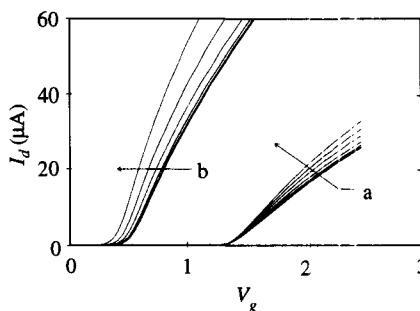
Now let's look at degradation in p-MOSFET's. The dominant type of damage is the trapping of electrons in the gate oxide at low- and mid-gate voltages [17], [18]. What is the effect of the trapped electron charge on p-MOSFET characteristics? The negative charge lowers the local threshold voltage in the drain region where the hot carrier induced damage occurs. As a result, the damaged region turns on before the main undamaged transistor. At threshold, therefore, the transistor appears to have a shorter



**Figure 21.3**  $I_d - V_g$  curves for a  $W = 12\text{ }\mu\text{m}$ ,  $L = 0.75\text{ }\mu\text{m}$  n-MOSFET before and after stress for 1000 seconds at  $V_{ds} = 5.0\text{ V}$  and  $V_{gs} = 1.8\text{ V}$ . [12]



**Figure 21.4** Percentage reduction in  $I_{d\text{sat}}$  vs. stress time for n-MOSFETs stressed at accelerated drain voltages and mid-gate voltages. [12]



**Figure 21.5**  $I_d - V_g$  curves for two p-MOSFET's stressed at high  $V_{ds}$  and low  $V_{gs}$ . (a) Gate length = 0.8  $\mu\text{m}$  and (b) gate length = 0.3  $\mu\text{m}$ . The curves for the longer device have been shifted along the x-axis for clarity. [17]

effective length. This results in an increase in the MOSFET drive current. For short-channel MOSFETs the negative charge can also lower the threshold voltage of the overall device, causing increased transistor OFF current. These characteristics are shown in Fig. 21.5, which illustrates  $I_d - V_g$  curves for p-MOSFETs as a function of stress time. Curves (a) for a longer device show that MOSFET drive current is increased with stress but that the threshold voltage is not changed. Curves (b) for a short device show both an increase in drive current as well as a lowering of the threshold voltage.

To summarize what we have discussed so far, n-MOSFET drive current is reduced as a consequence of hot carrier stress, while p-MOSFET OFF current can be increased. Paradoxically, the predominant carrier responsible for damage in n-MOSFET is the hot hole, while in p-MOSFETs it is the hot electron. For both n- and p-MOSFETs, damage is worst at low- to mid-gate voltages, and increases exponentially with drain voltage.

## 21.3 MODELING THE DEGRADATION

Most circuits are designed to tolerate some device skew and will still function adequately even if there are small shifts in MOSFET characteristics. However, when the MOSFET threshold voltage, ON current, or OFF current is degraded beyond some critical value, the circuit may no longer function properly. The level of degradation that can be tolerated is generally a function of both the circuit design and of the testing methodology. Typically, one of two methods is used to determine if a circuit will or will

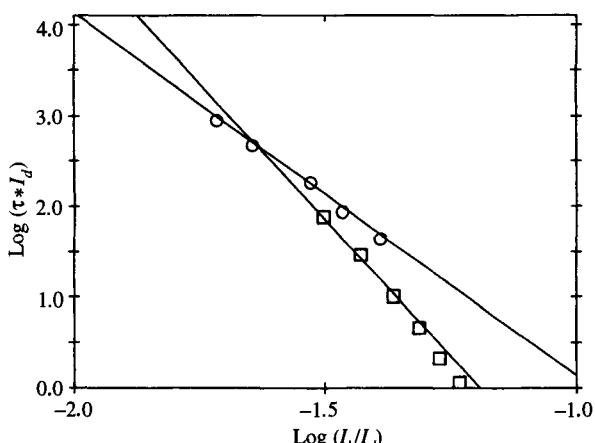
not fail. In the first method, it is assumed that a circuit will fail after a certain amount of degradation. The time-to-fail is then calculated for the MOSFET based on the bias conditions during use and on the failure definition. In the second method, a circuit simulation is employed wherein the degraded MOSFET parameters are used to determine the effect on circuit performance and functionality.

As we shall see later in this chapter, only a very small fraction of MOSFETs in a circuit get exposed to voltages significantly larger than nominal; these few transistors are the ones that will dominate the degradation response of the circuit. If a critical path is composed of, say, 10 delay stages, the odds are generally very small that transistors in more than one of the stages will be subjected to stress levels causing significant degradation. Therefore, if we allow that one stage to degrade, say 10%, then the overall path delay will not degrade by more than 1%, assuming each delay stage contributes equally to the path delay. Therefore, for most digital circuits, n-MOSFETs are considered to have degraded to failure when their saturated drain current degrades by 5% to 10% and their threshold voltage degrades 50–100 mV. Many analog circuits and other types of sensitive circuits can tolerate much less degradation in device parameters. For many such applications, the acceptable degradation in threshold voltage is less than 10 mV and in drive current less than 1%. Finally, p-MOSFET degradation limits will depend on how p-MOSFETs are used in circuits. Typically, up to a 10% increase in drive current or a 2–3 × increase in OFF current is tolerable for most digital circuits. The time required for the MOSFET to reach the failure criterion is referred to as the hot carrier failure time,  $\tau$ .

For n-MOSFETs, we will write two separate equations for  $\tau$  for the low- and mid-gate voltage ranges, respectively. Recall that in the medium-gate voltage range, interface traps are created as a result of the energy released when an interfacial trapped hole captures an injected electron. The failure time is [2]

$$1/\tau_{mid} = A \cdot I_d \cdot (I_b/I_d)^m = A \cdot I_b \cdot (I_b/I_d)^{(m-1)} \quad (21.5)$$

where  $A$  and  $m$  are empirical constants, and  $I_d$  and  $I_b$  are the drain and substrate currents, respectively. Accordingly, a plot of  $\log(\tau \cdot I_d)$  versus  $\log(I_b/I_d)$  should be a straight line with slope  $-m$ . Figure 21.6 shows the failure times,  $\tau$ , defined as a 5% degradation in  $I_{d,sat}$ , for the devices of Fig. 21.4, plotted as such. Indeed, a straight line results, with a slope of  $\sim 4$ .



**Figure 21.6** Failure times,  $\tau$ , plotted according to Eq. (21.5) for NMOS devices stressed at mid-gate voltages from Fig. 21.4 (circles), as well as for devices stressed at low-gate voltages (squares). [12]

While Eq. (21.5) is empirical, it has some basis in physics. Recall that at mid-gate voltages, interface traps are created due to the simultaneous injection of holes and electrons. Because the injection barrier for holes is larger than that for electrons, this larger energy barrier dominates the process. The reciprocal of the reliability lifetime will be proportional to the rate of hole injection into the oxide, given by Eq. (21.3). Combining Eqs. (21.1) and (21.3), we can write

$$1/\tau \propto 1/I_g, h \propto I_b \cdot (I_b/I_d)^{(\phi_h/\phi_i)} \quad (21.6)$$

Note that this has the same form as Eq. (21.5), with  $(m - 1) = (\phi_h/\phi_i)$ , and that the empirical value of  $(m - 1) * \phi_i = 3.9$  eV is in good agreement with the 4 eV barrier for hole injection. In other words,  $I_b$  represents the number of holes available in the channel to be heated, while the ratio  $(I_b/I_d)^{(m-1)}$  represents the fraction of these holes with sufficient energy to be injected into the oxide. Their product is a measure of the rate of damage creation.

At low stress gate voltages, interface traps are created directly by hot holes. We write [10]

$$1/\tau_{low} = B \cdot I_d \cdot (I_b/I_d)^n = B \cdot I_b \cdot (I_b/I_d)^{(n-1)} \quad (21.7)$$

This equation has the same form as Eq. (21.5), but has a different exponent,  $n$ . Also shown in Fig. 21.6, are data for NMOS devices stressed at high-drain bias and low-gate bias, and followed by a brief electron injection. Note that these data follow Eq. (21.7), with an empirical value of  $n = 6$ , within the range of 5 to 6 reported [16], [19]. For interface traps to be created directly by hot holes, the holes must have energy greater than that required to merely surmount the oxide potential barrier; they must have excess energy to break bonds and create interface states. Once again,  $I_b$  represents the number of holes available per second in the channel to be heated, while the ratio  $(I_b/I_d)^{(m-1)}$  represents the fraction of these holes with sufficient energy to be injected into the oxide *and then to create interface traps*. Accordingly,  $n > m$ .

For n-MOSFETs under AC stress conditions, three equations are needed. First, Eqs. (21.5) and (21.7) must be integrated over the time period,  $T$ , of the repetitive stress waveform and the currents are functions of time.

$$1/\tau_{mid}^* = (A/T) \int I_d \cdot (I_b/I_d)^m dt \quad (21.8)$$

$$1/\tau_{low}^* = (B/T) \int I_d \cdot (I_b/I_d)^n dt \quad (21.9)$$

Then, the AC stress lifetime is given by [19]

$$1/\tau_{ac} = 1/\tau_{mid}^* + 1/\tau_{low}^* \quad (21.10)$$

Another popular model uses a single equation of the same form [20]

$$1/\tau_{ac} = (C/T) \int I_d \cdot (I_b/I_d)^1 dt \quad (21.11)$$

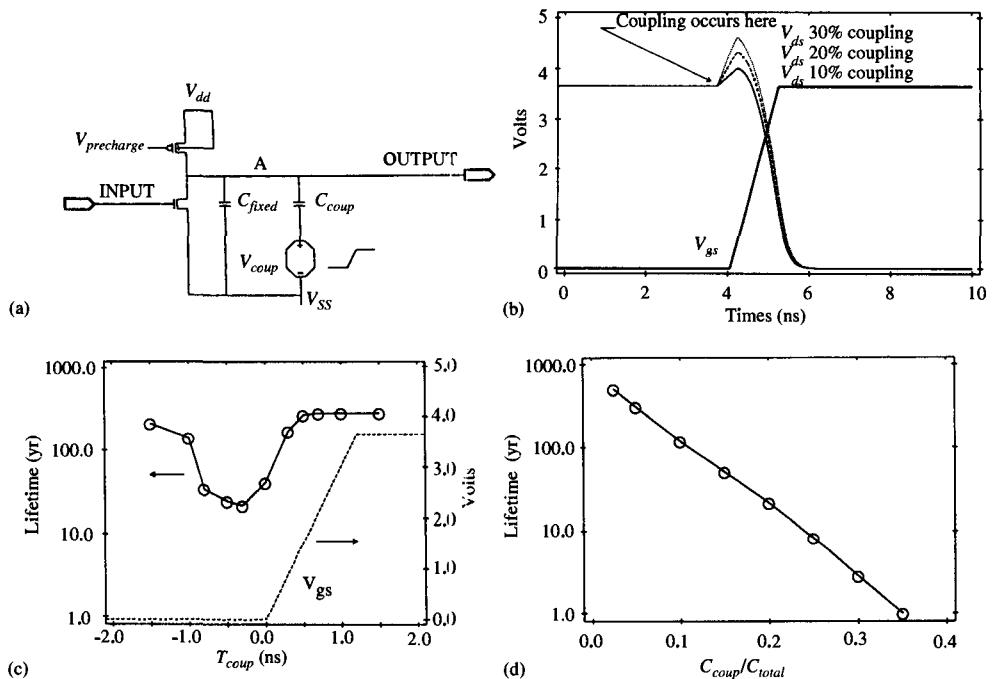
but treats  $C$  and 1 as empirical parameters that are bias dependent. Note that this formulation also requires three equations, Eq. (21.10), plus two equations to model  $C$  and 1 as functions of bias.

## 21.4 CIRCUIT EFFECTS

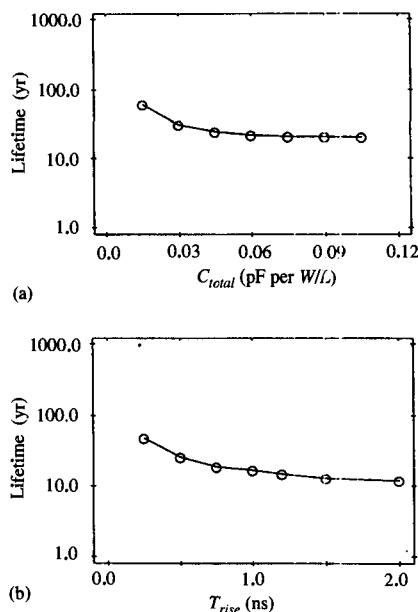
We have examined so far the mechanisms of hot carrier damage and their gate and drain voltage dependencies. For both n-MOSFETs and p-MOSFETs, damage is strongly increased at larger drain biases, because carrier heating is exponentially dependent on the lateral electric field. Also, for both types of MOSFET, damage is worst at low- to mid-gate voltages. Finally, we saw that damage for AC stress can be modeled in a quasi-static fashion, so that the longer the time the transistor spends in a high damage bias condition, the larger the degradation will be. In other words, gates with slow input edges, or large fan-out gates, where output edges are slower, could experience larger degradation compared to gates that switch faster, assuming both have similar voltages and cycle times. In this section, we will examine circuit effects on hot carrier reliability lifetime [21]. First, we will examine the noise environment of the transistor, showing that it has a critical impact on degradation. Next, we will study the impact of input and output edge rates. Finally, we will examine the charge redistribution effect in complex gates. In all cases, a 3.3 V technology is assumed with a 10 ns cycle time. Simulations are carried out at  $V_{DD} = 3.6$  V, allowing for a worst-case supply voltage 10% higher than nominal. Even though the transistors have a very large hot carrier lifetime at this voltage, we shall show that reliability problems can still occur unless care is taken in circuit design.

The noise environment of the circuit has perhaps the largest impact on hot carrier reliability, since voltage noise can increase the voltage applied to the transistor. The noise may originate from power supply ringing, from capacitive coupling due to on-chip interconnect, or from I/O ringing. Here we shall examine the impact of capacitive coupling. Figure 21.7(a) shows the circuit for a simple dynamic gate. Attached to the output node are both a load capacitor  $C_{fixed}$ , and a coupling capacitor,  $C_{coup}$ , representing the parasitic capacitance to a nearby signal node. The total capacitance on the driven node is then  $C_{total} = C_{fixed} + C_{coup}$ . Consider the case where the output is high and the input is low. If the nearby node switches just prior to the input node switching, capacitive coupling will raise the potential of node A above  $V_{DD}$ , as shown in Fig. 21.7(b). The voltage can be as high as 4.5 V, even though the transistor is rated for a nominal 3.3 V. Because the gate of the p-MOS precharge device is at  $V_{DD}$ , it will only weakly clamp the voltage at a threshold above  $V_{DD}$ . This excess voltage appears across the drain and source of the n-MOSFET and can reduce its reliability, depending on the phasing of the coupling event relative to the turn-on of the input to the n-MOSFET. If the coupling event occurs long before the n-MOSFET turns on, even weak p-MOSFET clamping will reduce the voltage before turn-on. If the voltage excursion occurs after the n-MOSFET has turned on,  $V_{ds}$  will already have been decreased from  $V_{DD}$ . However, if the coupling event occurs just before the n-MOSFET is turned on, a dramatic reduction in reliability lifetime can occur, as shown in Fig. 21.7(c). As the coupling ratio is increased, this reduction in lifetime can be dramatic. Figure 21.7(d) shows that in the absence of noise, the MOSFET reliability lifetime is several hundred years. With 35% coupling and worst-case phasing, the lifetime is just over one year! Note that a similar effect can impact p-MOSFETs when a node that is initially low is coupled below ground, increasing the drain-to-source voltage.

Input and output edge rates also play a role, albeit less dramatic than that of coupling noise. Figure 21.8(a) shows the hot carrier failure time for the circuit of



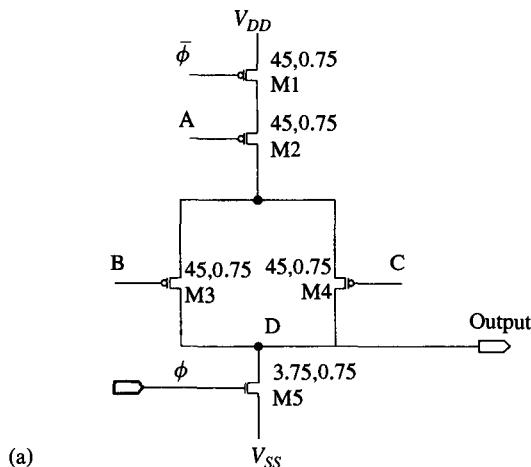
**Figure 21.7** Impact of the circuit noise environment. (a) Circuit for simple dynamic gate. Note the coupling capacitance on node A. (b) SPICE simulation results for  $C_{coup}/C_{total} = 0.1, 0.2$ , and  $0.3$ . (c) Simulated hot carrier failure times versus  $T_{coup}$ , the time at which the coupling event occurs relative to the rising voltage on the gate, and (d) simulated hot carrier failure times versus  $C_{coup}/C_{total}$  assuming worst-case phasing. [21]



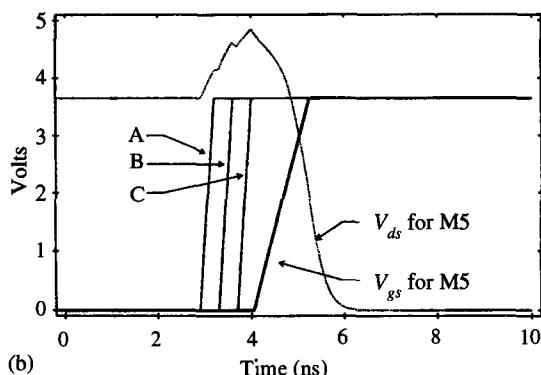
**Figure 21.8** Effect of fan-out and fan-in: (a) Simulated hot carrier failure times for the circuit in Fig. 21.7 versus  $C_{total}$ , with  $C_{coup}/C_{total} = 0.2$ , and (b) Simulated hot carrier failure times versus input rise time with  $C_{total} = 0.06$  pF per  $W/L$  and  $C_{coup}/C_{total} = 0.2$ . [21]

Fig. 21.7 where  $C_{coup}/C_{total}$  is kept constant at 20%, while the total capacitive load is increased. The larger the capacitance, the longer it takes for the drain voltage to be reduced, and so the larger is the damage. Therefore, the reliability lifetime is reduced as fan-out is increased. Figure 21.8(b) shows a similar effect when the input rise time is varied. A slower input pulse also reduces the reliability of the MOSFET.

Charge redistribution effects also can increase the voltage that appears across a MOSFET and thereby reduce reliability. This effect is illustrated with reference to the complex dynamic gate circuit of Fig. 21.9(a). Assume that the clock  $\phi$  is low (transistors M1 and M5 are OFF) and that M2, M3, and M4 are all ON. If M2, M3, and M4 were to be turned off, the stored inversion charge in these transistors cannot find its way to either  $V_{SS}$  or  $V_{DD}$ . Therefore the voltage on the drain of MOSFET M5 must increase, as shown in Fig. 21.9(b), where the voltage gets as high as 4.7 V! While the parasitic p-n junction diodes will reduce this potential eventually, their typical series resistance is so large that the charge bleeds off only slowly. The increase in voltage will dramatically reduce the reliability lifetime of M5 if it were to turn on just after M2, M3, and M4 were turned off. Because charge redistribution effects typically are not checked by automatic reliability tools—which tend to focus on noise and rise/fall times—the designer must pay careful attention to this effect during circuit design.



(a)



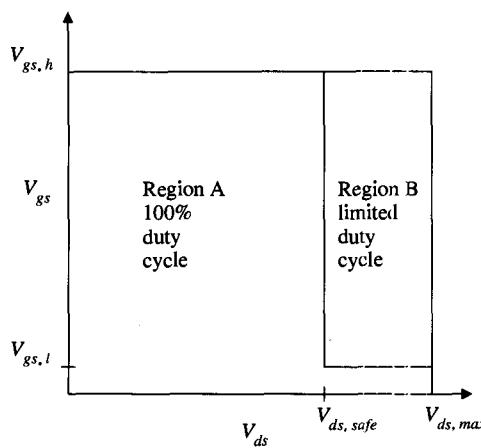
**Figure 21.9** Charge redistribution effect.  
 (a) Circuit for complex dynamic gate, and  
 (b) SPICE simulations for the drain voltage of transistor M5 as transistors M2, M3, and M4 are turned off. [21]

## 21.5 ENSURING CIRCUIT RELIABILITY

So far, we have seen that hot carrier reliability is strongly decreased at larger drain bias. We have also examined circuit effects, and have observed that the noise environment of the transistor, along with the input edge rate and the output load capacitance, play important roles in determining reliability. In this section, we will review how hot carrier reliability guidelines may be formulated [22], and design tools that can aid the circuit designer in ensuring reliable design.

Generally, the circuit designer will be provided with hot carrier design guidelines, which define the voltages that may be applied from the drain to the source of MOSFETs. In addition design tools are provided to assess the amount of damage a transistor will see, and the impact of this damage on circuit performance. Because simulating the impact of damage on circuit performance for all circuits would be very tedious, hot carrier guidelines generally assume a certain level of tolerable transistor level degradation. Figure 21.10 shows how such guidelines can be formulated. Below a voltage  $V_{ds,safe}$ , transistors can be operated at 100% duty cycle. Above this voltage, up to a maximum of  $V_{ds,max}$ , the transistor can be operated at 100% duty cycle if it is OFF, or at limited duty cycle if it is ON. Generally, a table or function is provided describing the allowable duty cycle as a function of bias in this regime. (The maximum-gate voltage is set by gate oxide breakdown and is not covered in this chapter.) One or more sets of such guidelines can be provided with different levels of allowable degradation. For example, a generic version for digital logic can tolerate more degradation and will have larger values of  $V_{ds,safe}$  and  $V_{ds,max}$  than a more restricted version for more sensitive circuits, such as analog circuits, which can tolerate less transistor level degradation and consequently have lower allowed values of  $V_{ds,safe}$  and  $V_{ds,max}$ . These guidelines assist the designer in the initial evaluation of a design at the circuit schematic level, but must be followed up by more exhaustive analysis after the layout has been drawn, using hot carrier analysis tools.

Hot carrier analysis tools fall into one of two categories. The first is a noise analysis tool similar to that used for noise margin analysis [21]. Typically, this tool will extract the coupling noise for each transistor or gate in the circuit, based on capacitance extractions from layout and timing information from a timing simulator. We have seen that the hot carrier reliability is strongly affected by the noise environment of



**Figure 21.10** Typical hot carrier reliability guidelines. Devices can operate up to a voltage  $V_{ds,safe}$  with 100% duty cycle, or at larger voltages up to  $V_{ds,max}$  at limited duty cycle.

the circuit. For noise margin, of course, we are concerned when a low node is coupled upwards or a high node is coupled downwards. For hot carrier reliability, we look for low nodes coupled downwards, or high nodes coupled upwards, since these will increase the drain-to-source voltage of p-MOSFETs and n-MOSFETs. The coupling information, along with the total capacitive load and  $W/L$  of the transistor will determine if the duty cycle versus voltage limitations set by the hot carrier reliability guidelines are met or not. We saw in Fig. 21.7(c) that the relative phasing of noise signals plays an important role in determining hot carrier reliability. Thus it is better if the noise analysis tool comprehends the phase of the coupling signals relative to the transistor input signal. However, certain older tools assume that all switching events occur at the clock edge, a worst-case assumption except for circuits employing cycle stealing techniques. The noise analysis tool will take the millions of transistors on a chip and identify the (hopefully) few transistors that fail the hot carrier guidelines.

These more limited set of circuits are dispositioned manually, using a circuit simulation tool that includes a hot carrier reliability module, such as BERT [20]. These tools are generally based around SPICE or some other circuit simulator. Accurate models for substrate current are included for hot carrier reliability estimates using equations similar to those outlined earlier in this chapter. Model parameters are also fed to these simulators from process reliability characterization of device stressed at accelerated voltages. The simplest form for the output of these simulators is a GO/NOGO result that says the reliability goal was either met or not met; at the next higher level, the simulator predicts the number of years the circuit will function prior to a certain fixed parametric shift. The most useful version, however, is one where the simulator predicts circuit behavior as a function of usage, significantly slowing down transistors exposed to larger stress, not slowing much those that are not exposed to large stress, and indeed speeding up p-MOSFETs that have larger drive current after stress. The net effect on circuit functionality can thus be predicted. Many times, circuits that failed during the noise simulation study may not really be a reliability risk. Circuit simulation may show, for example, that a given circuit can tolerate more transistor level degradation than was assumed in the guidelines. Alternatively, the timing information used for noise analysis may have been less than accurate and more detailed circuit simulation might find that the coupling from two nearby aggressors is slightly out of phase and so is not simply additive. We will examine some cases like this in the next section.

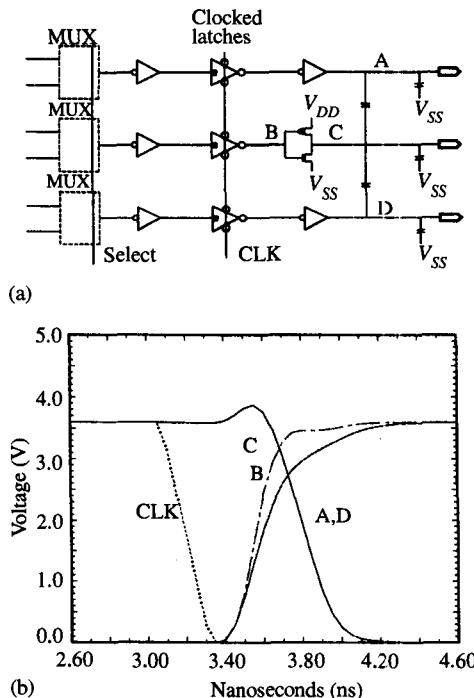
## 21.6 EXAMPLE ANALYSIS

In this section, we will go through a hot carrier analysis example using the instruction box of a high-performance RISC microprocessor designed in a  $0.5\text{ }\mu\text{m}$ ,  $3.3\text{ V}$  technology [21]. The  $V_{ds,max}$  value for this technology was  $4.3\text{ V}$  at a 5% duty cycle. First, a noise analysis program was used to estimate the coupling noise induced maximum  $V_{ds}$  for all n-MOSFETs in the circuit. The proprietary noise analysis program used assumes worst-case phasing for all signals and coupling events. Results for the 25,000 nodes in the circuit are shown in Table 21.1. Capacitive coupling caused 152 nodes (0.6%) to have  $V_{ds} > 4.0\text{ V}$ , and 12 nodes (0.05%) to have voltages greater than the  $V_{ds,max}$  value of  $4.3\text{ V}$ . Each of the 12 nodes with  $V_{ds} > V_{ds,max}$  was analyzed using a circuit reliability simulator and fixed if necessary. Two examples below highlight the analysis.

Figure 21.11(a) shows a portion of an internal bus within the Ibox of the microprocessor. The voltage on node C can be increased through capacitive coupling from

**TABLE 21.1** Coupling Induced  $V_{ds}$  for lbox from a RISC Microprocessor

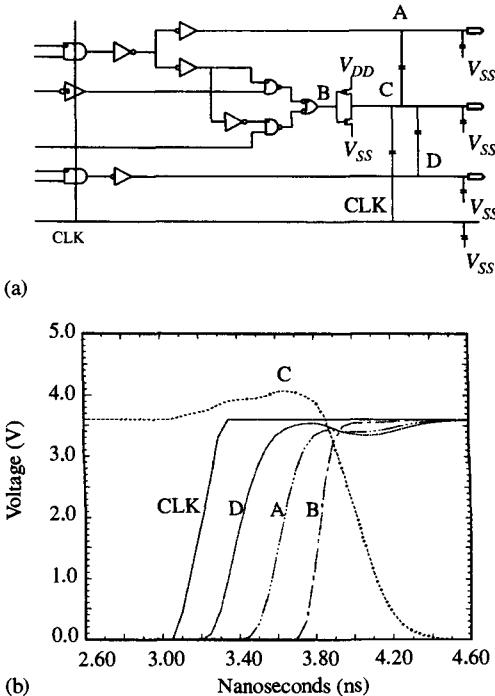
% Effective coupling	Max. $V_{ds}$	Number of nodes
0–4.9	3.78 V	1861
5–9.9	3.96	449
10–14.9	4.14	126
15–19.9	4.32	14
20–24.9	4.5	12
25–29.9	4.68	0



**Figure 21.11** (a) Portion of an internal bus from an *lbox* control circuit. (b) SPICE simulation results where nodes A and D transition at the same time as node B because of asymmetry in the clocked latch. [21]

nodes A and D,  $C_{coup}/C_{total}$  for each of 0.12, for a total of 24% coupling. The design of the clocked latch [23] produces an asymmetry. High-to-low transitions occur one gate delay before low-to-high transitions. Therefore, nodes A and D transition at the same time as node B. Further, since nodes A and D are more heavily loaded than node B, these transitions are slower. The result, shown in Fig. 21.11(b), is that the n-MOSFET pull-down turns on early enough to limit the maximum voltage to 3.9 V, even though the noise simulator estimate was  $> 4.3$  V. This example highlights the importance of the phasing assumptions in the noise simulator.

Figure 21.12(a) shows a portion of random logic from a cache tag control circuit. Node C was routed through a densely packed vertical and horizontal wiring channel. The voltage on node C can increase 26% due to coupling from at least three nodes in one clock phase. Although the voltage excursions caused by the coupling events are additive, they are spread out enough in time so that the static p-MOSFET pull-up can limit the voltage increase on node C to 4.1 V, as shown in Fig. 21.12(b). If nodes A and



**Figure 21.12** (a) Portion of random logic from a cache tag control circuit. (b) SPICE simulation results where the three high-up coupling events are additive but do not occur simultaneously, allowing the p-MOSFET to bleed some of the charge. [21]

D had transitioned simultaneously, the maximum voltage would have been higher. In this case, it might have been necessary to move node C away from either node A or D, or otherwise resolve the large noise coupling.

## 21.7 TRANSISTOR SCALING TRENDS

To close this chapter, we will give a very brief overview of transistor scaling trends and the impact they have had on hot carrier reliability. The hot carrier reliability problem became a serious scaling concern for the first submicron technologies. Several techniques were introduced to deal with the problem. The first, drain engineering, employed the using of graded or lightly doped source/drain extensions, using LDD [24] or LATID [25] structures. In these structures, the graded or lightly doped region near the drain end served to reduce the lateral electric field that is at the root of the hot carrier problem; these solutions came at a small cost penalty in terms of increased MOSFET series resistance. The use of nitrided gate oxides [26] made the gate dielectric and the silicon–silicon dioxide interface much more immune to hot carrier injection, resulting in more reliable devices that could tolerate higher applied voltages. A better understanding also evolved regarding how hydrogen and water-related species diffused from inter-level dielectric layers (ILD) to the transistor, degrading hot carrier reliability. These problems were resolved using ILD layers with lower H or OH content, or by using blocking layers that prevented the diffusion of these species to the transistor. An understanding also evolved about how process-induced charging damage could degrade hot carrier reliability and steps were taken to minimize process charging damage. All of

these steps, along with power supply voltage scaling, enabled transistors to scale into the deep submicron arena. For sub-0.25  $\mu\text{m}$  transistors, the technique of drain engineering became too expensive as the increased external series resistance hurt these scaled devices significantly. Fortunately, improvements in the gate dielectric and backend layers along with voltage scaling were enough to allow abrupt source drain extensions.

The final question is whether the hot carrier effect will continue to occur as voltage is scaled downwards. If the applied drain-to-source voltage is less than 1 V, for example, how can an electron gain the energy required for impact ionization, which is greater than 1 eV? It appears that some small number of electrons still can, partly because they have some thermal energy, and also because electrons can scatter with each other [27]. As a result of electron-electron scattering, two electrons each with energy less than 1 eV could give rise to one electron with minimal energy and the other with energy greater than 1 eV. So the hot carrier effect will diminish as voltages are lowered but it will not disappear.

## REFERENCES

- [1] J. Frey, "Where Do Hot Electrons Come From?", *IEEE Circuits & Devices*, p. 31, Nov. 1991.
- [2] C. Hu, S. C. Tam, F. -C. Hsu, P. K. Ko, T. -Y. Chan, and K. W. Terrell, "Hot-Electron-Induced MOSFET Degradation—Model, Monitor, Improvement," *IEEE Trans. Electron Devices*, vol. 32, p. 375, 1985.
- [3] S. A. Abbas and R. C. Dockerty, "N-Channel IGFET Design Limitation due to Hot-Electron Trapping," *IEDM Tech Dig.*, p. 35, 1975.
- [4] T. H. Ning, C. M. Osborn, and H. N. Yu, "Effect of Electron Trapping in IGFET Characteristics," *J. Electronic Materials*, p. 93, 1977.
- [5] P. E. Cottrell, R. R. Troutman, and T. H. Ning, "Hot-Electron Emission in n-Channel IGFET's," *IEEE Trans. Electron Devices*, vol. 26, p. 520, 1979.
- [6] E. Takeda, S. Shimura, and T. Hagiwara, "Role of Hot-Hole Injection in Hot Carrier Effect and Small Degraded Channel Region in MOSFETs," *IEEE Electron Dev. Letters*, vol. 4, p. 329, 1983.
- [7] K. R. Hoffman, C. Werner, W. Weber, and G. Dorda, "Hot-Electron and Hole Emission Effects in Short n-Channel MOSFETs," *IEEE Trans. Electron Devices*, vol. 32, p. 691, 1985.
- [8] W. Weber, "Dynamic Stress Experiments for Understanding Hot Carrier Degradation Phenomena," *IEEE Trans. Electron Devices*, vol. 35, p. 1476, 1988.
- [9] T. Tsuchiya, "Trapped Electron and Generated Interface-Trap Effects in Hot-Electron-Induced MOSFET Degradation," *IEEE Trans. Electron Devices*, vol. 34, p. 2291, 1987.
- [10] B. S. Doyle, M. Bourcerie, C. Bergonzoni, R. Bennechi, A. Bravais, K. Mistry, and A. Boudou, "The Generation and Characterization of Electron and Hole Traps Created by Hot Hole Injection During Low Gate Voltage Stressing of n-MOS Transistors," *IEEE Trans. Electron Devices*, vol. 37, p. 1869, 1990.
- [11] B. S. Doyle, M. Bourcerie, J. -C. Marachetaux, and A. Boudou, "Interface State Creation and Charge Trapping in the Medium to High Gate Voltage Range ( $V_d/2 > V_g > V_d$ ) During Hot Carrier Stressing of n-MOS Transistors," *IEEE Trans. Electron Devices*, vol. 37, p. 744, 1990.
- [12] K. Mistry and B. Doyle, "How Do Hot Carriers Degrade N-Channel MOSFETs," *IEEE Circuits & Devices*, p. 29, Jan. 1995.
- [13] S. K. Lai, "Two Carrier Nature of Interface State Generation in Hole Trapping and Radiation Damage," *Appl. Phys. Letters*, vol. 38, p. 58, 1981.

- [14] I. -C. Chen, S. Holland, and C. Hu, "Electron Trap Generation by Recombination of Electrons and Holes in  $\text{SiO}_2$ ," *J. Appl. Physics*, vol. 61, p. 4544, 1987.
- [15] B. S. Doyle, K. R. Mistry and J. Faricelli, "Examination of the Time Power Law Dependencies in Hot Carrier Stressing of n-MOS Transistors," *IEEE Electron Dev. Letters*, vol. 18, p. 51, 1997.
- [16] R. Bellens, P. Heremans, G. Groeseneken, and H. Maes, "Hot carrier Effects in n-Channel MOSFETs Under Alternate Stress Conditions," *IEEE Electron Dev. Letters*, vol. 9, p. 232, 1988.
- [17] B. S. Doyle and K. R. Mistry, "The Characterization of Hot Carrier Damage in p-Channel MOSFETs," *IEEE Trans. Electron Devices*, vol. 40, p. 152, 1993.
- [18] B. S. Doyle and K. R. Mistry, "New Hot Carrier Failure Criterion for p-Channel Transistors Based on Transistor Leakage Currents," *Solid-State Electronics*, vol. 39, p. 1681, 1996.
- [19] K. R. Mistry and B. S. Doyle, "AC Versus DC Hot Carrier Degradation in n-Channel MOSFETs," *IEEE Trans. Electron Devices*, vol. 40, p. 1284, 1993.
- [20] M. Kuo, et al., "Simulation of MOSFET Lifetime under AC Hot-Electron Stress," *IEEE Trans. Electron Devices*, vol. 35, p. 1004, 1988.
- [21] K. R. Mistry, R. Hokinson, B. Gieseke, T. F. Fox, R. P. Preston, and B. S. Doyle, "Voltage Overshoots and n-MOSFET Hot Carrier Reliability in VLSI Circuits," *IRPS Proceedings*, p. 65, 1994.
- [22] K. R. Mistry, T. F. Fox, R. P. Preston, N. D. Arora, B. S. Doyle, and D. E. Nelsen, "Circuit Design Guidelines for n-Channel MOSFET Hot Carrier Robustness," *IEEE Trans. Electron Devices*, vol. 40, p. 1284, 1993.
- [23] D. Dobberpuhl, et al., "A 200 MHz 64-bit Dual Issue CMOS Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 27, p. 1555, 1992.
- [24] J. J. Sanchez, K. K. Hsueh, and T. A. DeMassa, "Drain Engineered Hot-Electron Resistant Device Structures: A Review," *IEEE Trans. Electron Devices*, vol. 38, p. 1125, 1989.
- [25] T. Hori, K. Korimoto, T. Yabu, and G. Fuse, "A New Submicron MOSFET with LATID (Large Tilt Angle Implanted Drain) Structure," *Symp. VLSI Tech. Dig.*, p. 15, 1988.
- [26] B. J. Gross, K. S. Krisch, and C. G. Sodini, "An Optimized 850C Low-Pressure Furnace Reoxidized Nitrided Oxide (ROXNOX) Process," *IEEE Trans. Electron Devices*, vol. 38, p. 2036, 1991.
- [27] J. E. Chung, M. -C. Jeng, J. E. Moon, P. -K. Ko, and C. Hu, "Low-Voltage Hot-Electron Currents and Degradation in Deep-Submicrometer MOSFETs," *IEEE Trans. Electron Devices*, vol. 37, p. 1651, 1990.

PART  
**VIII** | **CAD TOOLS AND TEST**

Yao-Tsung Yen  
*YX Technologies, Inc.*

## 22.1 INTRODUCTION

A state-of-the-art microprocessor design must strive to achieve the goal of the highest performance with acceptable power dissipation and chip area. Therefore, the chip implementation methodology is usually aggressive [1]–[6] and has two distinct characteristics:

- The basic building block is a transistor. That is, each transistor in the design can be individually sized and combined with other transistors to form various logic family.
- The physical design is partitioned, planned in detail, and committed long before the details of implementation have been worked out.

By way of contrast, contemporary, mainstream ASIC (application-specific IC) implementation methodology is based on the following three technologies:

- A library of predesigned and precharacterized cells
- A logic synthesis tool that maps to only a static complementary logic family
- A cell-based placement and interconnect routing tool

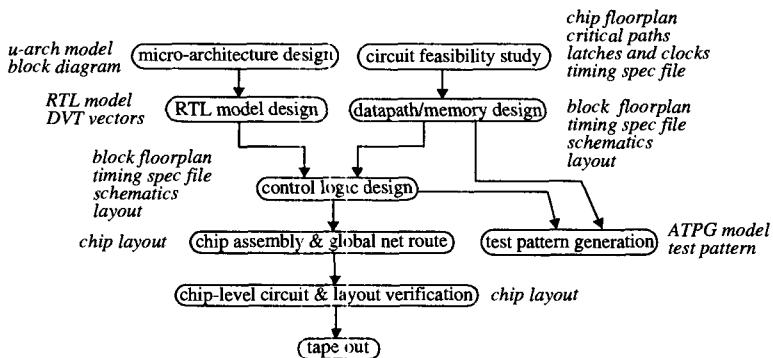
This methodology is not applicable to a microprocessor design because it compromises performance to an unacceptable degree.

Figure 22.1 shows a state-of-the-art microprocessor implementation flow. Each bubble is a design stage and the italic annotation beside each bubble is the design data that is created at that stage.

The design flow starts with two parallel stages: *micro-architecture design* and *circuit feasibility study*. The goals of the micro-architecture design stage are: (1) define the major features of micro-architecture, (2) create an architectural model for performance simulation, and (3) create block diagrams and documents to communicate with circuit designers, who are in the circuit feasibility study stage.

The goals of circuit feasibility study are: (1) create a chip floorplan based on the micro-architecture, (2) identify worst critical paths and perform feasibility design on these paths, (3) define the clocking strategy and design the associated latch library, and (4) define the power distribution strategy.

The micro-architecture design stage and circuit feasibility study stage interact with each other until it is clear that the micro-architecture is implementable and the design goals can be achieved. At this point, major datapaths, major memory arrays, and the major global signals connecting the datapaths and arrays are planned and defined.



**Figure 22.1** Flow diagram of a microprocessor design process.

The next step of the design flow is also composed of two parallel stages: *RTL (Register-Transfer Level) model design* and *datapath/memory design*. The goals of the RTL model design stage are to create an accurate register-transfer level simulation model that can withstand micro-architecture logic verification. The RTL model also serves as a detailed specification for the datapath, memory, and control schematics. The logic complexity of datapaths and memory arrays are simple and their structures are understood and agreed upon between micro-architects and circuit designers in the circuit feasibility study stage. Therefore, the design of schematics and layout of datapaths and memory arrays can proceed in parallel with the design of the RTL model.

The third step in the design flow is the design of control logic schematics and layout. At this point, since the datapaths and memory arrays are completed and committed, the design of control logic blocks must live within the remaining timing, area, and power budgets.

After the schematics and layout of all the constituent blocks (i.e., datapaths, memory arrays, and control logic) are completed, the chip layout is assembled and test patterns are created. In this last step before tape-out, all the previous planned power, ground, and global signals are realized in layout and verified.

In the following sections, we will describe the CAD tools required to support each stage of the design flow.

## 22.2 MICRO-ARCHITECTURE DESIGN AND CIRCUIT FEASIBILITY STUDY TOOLS

The goal of micro-architecture design is to create a performance model and a micro-architecture document that contains detailed block diagrams and the performance-critical critical paths. The performance model is usually highly parameterized so that many what-if tests can be tried on major architecture features. The design process is entirely manual. The performance model is usually written in a high-level programming language such as C or C++ so the only tools required are a good text editor and the language's compiler. For micro-architecture documentation, common personal computer word processors with graph-drawing features suffice.

The goal of the circuit feasibility study is to ensure that the micro-architecture is implementable. Based on a knowledge of the micro-architecture, circuit designers

partition the chip into multiple blocks and create a chip-level floorplan containing these blocks. Various schemes to distribute power are tested, and one is selected. More information may be found in Chapter 24, which discusses the design and analysis of power distribution networks. Clocking strategy (i.e., clock generation, clock distribution, and latch library) is also decided on and designed in the floorplan design stage. Clocking strategy is covered in Chapters 11 and 13.

With block definition, block placement, the power network, and the clocking scheme all planned, the inter- and intra-block critical paths are then designed and simulated. Since the floorplan is available, the routing of global nets among the blocks can be approximated and included in the critical path simulations. The results of floorplan and critical path designs are then provided to micro-architects so that the micro-architecture can be modified to accommodate the physical constraints. The critical path data passing through blocks (i.e., arrival time and departure time on block pins) are recorded in a global timing specification file which evolves eventually into a full chip-level timing specification as it includes all the pins of all the top-level blocks.

The essential tools for the circuit feasibility study stage are:

- schematic editor and netlister
- layout editor and parasitics extractor
- SPICE simulator [7]

Most commercial schematic editors, layout editors, parasitics extractors, and SPICE simulators have the capabilities needed for circuit feasibility study.

To facilitate the design of the chip floorplan, tools that perform what-if analysis of chip partitioning, block pin placement, and global net routes with a variety of different user-controllable algorithms would be very helpful. However, since the floorplan is created at the earliest stage of design, the tools must operate with partially completed design data. Most commercial tools are not designed for this purpose.

To ensure the integrity of the global timing specification, a tool to check the consistency of timing specification among top-level blocks and global net routes is required. Whenever there is a change in the global timing specification due to local changes in some blocks, the tool is run to check for problems with the new timing specification.

## 22.3 RTL MODEL DESIGN TOOLS

The goals of the RTL model design is to create a RTL-level simulation model that reflects the intent of the micro-architecture, can be exercised by the logic verification engineers to check for logical correctness, and contains enough details to serve as a specification of the micro-architecture to the circuit designers.

The first obvious tool is a logic simulator. Most commercial logic simulators are sufficient for the purpose of simulating the RTL model.

Since a state-of-the-art microprocessor is very complex, the RTL model must be created collectively. Therefore, certain RTL model coding style guidelines must be defined so that the RTL codes created by individuals in the design team are compatible enough to interoperate. The RTL model coding style guidelines need to cover at least the following aspects:

- clocking and latches style
- simulation performance

- synthesizable subset
- equivalency check against schematics
- hierarchy style
- signal naming convention

A tool is required to check these guidelines.

Since three design databases (RTL model, floorplan, and datapath/memory schematics) exist in this stage, a mechanism must be devised to keep the design databases in synch. The data in the RTL model that can affect the floorplan are partitions (i.e., blocks) and pins. To ensure that the floorplan is always in synch with the RTL model, a tool (RTL-to-fp) can be used to regularly export the most current partition and pin information to the floorplan. To maintain the linkage between the datapath/memory schematics and the RTL model, equivalency checking tools may be used. The consistency between datapath/memory schematics and their block representations in the floorplan also needs to be checked by a tool (schematics-to-fp). Figure 22.2 shows the tools required to support this synchronization mechanism.

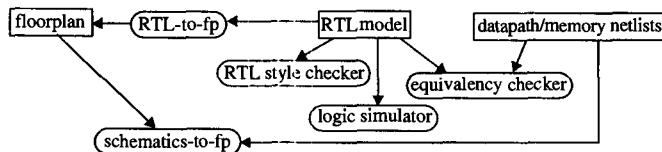


Figure 22.2 RTL model design tools.

Most commercial equivalency checkers are not intelligent enough to verify the equivalency between an RTL model and the corresponding schematic netlists. The major deficiencies of these commercial tools are: (1) the equivalency-checking algorithm is not smart enough to deal with a high level RTL model, and (2) the tools do not comprehend the variety of transistor-level circuits present in the schematics.

There are two ways to work around the tools' deficiencies: (1) writing the RTL model at a lower level, or (2) constructing a new RTL model [i.e., RTL model2 in Fig. 22.3(a)] for an “equivalency check” and checking this new RTL model against the original RTL model via logic simulation. In Fig. 22.3(a), model2 is first checked against datapath/memory netlists for equivalency. After the equivalency is established, model2 is then checked against the original RTL model using vector-based logic simulation.

Neither work-arounds is satisfactory. With the first work-around, the RTL model is written at a lower level and the simulation speed may be slowed down by an order of

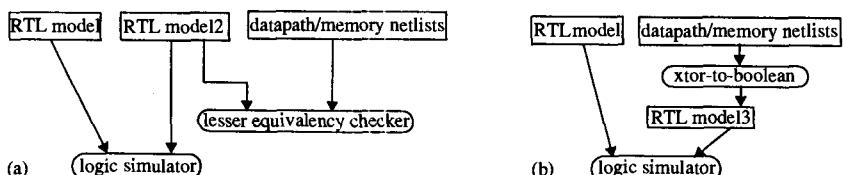


Figure 22.3 Alternative equivalency checking processes.

magnitude. With the second work-around, two versions of RTL model are required—this doubles the RTL model maintenance effort.

Another approach is to use a transistor-to-boolean-network tool to translate the schematic netlists into an RTL model [i.e., RTL model3 in Fig. 22.3(b)] and use a logic simulator to verify the equivalency between RTL model3 and the original RTL model. This approach is more efficient because model3 is generated automatically and no maintenance is required. However, it requires a tool that is intelligent enough to at least understand the logical function of each transistor in the design.

## 22.4 DATAPATH/MEMORY DESIGN TOOLS

The goal of the datapath/memory design stage is to create schematics and layout of datapath and memory blocks that meet the performance, area, and power targets. Since the basic building block is the transistor, tools for this stage of design must be operating at the transistor level.

Figure 22.4 shows the design flow of datapath/memory design and the minimal set of tools required to support the flow.

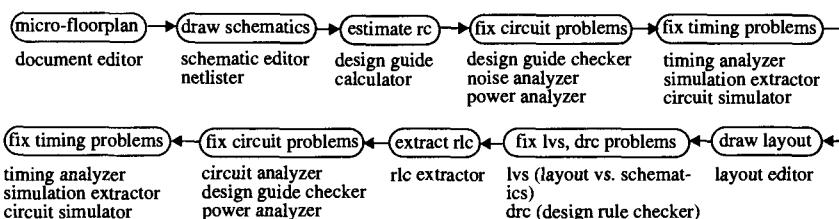


Figure 22.4 Datapath/memory design flow.

### 22.4.1 Datapath/Memory Schematic Design Tools

In the micro-floorplan design stage, designers can use commercial personal documentation tools to document the micro-floorplan. Alternatively, a designer can use a layout editor or a schematic editor to document the micro-floorplan. The micro-floorplan is based on incomplete, high-level design data. There is no special purpose commercial tool to aid the designers in this process.

After the micro-floorplan is constructed, the schematic drawing starts. Since known critical paths had been identified and designed at the earlier circuit feasibility study stage (see Sect. 22.2), the schematics are usually drawn using these critical paths as anchors. During the drawing of schematics, the timing specification of the top-level pins in the global timing specification file (see Sect. 22.2) is continuously updated to reflect the committed circuit implementation.

Based on the micro-floorplan, the capacitances and resistances (RC) of major signals are estimated and entered either as schematic objects or as properties of some schematic objects. The RC estimation can be done manually using a calculator and design guidelines. Alternatively, the calculator and design guidelines can be integrated with the schematic editor to improve the productivity of the schematic design process.

Most commercial schematic editors are sufficient for the schematic drawing tasks. However, it usually requires significant work to customize the netlister that comes with the schematic editor to accommodate the preferred style of the design team. Most of the customization work concerns the processing of hierarchy, parameter passing, nontransistor schematic objects, properties of schematic objects, and the estimation of transistor and interconnect parasitics.

#### **22.4.1.1 Circuit Guidelines and Noise Immunity**

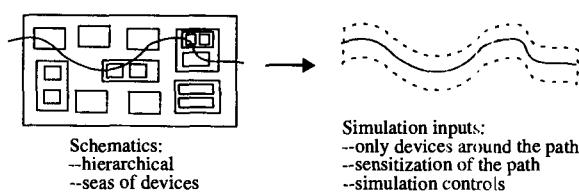
To ensure the robustness of a transistor-based design, circuit design guidelines are established up-front by the lead designers on the team. The rules include but are not limited to: allowed circuit families,  $\beta$ -ratios, allowed interconnections among circuit families, device sizes, schematic styles, clock usage, nodal rise/fall times, and latching rules. By requiring that these rules be observed in all schematics and by all members of the design team, the robustness and quality of the design can be achieved. This practice is not used in cell-based ASIC design flow. Instead, ASIC flows ensure the robustness of design through the practice of allowing only static complementary circuit family that has been characterized for worst-case usage.

In order to check that the circuit design guidelines have been observed, a tool is required [12]. This tool must understand the transistor-level circuits quite well. Once schematics are drawn, the netlists are created and passed to this tool which then checks for the adherence to the design guidelines and for robustness in the presence of noise. After the schematics are checked and proven robust, timing analysis is performed on the design. Timing analysis tools are covered in Chapter 23.

#### **22.4.1.2 Critical Path Simulation Input Extraction**

Most commercial timing analyzers do not have the accuracy of a circuit simulator. To remedy this deficiency, it is desirable to perform circuit simulation on each slow path reported by the timing analyzer to determine if the path is really critical. To accomplish this, a tool [11] is required that extracts a critical path from a schematic netlist and prepares the necessary inputs to sensitize the path. Figure 22.5 shows the function of the critical path simulation input extractor. Again, this critical path extraction tool must have a good knowledge of the operation of transistors in various circuit families. This tool can also be used to extract critical paths in memory blocks, which usually contain uncommon circuits (e.g., sense-amp, matching devices, pulse generators) and are beyond the analysis capability of commercial timing analyzers. Chapters 14 to 16 cover the design and analysis of memory circuits.

The last step in pre-layout circuit analysis is power estimation. There are no commercial static power estimation tools available today. Consequently, power



**Figure 22.5** Critical path simulation input extractor.

estimation is done by circuit simulation. Since the simulation must be done for all the devices in the schematics, fast but less accurate circuit simulators [8]–[10] can be used to shorten the simulation time.

### 22.4.2 Datapath/Memory Layout Design Tools

Layout drawing starts right after schematics have passed pre-layout circuit and timing analysis. Most commercial layout editors are sufficient for the layout drawing tasks. To increase the productivity of the layout design process, most repetitive, simple operations on layout database are automated using the API (application program interface) that comes with the layout editors.

The completed layout databases are then checked for any design rule violations and the equivalency against the schematics. Most commercial design rule checkers (a.k.a. DRC) and layout-schematic equivalency checkers (a.k.a. LVS) are sufficient for the required checks.

After a layout is LVS and DRC clean, post-layout circuit and timing analysis starts. The first step is to extract the parasitic resistances and capacitances of the transistors, and the interconnect capacitances, resistances, and inductances of the nodes. Most commercial layout parasitic extraction tools can extract transistor parasitics properly; however, they cannot handle the large amount of data associated with extracting the interconnect RLC parasitics for large blocks. Therefore, the sizes of datapath/memory blocks must be defined during the floorplanning stage to fit within the capacity of interconnect RLC extraction tools. If each constituent block can be extracted fully, then the chip-level extraction data can be constructed by stitching individual blocks together. This stitching process is described in Section 22.6.

## 22.5 CONTROL LOGIC DESIGN TOOLS

Control logic design starts after datapath and memory array designs are completed. Using chip floorplan and datapath/memory block timing specifications, the timing and area specifications of random logic blocks can be derived definitively.

Control logic is random in nature and, therefore, it is reasonable to use commercial logic synthesis and cell-based place&route tools to get the first-cut implementation, given the timing and area specification. However, since these commercial tools can only operate with a predefined library of static-complementary gates, the first-cut implementation usually does not meet the performance and area targets. In this case, the design team has two options: (1) refining the first-cut implementation to meet the design targets; or (2) start from scratch and use a transistor-based design flow. If option (2) is chosen, the tools described in Section 22.4 are used. On the other hand, if option (1) is chosen, the following tools greatly assist the design refinement process:

- ***Macro replacement tool:*** To facilitate the replacement of a group of standard cells in the synthesized netlist and layout by a custom-designed macro.
- ***RTL model re-partitioning tool:*** To facilitate the repartitioning of control logic RTL code and micro-floorplanning of control logic implementation. The repartitioning must be done for both the timing and the physical spaces.

- **Schematic drawing tool:** To facilitate the schematic redesign process. This tool accepts a cell-level netlist generated by synthesis tools and automatically creates a logic schematic drawing. Most commercial logic synthesis tools come with some schematic drawing tools. But, the readability of the resulting schematic is too low to be of much use in the redesign process. A significant in-house tool development effort is required to make the automatically drawn schematic readable.

## 22.6 CHIP ASSEMBLY AND GLOBAL NET ROUTE

With datapath, memory, and control logic designs completed, the chip can be assembled. At this stage of design, global nets are routed to completion using the reserved tracks. The tool used for this assembly process is the layout editor because, as explained below, the process is mostly manual.

Throughout the design process, the floorplan and its constituent blocks are constantly interacting with each other, as shown in Fig. 22.6. These changes have to be managed manually such that the floorplan and its constituent blocks are always consistent in the physical and timing spaces. There are no commercial tools that are intelligent and friendly enough to assist with the management of this complex interacting process. Consequently, the process is managed manually throughout the design process. Because the floorplan and reserved global net routing tracks are iterated constantly, by the time the design for all the constituent blocks is completed, the process of chip assembling is simple and can be done manually using a layout editor.

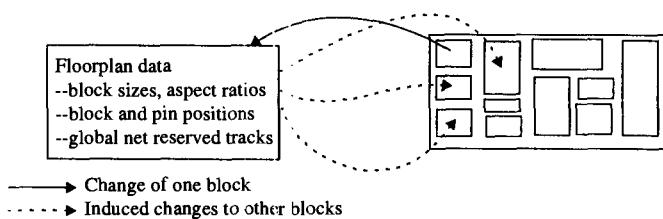


Figure 22.6 Interaction between floorplan and its constituent blocks.

## 22.7 CHIP-LEVEL LAYOUT, CIRCUIT, AND TIMING VERIFICATION

### 22.7.1 Layout Verification

After the chip is assembled, chip-level layout verification starts. Three tasks are performed at this design stage: LVS, DRC, and parasitics extraction. Most commercial tools are capable of performing hierarchical LVS and DRC operations. However, a significant amount of effort is required to get full-chip parasitics extraction to work.

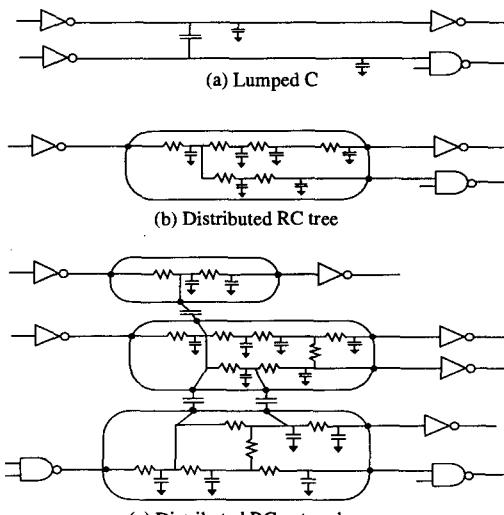
Most commercial tools cannot handle the vast quantity of data in a state-of-the-art microprocessor. So many glue tools are required to integrate the commercial tools to

form a complete post-layout analysis flow. These glue tools perform the following tasks:

- Partition chip layout database into multiple blocks such that the data size of each block is within the capacity limit of commercial parasitics extraction tools
- Tie up loss ends of each partitioned block by adding artifacts (e.g., dummy pins and devices) and labels to the layout database
- Stitch together the parasitics data from individual blocks to form a chip-level parasitics database
- Back-annotate the chip-level parasitic data into the schematic netlists

Another dimension of complexity comes from the parasitic models. There are three types of parasitic models: (1) Lumped C, (2) distributed RC tree, and (3) distributed RC network, as shown in Fig. 22.7. Which model to use is largely determined by the analysis tools and the scope (i.e., chip, block, or sub-block) of the analysis. The following three examples show how to select the proper extraction model for different purpose:

1. None of the commercial timing analysis tools can understand distributed RC network. Therefore, only the lumped C or distributed RC tree model can be used for timing analysis.
2. To accurately analyze major signals in an array, a distributed RC network model should be used. However, since distributed RC networks are very compute intensive to extract, only a handful of critical nets should be extracted.
3. For static circuit analysis and design guideline checking, the lumped C model is adequate.



**Figure 22.7** Three extracted parasitics models.

(c) Distributed RC network

## 22.7.2 Circuit and Timing Analysis

The chip-level timing analysis problem is partitioned into an interblock global timing specification consistency problem and a block-level timing analysis problem. These two subproblems are solved incrementally while the design of the datapath,

memory, and control blocks are progressing. By the time the chip is ready for assembly, timing problems local to the blocks have already been fixed and interblock timing specification is consistent among the blocks and the estimated global net routes. Any remaining local circuit problems in the blocks are fixed at this time. After the chip is assembled, global net routes are committed and their RC parasitics are extracted. The goal of global chip-level circuit and timing analysis is to catch circuit and timing problems that fell through the cracks in the local block analysis process.

It is highly desirable that chip-level circuit and timing analysis tools are exactly the same as local block-level analysis tools, in both pre-layout and post-layout modes. To achieve this, the tool flow needs to be set up such that the only difference between the pre-layout and post-layout flows are the RC values used by the analysis tools. Many glue tools are needed in order to achieve this. The major problem to be solved is: Extracted netlists are “flat” at the transistor level, while the pre-layout netlists are highly hierarchical. If the analysis tools are working at the “flat” transistor level, the required run times will be long and the process of back-annotation to schematic for cross-probing is more difficult. On the other hand, if the analysis tools can operate hierarchically, then the process of annotating the “flat” extracted parasitics to the hierarchical data structure utilized in the tools is difficult. At present, the solutions to these difficult CAD problems has just started appearing commercially (e.g., XE tool from YX Technology, Inc.) but still not generally available. Therefore, a significant in-house development effort is required.

## 22.8 TEST PATTERNS GENERATION

Test patterns can be generated randomly, manually, or automatically. To achieve the highest coverage, automatic test pattern generation should be used. Most commercial ATPG (automatic test pattern generation) tools are adequate to generate high-coverage test patterns. However, these tools require a gate-level netlist. In a transistor-based microprocessor design, as the design approaches over 100 million transistors, it is impossible to create a gate-level netlist manually from a transistor netlist. It is highly desirable to have a tool to translate a transistor-level netlist into an equivalent gate-level netlist. This tool needs to understand circuit families used in the design and perform the following tasks:

- Preserve the hierarchy in the original schematic netlists
- Abstract away the transistors that are only for circuit purpose (e.g., precharge and discharge transistors, feedback devices to hold the states)
- Map the remaining logic transistors into ATPG tool primitives

Most commercial ATPG tools do not perform these tasks. Again, a significant in-house development effort is required.

## 22.9 CONCLUSION

We have reviewed a state-of-the-art microprocessor design methodology and outlined the tools required to support the methodology. There are two characteristics that distinguish a microprocessor design from the mainstream, standard-cell based ASIC

designs: (1) transistor-based, and (2) up-front planning and early commitment in the physical implementation. To support these two paradigms, CAD tools need to understand the function of transistors in the design and be able to work with partially completed data. Most commercial tools do not have these capabilities. As a result, a microprocessor design team must invest heavily in in-house tools to create the required tool flows. Since the microprocessor design community is much smaller than the ASIC design community, commercial tools vendors are reluctant to invest in the transistor-level tools for microprocessor designs. Therefore, significant CAD resources are needed to support state-of-the-art microprocessor projects.

## REFERENCES

- [1] W. Bowhill, et al., "A 300 MHz 64b Quad-Issue CMOS RISC Microprocessor," *ISSCC Digest of Technical Papers*, pp. 182–182, Feb. 1995.
- [2] P. Gronowski, et al., "A 433 MHz Quad-Issue RISC Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 31, no. 11, pp. 1687–1796, Nov. 1996.
- [3] J. Montanaro, et al., "A 160 MHz, 32-b, 0.5 W CMOS RISC Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 31, no. 11, Nov. 1996.
- [4] B. Gieseke, et al., "A 600 MHz Superscalar RISC Microprocessor with Out-of-order Execution," *ISSCC Digest of Technical Papers*, pp. 176–177, Feb. 1997.
- [5] J. Warnock, et al., "High Performance CMOS Circuit Techniques for the G-4 S/390 Microprocessor," *Proceedings of ICCD*, pp. 247–252, 1997.
- [6] W. Grundmann, et al., "Designing High Performance CMOS Microprocessor Using Full Custom Techniques," *Proceedings of DAC*, pp. 722–727, 1997.
- [7] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," *Report ERL-M520*, Electronics Research Laboratory, UC Berkeley, May 1975.
- [8] C. Visweswarish and R. Rohrer, "Piecewise Approximate Circuit Simulation," *IEEE Transaction on CAD*, vol. 10, no. 7, pp. 861–870, July 1991.
- [9] E. Ngoya, et al., "Newton–Raphson Iteration Speed-up Algorithm for the Solution of Nonlinear Circuit Equations in General-Purpose CAD Programs," *IEEE Transaction on CAD*, vol. 16, no. 6, pp. 638–644, June 1997.
- [10] A. Devgan, "Transient Simulation of Integrated Circuits in the Charge-Voltage Plan," *IEEE Transaction on CAD*, vol. 15, no. 11, pp. 1379–1390, Nov. 1996.
- [11] M. Desai and Y. Yen, "A Systematic Technique for Verifying Critical Path Delays in a 300 MHz Alpha CPU Design Using Circuit Simulation," *Proceedings of DAC*, pp. 125–130, 1996.
- [12] W. Grundmann and Y. Yen, "XREF/Coupling: Capacitive Coupling Error Checker," *Proceedings of ICCAD*, pp. 244–247, 1990.

Victor Peng  
*MIPS Technologies, Inc.*

## 23.1 INTRODUCTION

The challenge of high-speed microprocessor timing verification goes beyond CAD technology and usage. Timing verification and design methodology must be considered together starting with product definition. There are many valid methods that can be used to ensure a design meets its timing targets. The optimal method depends on a diverse set of parameters such as the circuit techniques used, cost goals, power constraints, speed bin strategy, process technology, porting considerations, CAD technology, and a dose of the design “religion” practiced by the development team.

This chapter focuses on the practical aspects of timing verification for custom high-speed circuits prior to the availability of chip prototypes. We will discuss a holistic approach to timing verification. CAD tool requirements and usage are discussed; however, the details of how timing verification tools work are not.

This chapter covers:

- The goals and basics of timing verification
- Key factors that must be accounted for in timing verification
- Timing verification of typical nonmemory custom circuits
- Timing verification of typical embedded memory circuits
- The design flow and full-chip verification
- Future challenges

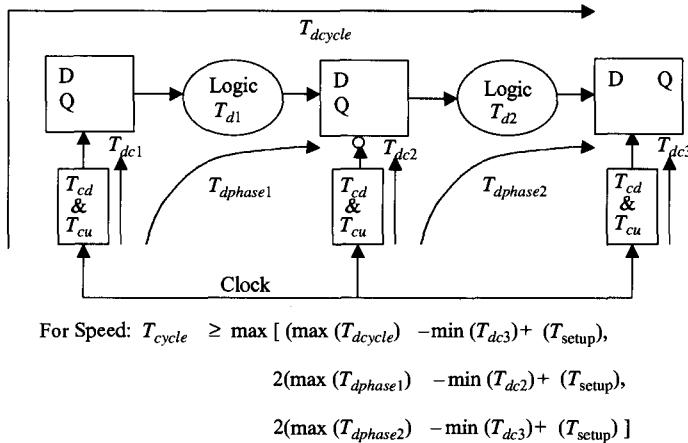
## 23.2 TIMING VERIFICATION GOALS AND ANALYSIS

The goal of timing verification is to ensure a chip will operate at a frequency or range of frequencies with a desired yield under the specified range of operating conditions. Conceptually it's useful to consider separately issues concerning full-speed operation and issues concerning correct functional behavior at any frequency.

### 23.2.1 Speed (Set-Up Time) Analysis

Verifying that a chip design will run at full speed (i.e., meets the maximum frequency target) involves identifying the worst-case maximum signal delay path through logic between two synchronization points and determining its delay accurately. The minimum cycle time is set by the worst-case sum of the maximum signal path delay (including wire delays) between two synchronization points, all storage element delays

in the path, set-up time required at the ending synchronization point, and the maximum uncertainty of the synchronizing clock edges. The worst-case sum can be based on a path that must complete in a clock phase, a cycle, or multiple cycles in the case of a latch-based design. The typical set-up and hold-time constraints on the cycle time for a two-phase latch-based design are depicted in Fig. 23.1.



**Figure 23.1** Speed (set-up) and functionality (hold) timing requirements for latch-based designs.

A synchronization point is a point in a circuit at which signal propagation is gated by a clock. Level-sensitive latches, edge-triggered flip-flops, dynamic logic gates, and clock qualification gates all contain synchronization points. Clock uncertainty is any deviation from an ideal time reference. Clock uncertainty is caused by clock jitter and clock skew. Clock jitter is random variation in the cycle time of the clock waveform primarily due to noise effects on the clock generation and distribution circuits. Clock skew is the difference in clock arrival times at different points of the clock network due to clock network delays.

A consistent set of process and chip operating conditions assumptions must be made during design and verification of chip timing goals. How conservative these assumptions are depends on performance and cost goals. A conservative speed analysis which isn't based on absolute worst-case assumptions will usually still result in a functioning product but with lower than planned yield at the target frequency.

### 23.2.2 Functional (Hold Time) Analysis

Signal propagation delays can impose limits on maximum and minimum chip operating frequency, or even worse, result in a chip that doesn't operate reliably at any frequency. This occurs when a chip contains circuits that have race hazards. A circuit contains a race if its correct operation depends on signal propagation delays that are independent of the chip clock frequency. The signal path that causes incorrect behavior in a race will be referred to as the aggressor path. The path that is racing against the aggressor path will be referred to as the victim path. Race analysis involves identifying race hazards and determining if a failure exists by checking the worst-case minimum delay of the aggressor path against the worst-case maximum delay of the victim path.

A latch data hold-time failure is a simple example of a race where new data propagates into a storage element one cycle in advance of when it should. In this case, the data signal path is the aggressor path and the victim path is the de-assertion of the storage element clock pin. A race failure exists if the sum of the worst-case minimum aggressor path delay and the minimum storage element hold time is less than the maximum storage element clock de-assertion delay measured from a common clock edge.

While basic hold-time failures are the most common race problems, the use of self-timed circuits introduces additional and often complex race paths. In the pursuit of higher and higher clock frequencies, self-timed circuits are used in many microprocessors despite the risk of timing failures they introduce. The most typical application of self-timed circuits are in embedded memory structures and sometimes speed critical data path blocks.

The assumptions about process and chip operating conditions must be very conservatively chosen for functional timing analyses. The consequences of optimistic assumptions can result in a total yield bust and a very difficult debug problem.

### 23.2.3 The Timing Analysis Process

Only static timing analysis will be considered since it is the standard method used for processors and ASICs today. Dynamic timing analysis, which involves sensitizing timing paths through a network by applying a set of input vectors, is an older pattern dependent verification methodology.

The basic steps in the static timing analysis of a netlist are as follows:

1. Determine primary output pin loading and departure time requirements.
2. Determine primary input pin loading and arrival time.
3. Identify all clocks.
4. All circuits in the block must be recognized to understand how signals propagate and identify synchronization points.
5. Generate timing constraints at all circuit pins.
6. Propagate minimum/maximum clock timing through the block from primary input to all synchronization points as appropriate for a timing or functional analysis.
7. Propagate minimum/maximum rising and falling signal timing as appropriate from either primary inputs to primary outputs or outputs to inputs for a timing or functional analysis.
8. While propagating timing, estimate the delay of each circuit element (transistor, gate, or a black-boxed subcircuit) in the block with the appropriate min/max load models.
9. Check whether timing constraints through valid paths are met, accounting for clock uncertainty, and note all violations.

This process seems simple but in reality is very complex for both humans and programs when it is applied to multimillion transistor processors designed using exotic circuit techniques and fabricated in multithreshold, six to eight metal layer CMOS technologies.

### 23.2.4 Max Path Modeling

The following factors should be accounted for in determining the maximum delay through a signal path:

1. A single path from a circuit output to power or ground through transistor source/drain, referred to as a channel connected region (CCR), is assumed to switch on to model circuit delay. The single slowest CCR for each gate circuit is used.
2. The worst-case process and operating conditions for speed are assumed.
3. All wire RC delays in the path to be maximized for worst-case speed, and any potential crosstalk capacitance is assumed to switch in the opposite direction as the signal of interest.
4. All off-path wire loads that have significant RC delay are assumed to minimum  $R$  with maximum  $C$  lumped at the end of the wire. This maximizes the effective load on the path of interest by coupling large capacitive loads through smaller resistors.
5. Assume maximum device source/drain capacitance connected to each CCR in the path.
6. Off-path gate loads are maximized with other input state set to maximize gate capacitance and Miller effects.
7. Precharged, or predischarged, nodes in the CCR are assumed to be charged to the maximum possible power/ground levels.

### 23.2.5 Min Path Modeling

The following factors should be accounted for in calculating the minimum delay through a signal path:

1. All parallel CCRs from a circuit output to power or ground are assumed to be simultaneously switched on to model circuit delay.
2. The worst-case process and operating conditions for races are assumed. The specific process corner and operating conditions that are worst-case for races are a function of the process technology and the circuit design styles used in the design.
3. All wire RC delays in the path to be minimized are assumed to have minimum  $R$ , minimum total  $C$ , and any potential crosstalk capacitance transitioning in the same direction as the wire of interest.
4. All off-path wire loads that have significant RC delay are assumed to maximum  $R$  with minimum  $C$  lumped at the end of the maximum  $R$ . This minimizes the effective load on the path of interest by decoupling capacitive loads through larger resistors.
5. Assume minimum device source drain capacitance connected to each CCR in the path.
6. Off-path gate loads are minimized with other input state set to minimize gate capacitance and eliminate/reduce any Miller effects.
7. Precharged (predischarged) nodes in the CCR are assumed to have the minimum (maximum) possible precharged (predischarged) voltage to reduce the evaluate time of the CCR. The voltage offset may be due to IR drops and/or coupling to the dynamic node after the precharge (predischarge) device is off.

Some of the above factors may not apply to a particular signal path. For cases that rely on a logical assertion, logical verification to validate the assertion is required.

## 23.3 KEY FACTORS IN HIGH-SPEED DESIGN AND TIMING VERIFICATION

### 23.3.1 Product Goals

Product goals are the primary driver for all development considerations. While high-frequency operation is by definition a key goal for high-performance microprocessors, some performance may be traded off to satisfy other goals such as cost, power dissipation, design reuse, or development schedule. The capabilities of CAD tools and the underlying assumptions built into the tools must also be considered. The exact weighting of these conflicting goals influences which design and verification strategies are best suited to a particular product developments.

A key design question, which must be answered during the definition stage of a product, is under what process and operating conditions must the product meet its frequency target(s)? For the highest-performance processors targeted for less cost-sensitive applications such as workstations and servers, a common strategy is to ensure the design meets its target frequency assuming typical process (transistor and interconnect models, no on-chip variations) and worst-case operating conditions (temperature, voltage, and data). However, functional failure rules and analyses assume worst-case process and operating conditions.

Embedded processors and processor cores intended for integration on an ASSP or ASIC are typically more cost sensitive than standard microprocessors. Consequently it may be necessary to address speed and functional issues assuming worst-case process and operating conditions. Whatever the overall design strategy, it is imperative that the basic process and operating condition assumptions are consistently applied to all aspects of the design and verification methods and tools.

### 23.3.2 Clocking and Storage Element Strategy

#### 23.3.2.1 Clocking Systems

The clocking system of a chip greatly impacts its design and timing verification. Non-overlapped clock systems enable latch-based designs with minimal hold-time failure risk, and reduces clock edge rate requirements. Clock buffering and gating can be utilized with ample design margin for hold time. Non-overlapping clocks also provide several clock edges, which are useful for implementing dynamic logic and embedded memories. However, generating guaranteed non-overlapped clocks is more difficult at higher frequencies so fewer high-speed processors use these clock systems.

A commonly used clocking system employed in high-performance processors [1], [3], [4] is a nominally zero-overlapped two-phase clock system. In this scheme, a globally distributed clock that has a nominal 50/50 duty cycle defines the cycle time. Phase 1 of the cycle is defined to be when the global clock is high, and phase 2 is when the clock is low. The global clock is locally buffered, and possibly qualified (gated) and/or pulse-shaped, prior to driving the end loads. This two-phase clock system will be the assumed clocking methodology for the remainder of this chapter. Nevertheless, the timing verification concepts discussed are applicable to other clock systems too.

### 23.3.2.2 Clock Uncertainty Specifications

Clock uncertainty primarily results from clock jitter, clock duty cycle variation, and clock skew. Clock jitter reduces the effective cycle time from its nominal value. Therefore clock jitter must be accounted for in speed (set-up) time analyses. It is typically accounted for by reducing the cycle time by the worst-case jitter specification. The jitter specification should be consistent with the worst-case process and operating conditions assumptions for speed analyses. Clock jitter does not impact race through (hold time) problems since variations in clock waveforms and frequency doesn't impact clock arrival time variations.

Clock duty cycle variation has a similar impact on the design as clock jitter. Duty cycle variation may impact circuit speed but not circuit functionality. In a two-phase clock system the duration of each phase is determined by the clock duty cycle; therefore, variations from the nominal duty cycle reduce the effective phase and cycle times.

Clock skew can negatively impact the speed of logic and exacerbate race-through. Since clock skew must be accounted for in speed and race analyses, different skew specifications may be needed for set-up and hold-time checks. Each skew specification must be derived under the appropriate worst-case assumptions for speed and race analyses. Alternatively, one specification for clock skew can be derived for both set-up and hold-time checks under the appropriate conditions, and additional constraints can be applied for each type of check.

Clock skew is typically greatest between physically distant points on the chip since the clock interconnect RC delay between such points can be significant. The worst-case global clock skew specification must account for variations across the chip such as:

- Interconnect resistance and capacitance (including coupling capacitance)
- Data-dependent device loads
- Power and ground noise
- Thermal differences
- Intra-die process variations

Note that intra-die process variation can exist for physically local and apparently identically drawn circuit layouts. The amount of variation is process dependent but is generally significant enough that it must be considered for functional analyses.

When clock propagation relative to two synchronization points is in the opposite direction as data propagation between the synchronization points, set-up time constraints are more difficult to meet. When clock propagation is in the same direction as data propagation in a signal path between synchronization points, hold-time constraints are more difficult to meet. Both these conditions can exist simultaneously for signal paths through the same logic block. This is sometimes a result of poor choice of connections to the global clock lines, and/or poor clock routing, which results in significant clock RC delays to exist between physically local synchronization points.

### 23.3.2.3 Multiple Clock Domains

Multiple clock domains on a single chip can arise from a number of situations including a hierarchical partitioning of a single synchronous clock domain for skew

reduction; the existence of two or more synchronous clock domains which operate at multiples of a common frequency; or two or more totally asynchronous clock domains. Many processors have at least two synchronous clock domains: the processor core clock and the processor bus interface unit (BIU) clock. With increasing levels of integration on a chip, it is not uncommon to include completely asynchronous clock domains of IO logic.

Clock skew as a percentage of processor cycle time has increased as minimum features have shrunk, die sizes have grown, and frequency targets increased. Widening clock interconnect lines and distributing global clocks with low-resistance grid topologies to reduce skew has the drawback of increasing the power dissipated by the clocking system. One solution to the growing clock skew and power dissipation problems is to implement a hierarchical clocking scheme where a global clock is used to generate local clocks which are then distributed across smaller chip partitions.

The advantage of a hierarchical clock scheme is that the clock skew within the local clock domains can be controlled to a tighter skew specification since clock interconnect delays and the variation in circuit operating conditions are reduced. However, the skew between clock domains is greater than if a single clock network were designed for the entire chip. This can be a good trade-off if the chip regions are defined such that the number of signal paths that cross clock domains are reasonable and the larger skews have been accounted for up-front. This benefit comes at the expense of extra analysis of paths that cross multiple clock domains since intra- and interdomain clock skews are different.

Multiple synchronous on-chip clock domains that operate at different frequencies introduce larger skews at domain boundaries. There is the added complication that the frequencies of the time domains differ by some multiple. In the common case of the core clock and the BIU clock frequencies, their ratio is programmable and may include odd and half-fractional multiples.

On-chip IO bus interface blocks are often run by a slow clock that is asynchronous to the other on-chip clocks. The communication between asynchronous blocks is typically accomplished with the use of handshake signals that cross the clock boundaries through synchronizing circuits. Functional operation depends on the correct operation of the handshake protocol and the synchronizing circuits.

Today's static timing analysis (STA) tools are not capable of directly analyzing the timing of a chip containing multiple clock domains. A typical methodology for verifying a chip with multiple synchronous clock domains is to run STA on each clock domain. Then interdomain timing paths are verified by extracting the portions of each clock domain which affect these paths and running an STA on the extracted networks. The specification of the domain interfaces, the extraction of the interdomain networks, and the worst-case assumptions of clock skew, process variation, etc., must be correct and consistent for the assertion that the timing is "correct by superposition" to be true. Each additional manual step that cannot be exhaustively checked by programs increases the risk of error.

#### 23.3.2.4 Clock Gating and Pulse Shaping

Another consequence of increasing processor frequencies is that maintaining constant power dissipation, which is a system level requirement, is even more challenging. Gating the toggling of clocks has become a common method to reduce average power

dissipation on microprocessors and embedded processors. Conditional toggling of clocks saves power by preventing both the logic and the clock node from toggling when logic evaluation is not needed. Clock pulse shaping is also commonly employed to reduce short-circuit currents in dynamic circuits.

The introduction of gated clocks increases the potential for hold-time problems since clock skew will increase even between synchronization points that are physically local. Consider the clock gating and buffer logic shown in Fig. 23.2. The rising and falling transition delays must be matched between the gating and nongating logic to minimize skew. NAND and NOR gate delays do not match well with inverters across process and operation condition variations. NANDs and NORs also present the global clock with data-dependent gate loads. Furthermore, the layout of the clock gating logic and the routing of the clock must be carefully planned. These issues make clock gating a bad design practice if a highly automated implementation methodology is used to design the chip.

Clock gating increases the number, and potentially the type, of timing constraints that must be identified and checked by STA tools. It also increases the complexity of modeling worst-case on and off-path electrical loads. STA tools must recognize clock waveform shaping circuits and treat them differently than ordinary gates since the correct delay propagation for these two cases are different.

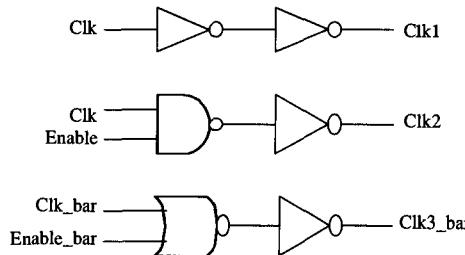


Figure 23.2 Clock buffer example.

### 23.3.2.5 Flip-Flop-Based Designs

A design that uses only edge-triggered storage elements can be more straightforward to design and verify. The cycle time for a flop-based design is the sum of the clock to output delay of a flop, plus the total delay of the worst-case circuit path to the next flop, plus the receiving flop set-up time, plus clock uncertainty. The price of ease of analysis afforded by flop-based designs is that clock uncertainty cannot be hidden, which reduces the portion of the cycle time available for logic evaluation. Some high-performance processors [1], [4], [5] use differential amplifier edge-triggered flops to offset the clock uncertainty penalty by reducing the total storage delay in a cycle relative to a latch-based design. The storage delay penalty per cycle for such designs is the delay of one fast flop instead of two level-sensitive latches.

### 23.3.2.6 Latch-Based Designs

The main advantage that latch-based designs can have over edge-triggered designs is that the cycle time overhead of clock uncertainty can be eliminated if the circuit delays between latches are balanced. In addition, the logic evaluation time available between two latches can roughly be up to the number of clock phases between the latches plus

one phase since the latch input is sampled for an entire phase and not at an edge. This is not the case for flop-based designs. Taking advantage of this design flexibility is called *timing borrowing* or *cycle stealing*. Time borrowing can be used in a logic path that loops back on itself (e.g., a multicycle divider/multiplier block) to accommodate a loop delay, including latch input to output delays, that is longer than the nominal cycle time.

As performance targets get faster and interconnect delays increase relative to transistor delays with each process generation (copper interconnects notwithstanding), the penalty that clock uncertainty imposes on cycle time becomes more costly. Consequently a number of high-speed microprocessor designs [2], [3], [6] have been latched based. However, the time-borrowing aspect of latch-based design increases the complexity of the timing verification problem. STA tools and chip designers have difficulty analyzing multicycle timing paths.

### 23.3.2.7 Storage Element Timing Characterization

The set-up time requirement of storage elements for circuits such as shown in Fig. 23.3 is usually defined as the latest data arrival time relative to the clock de-asserting edge such that the storage node  $Z$  is charged to within 90–95% of the correct final voltage. For combinatorial latches such as the RS latch in Fig. 23.3, the set-up time is defined to be the latest data arrival time relative to the clock de-asserting edge such that the valid data value has propagated to the feedback node.

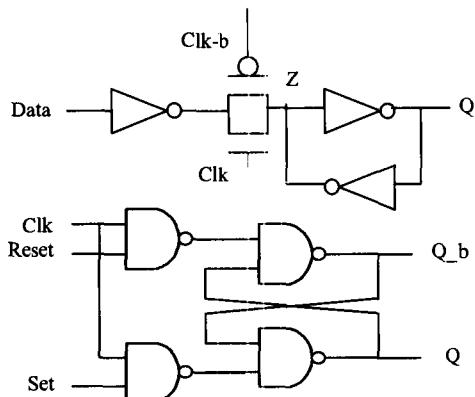


Figure 23.3 D latch and RS latch.

Set-up time must also account for the impact that a late-arriving input can have on the specified input to output delay ( $t_{du}$ ), and the clock to output delay ( $t_{cq}$ ). Once the set-up time for correctly capturing the data is determined, the  $t_{dq}$  and  $t_{cq}$  delays must be re-verified for data arriving with the minimum set-up time to ensure that all state element specifications are consistent.

Hold time is defined as the earliest time an input signal can change after the clock de-asserting edge such that the storage or feedback node doesn't change more than ~5% from the correct voltage level. An additional constraint is that the state element output(s) should not glitch since glitches can cause functional failures in downstream logic.

The circuit simulations for determining the state element timing constraints must appropriately worse case the input data, feedback, and clock paths in the state element.

**TABLE 23.1** Circuit Simulation Worst-Case Conditions for Storage Element Timing Characterization

Storage element timing parameter	Data and feedback path delay	Clock path delay	Worst-case conditions
Min input to output delay, $(T_{dq})_{\min}$	Min	DC	Functional
Max input to output delay, $(T_{dq})_{\max}$	Max	DC	Speed
Min clock to output delay, $(T_{cq})_{\min}$	Min	Min	Functional
Max clock to output delay, $(T_{cq})_{\max}$	Max	Max	Speed
Data set-up time, $(T_{su})$	Max	Min	Speed
Data hold time, $(T_{hd})$	Min	Max	Functional

Table 23.1 shows which paths are maximized or minimized under the appropriate set of worst-case assumptions for storage elements.

### 23.3.3 Circuit Structures and Design Guidelines

The circuit design styles and guidelines used to implement the processor heavily influence the timing verification strategy and CAD tool requirements. Few high-speed processors are implemented using only complementary static logic. Most fast microprocessors are implemented using some or all of the following: dynamic logic, pass-gate logic, differential cascode logic, low-voltage swing circuits, and self-timed circuits. Each of these logic families has a unique set of constraints and characteristics. Dynamic logic, pass-gate logic, and self-timed logic will be discussed further in later sections. Low-voltage swing circuits and other novel circuit techniques are typically treated as special cases and are beyond the scope of this chapter.

The greater the diversity of circuit techniques used in a design, the greater the verification challenge. Most of today's STA tools can only recognize a few circuit design styles beyond complementary logic. This introduces the time-intensive task of "black boxing" unrecognized circuits. In addition, the timing models used by STA tools are much less accurate for the custom circuits that they do recognize ( $> 10\%$ ) than they are for complementary logic ( $< 5\%$ ).

General circuit and electrical guidelines must be factored into the timing verification assumptions made by designers and CAD tools. Custom processor design projects usually define at least the following set of general design guidelines which must be factored into timing verification:

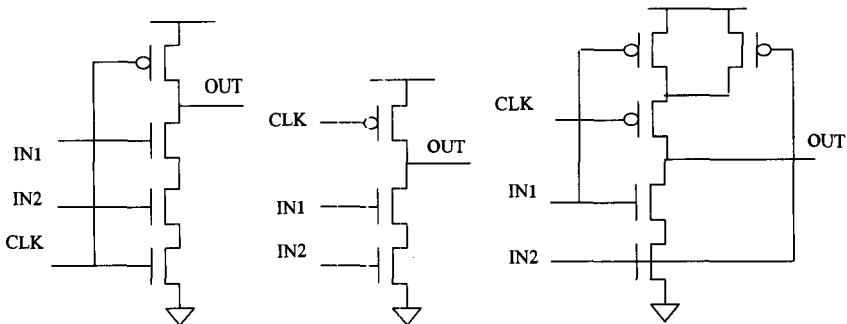
- Min/max signal and clock slew rate.
- Min/max clock skew and jitter.
- Clock fan-out restrictions.
- Max signal RC delay.
- Signal and clock capacitive coupling limits.
- Power and ground maximum IR drop limits.
- Minimum precharge levels on dynamic nodes.

These guidelines affect timing modeling, load modeling, skew specifications, and set-up and hold-time constraints.

## 23.4 TIMING VERIFICATION OF NONMEMORY CUSTOM BLOCKS

### 23.4.1 Dynamic Logic

There are numerous families of dynamic logic. Figure 23.4 illustrates a few examples of commonly used dynamic logic families: domino logic, nonfooted domino logic, and data precharged dynamic logic. The diversity of dynamic circuit structures and their unique set of timing constraints make their verification a challenge. Most commercially available STA tools have difficulty recognizing a broad range of circuits and correctly generating the appropriate timing constraints. Many high-performance microprocessor design groups have developed proprietary STA tools to support processor implementations incorporating dynamic circuits [7], [8], [9].



**Figure 23.4** Examples of domino, nonfooted domino, and data precharged logic, respectively.

The following is an example set of timing constraints for domino logic signal inputs:

1. High-to-low (HL) input transitions have positive set-up relative to the clock asserting edge to avoid unintended discharge of the dynamic node.
2. HL input transitions meet a hold time relative to the clock de-asserting edge to avoid output glitches and potential races.
3. Low-to-high (LH) input transitions meet the set-up time relative to the clock de-asserting edge needed to fully discharge the dynamic node.
4. LH input transitions have nonnegative hold time relative to the clock de-asserting edge to avoid a high-impedance low voltage on the dynamic node.
5. The clock low pulse width is sufficient to fully precharge the dynamic node.

Constraint 4 can be dropped if its associated potential failure mechanisms are addressed by design or other verification means. For simplicity, constraints 2 and 4 can be combined requiring that a hold-time spec for any input transition must be satisfied.

The following is an example set of timing constraints for nonfooted domino logic:

1. HL input transitions occur within a time window around the precharge clock assertion to avoid high-impedance low voltage or an excessive short-circuit current between the precharge and discharge paths.

2. LH input transitions meet the set-up time relative to the precharge clock asserting edge needed to fully discharge the dynamic node.
3. LH input transitions occur no earlier than some delta before the precharge clock de-assertion to avoid excessive short-circuit current between precharge and discharge paths.
4. The precharge clock low pulse width is sufficient to fully precharge the dynamic node.

Data precharged dynamic logic is designed to remove the need to have the input delays track the precharge clock delay to avoid excessive short-circuit current. While this is an attractive feature, the timing constraints are not easily verified. The timing constraints for data precharged logic are:

1. The time window during which all inputs are simultaneously high must be sufficiently long to fully discharge the dynamic node.
2. The overlap of the input low pulse width and the clock de-assertion is sufficiently wide to fully precharge the dynamic node.

The dynamic circuit examples shown in Fig. 23.4 do not have precharge circuits for the nodes connecting the NMOS transistor source/drains. It's a common practice to add such circuits to high fan-in dynamic gates to prevent functional failures due to charge-sharing. Internal precharge circuits introduce additional timing constraints on the clock, and possibly the data inputs, to ensure that all internal nodes are fully precharged.

### 23.4.2 Pass-Gate Logic

The difficulty of verifying the timing of pass-gate logic depends on how constrained the logic is in the design. The less-restricted the use, the more challenging the verification problem. The difficulties arise in identifying valid signal paths and accurately modeling time delays. If pass-gate logic is restricted to implementations of single-level multiplexers, then the path identification problem is simplified. If multiple levels of pass gates can be cascaded and demultiplexers as well as multiplexers are allowed in a design, then path identification is nontrivial for humans and programs. If such freedom is allowed in a design, it is prudent to apply equivalence checking tools or equivalent verification techniques to ensure functional and timing correctness.

Accurate path delay modeling of networks which contain pass-gate logic is difficult because a pass gate presents a low-impedance path between input and output nodes. This makes the delay of pass-gate logic blocks and the delay of gates which fan-out to the inputs of pass-gate logic blocks very data dependent. Clearly cascading levels of pass gates exacerbates this problem and is beyond the capabilities of STA programs. For some STA tools such circuits must be black boxed. A trade-off must be made between the accuracy of the black box timing model and the complexity/verification time of the black box.

There are two propagation delays to consider for pass-gate logic: transistor source/drain input-to-output delay ( $T_{dio}$ ) with transistor gate inputs set up, and gate input-to-output delay ( $T_{dgo}$ ) with source/drain inputs set up. For both cases, accurate load modeling of the input and output nodes, and the drive capability of the input drivers are important. For  $T_{dgo}$  modeling, the edge rate and overlap of gate input signals are also key delay parameters. Typical STA tools assume only one input signal transition at

a time for delay calculations, so all these parameters would need to be accounted for in the black box timing model or by some fudge factor.

### 23.4.3 Self-Timed Logic

Self-timed or asynchronous logic does not rely on a synchronous clock for operation. Conceptually a self-timed logic block has two states: a *ready* state and an *evaluate* state. An enable signal causes the logic to evaluate, and when the logic is done evaluating it returns to the ready state. If the signal indicating the logic has completed evaluation is generated combinatorially the logic is race free.

It is more common in practice for the completion signal of a self-timed block to be generated by a control or “dummy” signal that is designed to have a delay that is guaranteed to be slightly longer than the actual logic delay. The delay of the completion signal must track the delay of the actual logic across process and operating condition variations. This style of self-timed logic is commonly used in embedded memories and on occasion in datapath blocks. Processors completely designed with asynchronous logic are still largely academic research projects [10].

## 23.5 TIMING VERIFICATION OF MEMORY BLOCKS

Today’s STA tools cannot verify custom memory blocks without significant intervention by the circuit designer to guide them. STA tools cannot analyze embedded memories due to problems with: circuit recognition and constraint generation, delay calculation, and path tracing.

Consider the high-level block diagram of a basic SRAM in Fig. 23.5. Most embedded SRAMs are self-timed using pulse generation circuits that are designed to be delay matched with the critical circuit blocks. A read operation begins with a clock transition that enables the row and column decoder evaluation and latching. After a delay for decoding the address, the word line (WL) and column line (CL) are asserted. The sense amplifier enable signal (SAE) is generated from a pulse generation circuit which matches the assertion of SAE with the delay of the word line assertion plus the

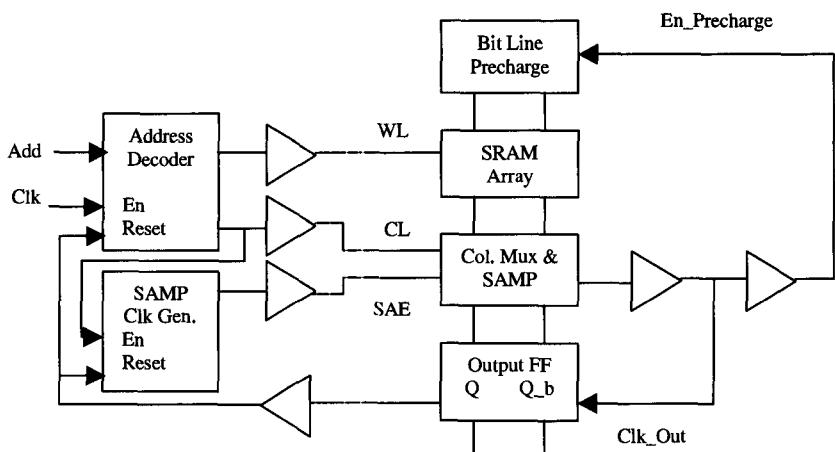


Figure 23.5 Simplified diagram (only read blocks shown) of an embedded memory.

time needed for the required voltage differential to develop across the bit lines. The de-assertion of SAE is matched with the hold time of the output data latch. The WL and CL are also de-asserted after the output FF hold constraint is met. Finally, the bit lines are precharged, and the pulse generator is reset for the next operation. Write operations are generally no more difficult to verify than read operations so they won't be described.

The circuits in memory blocks which STA programs typically don't recognize are: sense amplifiers and other low-voltage swing circuits, clock generation and delay circuits, novel address decoder circuits, complex precharge circuits, dummy devices, and dummy loads. Circuit recognition problems can be overcome via hardwired pattern recognition in the STA program if modifying the tools is an option and the number of difficult circuits is not too large. However, network traversal for all but the smallest memory arrays remains a problem. Accurate loading and delay calculation for differential small-swing circuits is not possible with the timing and load models underlying today's STA tools.

A typical approach for timing verification of embedded memories is to black box the core of the memory array plus any unrecognized memory peripheral circuits such as the sense amplifiers. An STA program is used to identify critical timing paths through the memory block not previously known to the designer. These newly identified paths, and the known critical paths through the entire array, are verified using manually generated simulation models and circuit simulation.

An alternate methodology is to identify a design boundary in the chip that encompasses the embedded memory and accurately model this in the chip RTL model. The vectors for stimulating timing paths within the netlist of the design partition defined by this boundary are generated by RTL simulation. The timing constraints for complex or novel circuit in the netlist must still be manually generated, but the delay calculation problem is solved with a fast circuit simulator. This approach may reduce the manual effort required for timing verification. It also reduces the risk of introducing errors when significant portions of the memory are black boxed. However, there's still the potential for error when timing constraints are manually generated. Another drawback of this approach is the total computer run time of such analyses. The total run time can be prohibitive depending on the memory block size and the performance of the fast circuit simulator.

Regardless of the verification method used, the set-up time analysis for a memory read operation should account for effects that are not typically considered in the timing analysis of logic blocks. Examples of these effects include:

- Worst-case initial bit line voltage differential due to a previous read/write followed by a precharge operation
- Minimum SRAM cell to bit line capacitance ratio
- Worst-case initial SRAM cell voltage levels
- Worst-case bit line, column mux, and sense amplifier parasitic and transistor imbalance
- Maximum address decode and row line assertion delay
- Minimum sense amplifier enable signal assertion and de-assertion delay
- Additional worst-case clock skews if special memory clocks are used

Note that both maximum and minimum delays must be modeled for the set-up analysis of this example memory circuit. Although not explicitly mentioned in the above list, the usual practice of modeling worst-case parasitic loads and capacitive coupling to nodes in the critical path must also be carried out.

Hold-time analysis of the memory read operation for the example memory should account for the factors listed above with the following modifications/additions:

- Minimum row line de-assertion delay is used instead of the maximum
- Minimum precharge assertion delay and minimum precharge delay

Modeling minimum parasitic loads and worst-case capacitive coupling on nodes that are part of minimum delay paths must also be done as with any hold-time analyses.

The added challenge of verifying memories that utilize self-timed clocks is that both set-up and hold-time analyses must be sufficiently accurate and conservative to ensure functional operation. The analyses must be carried out assuming a wider (typically 3 sigma) process variation to ensure reasonable yield. The risk associated with self-timed paths is managed by designing the paths to have delays adjustable via metal mask changes, laser fuses, or control registers. Nevertheless, the design and verification of such circuits are nontrivial.

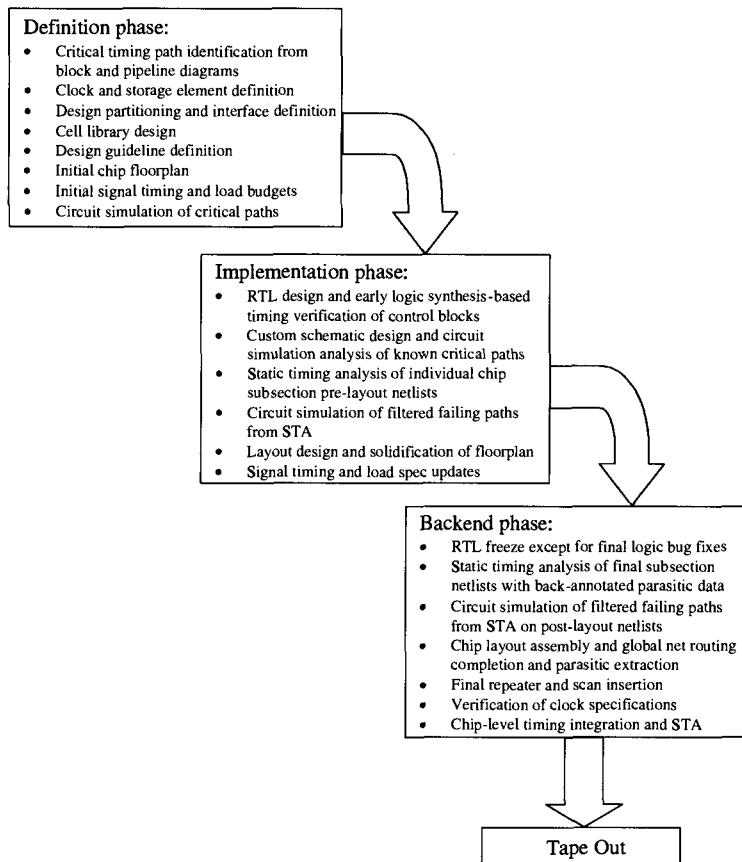
## 23.6 DESIGN FLOW AND FULL-CHIP TIMING VERIFICATION

The highlights, from a timing perspective, of a typical custom processor development flow are shown in Fig. 23.6. While there are many variations on this example design and verification flow, most of the major steps are the same. Timing goals and design assumptions must be consistent from the outset of the design and methodology definition stage. As previously discussed, timing considerations must be factored into the clock and storage element strategy definition, circuit and layout design guideline definition, pipeline design, design partitioning, and early floorplanning.

Timing verification is carried out throughout the definition and development with increasing detail and accuracy. Prior to the existence of a chip RTL model, critical timing paths are identified by noting pipeline stages in which many levels of logic are needed to process a data or control flow and/or signals need to be transmitted to physically distant parts of the chip. The latter consideration implies assumptions are made about the chip floorplan and the process technology is understood well enough to estimate wire RC and signal repeater delays. The validity of any timing analysis is closely coupled to the accuracy of the underlying physical assumptions. This is because wire RC and repeater delays can be a significant portion of a signal path delay, and in addition clock skews are greater between physically distant locations on the die.

Once chip partitions and partition interfaces are defined, signal timing budgets are developed in parallel with the chip floorplan. Timing budgets must specify the earliest/latest signal transition times at output pins, the earliest/latest signal transitions that are allowed to occur at input pins, and delay estimates for a model of the signal time of flight from output to input pins. The signal flight delay is a function of interconnect and gate loading so floorplan assumptions are embedded in the timing budgets. The large number of signal timing budgets necessitates some form of automated consistency checking of the budgets.

Once timing budgets are established, timing verification is carried out on each chip partition as it is designed. Partitions to be designed using logic synthesis tools are typically run through an approximate timing analysis performed by the synthesis tool prior to attempting physical placement and routing of the partition. Timing verification



**Figure 23.6** Timing design and verification flow from product definition.

is repeated after the partition has been placed and routed, and once again after parasitic R's and C's have been extracted and back-annotated to the partition netlist.

Prior to the availability of a complete netlist, analysis of known timing paths through partitions designed at the transistor level is done using manually generated critical path schematics and a circuit simulator. Once a complete transistor netlist is available STA is typically performed prior to the completion of the custom layout design and once again when a netlist that contains back-annotated layout parasitic data is available.

As multiple chip partition designs are completed and run individually through STA, the timing integration process begins. This can be carried out by simply combining smaller partitions into larger ones and verifying timing in this fashion until the entire chip has been assembled and verified. The advantage of this safe but compute and time-intensive process is that errors in the analysis are minimized if the full-chip timing verification can complete successfully. For many large and complex microprocessor designs this method is not an option due to STA tool and/or compute resource limitations.

An alternate method of integrating the chip timing analyses is to combine chip partitions into larger partitions, but to stop this process at some level in the design hierarchy below the chip level. Signal paths that are only completed at a higher level of the hierarchy are verified by extracting a netlist of the circuits that are part of these signal

paths. Layout-extracted parasitic data must be eventually included in this interpartition netlist for timing verification. Once there are no failures with the timing analyses run on the individual chip partitions and the extracted interpartition netlist, the full-chip timing has been effectively verified. The advantage of this approach is that it is much less taxing on the STA tools and requires significantly less compute time. The disadvantage is that it relies on a special interpartition netlist that doesn't exist in the chip design hierarchy and must be generated manually. Back-annotation of layout-extracted parasitic data to this special netlist must also be supported and done correctly. These extra steps and facilities introduce potential for errors during the timing integration process.

The process of timing verification, particularly timing integration, is conceptually simple but the devil is in the details. There are at least three common problem areas in timing verification that become acute during the integration process. The first is managing false paths and false timing violations. The second is managing all the design data and metadata in the face of many and rapid design changes. The last is the determination of when the chip timing goals have been met.

STA tools are deliberately conservative to prevent false positive results. Consequently false negatives occur, that is, timing violations that are flagged but are not real violations. These false violations may be due to the tool tracing a logically impossible path, inaccuracies in delay calculations due to load or timing model errors, a circuit recognition problem, or some combination of these factors. All STA tools support means to suppress false paths from being reported, but in some cases the method by which the reporting of the path is suppressed can prevent real failing paths from being reported. There is also the question of how the paths that are deemed logically impossible are verified as impossible. The problem of false violations resulting from erroneous delay calculations and circuit recognition is often solved by creating black boxes. As mentioned earlier, this process is time consuming, requires additional data management, and increases the risk of errors.

Since timing verification is only as valid as the assumptions made during the analysis, keeping the data which can impact timing up to date in the face of design changes is an enormous challenge. These data at least include:

- Signal timing specifications
- Clock specifications
- RTL model
- Synthesis tool timing and wire delay models
- STA tool technology file and models
- Transistor netlists
- Floorplans and layout design
- Layout-extracted parasitic data
- False path data
- Black box data
- STA output reports
- Any special schematics/netlists and associated circuit simulation data supporting timing verification

Note that if the straightforward approach to timing integration is taken, i.e., combine chip partitions in the timing analysis until the full chip is analyzed, then changes to any of the above data may require rerunning timing verification on the chip.

Because today's STA tools do not recognize or accurately model the timing of some custom logic circuit designs and embedded memories, timing verification of high-performance processors always involves some combination of manual analysis of circuit timing using a circuit simulator and an STA tool. The broader the variety and application of circuit techniques which the STA tool does not accurately analyze or recognize, the greater the reliance on manual analyses and/or extensive black boxing of problematic circuits. If extensive black boxing is not done, too many false errors would be reported and would need to be waived. Usually an effective cycle time for STA purposes is defined which must correlate to the intended target frequency in silicon. The determination of this effective cycle time is both complex and crucial.

In summary, designing and verifying signal timing on a high-performance custom processor is both a technical and a project management challenge. Timing must be considered from design conception through to manufacturing. While CAD tools have improved significantly over the years, there are many shortcomings that must be compensated for by the judicious choice of design methodology and rigorous attention to detail by the design team.

## 23.7 FUTURE CHALLENGES

Competitive forces in the processor markets continue to push microprocessor, embedded processor, and embedded core frequencies higher with each new product generation. The same market forces are also forcing the development cycle of products and process technologies to shrink. Designers are utilizing a wide variety of custom circuit design techniques, integrating greater amounts of embedded and specialty memories on-chip, and taking advantage of new process capabilities such as high-speed interconnects (Cu and low-k dielectrics) and new substrates (e.g., SOI). The combination of these trends results in a tremendous need to develop better tools and methods to verify, prior to first silicon, that custom high-speed processor designs will operate over their intended frequency range.

Currently, seasoned custom IC designers are much better than STA tools at:

- Recognizing the custom transistor level circuits and the timing constraints they impose
- Recognizing false paths
- Correctly applying appropriate off-path load modeling heuristics
- Recognizing and accounting for multiple input state transitions for delay calculations
- Modeling RC effects in a critical timing path
- Finding and modeling signal race hazards

STA tools are much better than designers at:

- Exhaustively and consistently propagating timing through very large networks
- Exhaustively and consistently checking all timing constraints
- Reporting on thousands of paths and timing requirements
- Doing all the above things quickly

Clearly there's a need for STA tools to become better in the areas where designers excel today. In addition, advancements in the following areas are needed to improve or at least maintain design productivity as target frequency goals increase, critical feature sizes shrink, and die size grows:

- Better timing-driven layout planning, synthesis, and STA tools that can analyze high-level design descriptions and/or partially specified designs
- A more complete CAD solution for incremental timing verification, timing integration, and the verification and data management of timing-related data
- Better STA support for multiple clock domains and hierarchies
- More automated and accurate STA support for embedded memories
- Better STA support for wire crosstalk and RC delay analysis, and incorporation of clock and signal min/max time delays in the analysis
- Better tool support for user-defined worst-case conditions for specific timing analyses
- STA timing model improvements and/or automatic selection of models with variable levels of accuracy for more accurate path delay calculation
- Models for new technologies and circuit design techniques associated with these technologies

Several if not all of these areas for improvement are being pursued by CAD and design engineers today. Their success will, to a large degree, determine how long Moore's law will hold true in the years ahead.

## REFERENCES

- [1] D. Bailey and B. Benschnieder, "Clocking Design and Analysis for a 600 MHz Alpha Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 33, no. 11, pp. 1627–1633, 1998.
- [2] J. Silberman, et al., "A 1.0 GHz Single-Issue 64b PowerPC Interger Processor," *IEEE Journal of Solid State Circuits*, vol. 33, no. 11, pp. 1600–1608, 1998.
- [3] P. Gronowski, et al., "A 433 MHz 64b Quad-Issue RISC Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 31, no. 11, pp. 1687–1696, 1996.
- [4] D. Draper, et al., "An X86 Microprocessor with Multimedia Extensions," *IEEE International Solid State Circuits Conference Technical Digest*, pp. 172–173, 1997.
- [5] T. Horel and G. Lauterbach, "UltraSPARC III: Designing the Third Generation 64-Bit Performance," *IEEE Micro*, vol. 19, no. 3, pp. 73–85, 1999.
- [6] N. Vasseghi, et al., "200-MHz Superscalar RISC Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 31, no. 11, pp. 1675–1686, 1996.
- [7] K. Venkat, et al., "Timing Verification of Dynamic Circuits," *IEEE Journal of Solid State Circuits*, vol. 31, no. 3, pp. 452–455, 1996.
- [8] E. J. Shiver, et al., "Timing Verification of the 21264: A 600 MHz Full-Custom Microprocessor," *IEEE International Conference on Computer Design*, Austin, TX, pp. 96–103, 1998.
- [9] A. Dharchoudhury, et al., "Transistor-level Sizing and Timing Verification of Domino Circuits in the Power PC™ Microprocessor," *IEEE International Conference on Computer Design*, Austin, TX, pp. 143–148, 1997.
- [10] R. M. Davies and J. V. Woods, "Timing Verification for Asynchronous Design," *Euro-Design Automation Conference*, pp. 78–83, 1996.

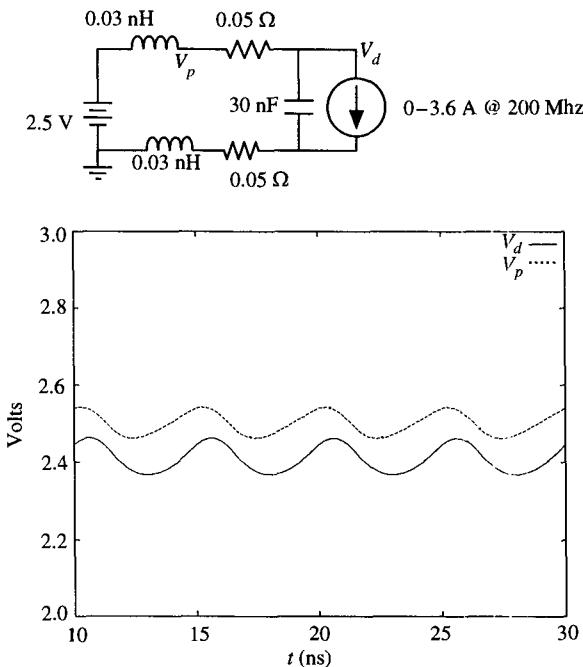
David Blaauw, Rajendran Panda, and Rajat Chaudhry  
*Motorola, Inc.*

## 24.1 INTRODUCTION

Deep submicron dimensions, gigahertz clock frequencies, and transistor counts in excess of 50 million are becoming commonplace in the microprocessor arena. These trends are fueled by the ever-increasing demand for more powerful computers and by rapid advances in process technology, circuit design, and architecture. For large, high-performance microprocessors, the design and verification of power distribution networks have become a critical tasks. These networks distribute power and ground voltages from pad locations to all devices in a design. As they switch and draw current from the power distribution network, a voltage drop develops across the power distribution network due to its resistance. This voltage drop is commonly referred to as the *IR-drop*. Packaging creates another contribution to voltage drop. The package supplies current to the pads of a power grid, either through package leads or through a bump array [24]. Although the resistance of a package lead is quite small, their inductance is significant. Due to the time-varying behavior of the current drawn by circuit devices, lead inductance causes a voltage drop at each pad location, commonly referred to as the  $dI/dt$  voltage drop [1], [10], [7], [17]. The total voltage drop seen at a device is the sum of these two voltage drops:  $V_{device} = Vdd - V_{dI/dt} - V_{IR}$ .

Excessive voltage drop at a distribution point can cause a number of problems. First, the switching time of a gate is increased by the reduced voltage, leading to potential performance problems. Second, the diminished rail-to-rail voltage seen by a device reduces its noise margins. Third, fluctuation in the supplied voltage injects noise into the circuit, making the circuit prone to functional and performance problems. Finally, a high current density can lead to excessive wear on metal lines due to electromigration, causing them to fail over time [2].

Capacitance that connects the *Vdd* and *Gnd* distribution networks reduces the voltage drop at the distribution points. Several parasitic capacitances exist between the power and ground network: parasitic capacitance between the metal wires of the *Vdd* and *Gnd* distribution networks, device capacitances that couple the *Vdd* and *Gnd* networks, and capacitance between the N-well and the substrate. In addition, the designer often adds intentional decoupling capacitance structures on the die at strategic locations. Decoupling capacitance acts as local charge storage and reduces the peak current drawn from the supply, thereby reducing the  $V_{IR}$  and  $V_{dI/dt}$  voltage drops. Figure 24.1(a) shows an abstract view of a power distribution network. The package



**Figure 24.1** Abstract model of a power grid network and its voltage fluctuation.

leads are represented by simple inductors, the on-chip power distribution network by simple resistors, the parasitic decoupling capacitances by a capacitor, and the devices by a time-varying current source. This highly simplified model illustrates the interaction of package inductance with on-chip resistance and capacitance and generates a voltage drop in that has both AC and a DC components. A detailed model of the power grid is presented in Section 24.4. A designer should be mindful of the resonance frequency of the package inductance in combination with the on-chip capacitance,  $f = 1/(2\pi\sqrt{L \cdot C})$  for the simplified model. If this resonance frequency lies close to the clock frequency of the design, dangerous voltage swings can develop in the grid [31].

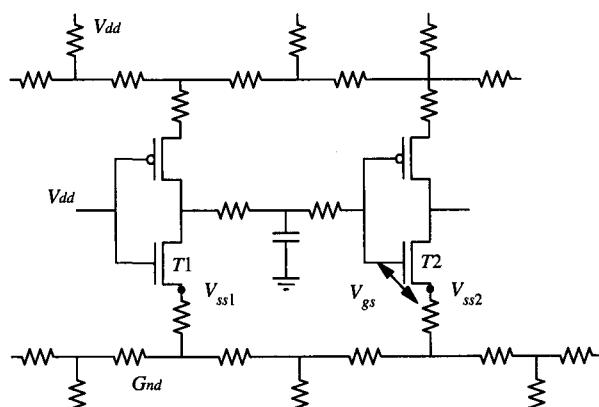
Rapid increases in the complexity and performance of microprocessors has led to a steady increase in their power consumption. Next-generation processors are expected to consume over 100 W. At the same time, the operating voltage of processors is decreasing steadily, due in large part to process scaling. This simultaneous increase in power consumption and decrease in operating voltage requires significant increases in supply currents. Today's high-end processors require power distribution currents in the range of 10 A to 30 A. In the near future, we expect to see microprocessors drawing over 100 A currents. Such high currents are a heavy strain and require that a significant percentage of design effort and chip resources be devoted to the power distribution network. This is already apparent in high-end processors, where a significant percentage of the metallization is devoted to the power grid. In the future, the metal devoted to the power grid and the die area devoted to decoupling capacitances are expected to increase and become significant cost factors in the design of high-end microprocessors. Considering these issues, it is clear that proper power grid design and verification is essential for high-performance design.

## 24.2 POWER DISTRIBUTION DESIGN

### 24.2.1 Power Grid Integrity Problems

Excessive voltage drop at device distribution points can cause performance and functional failures. A performance failure manifests as situations where a circuit malfunctions for high clock frequencies but works as intended when the frequency is reduced. Under a functional failure, the device malfunctions at any frequency. Below, we describe several failures of both types attributable to voltage fluctuations in the power distribution network.

1. Reduced voltage at the devices decreases their drive, thereby reducing processor performance. Typically, a processor is designed with an expected voltage drop budget of between 5 and 10% of  $V_{dd}$ . The simulation and timing verification of the processor are performed while accounting for this budget. However, if the voltage drop in the power grid exceeds the budget, speed-critical devices will be slowed and the chip will not meet its intended performance goal. It is possible that the processor will meet its performance goal for most operations but fail under certain execution sequences, since the voltage drop depends on the executed instructions. This is a particularly difficult problem to diagnose and correct after a chip has been fabricated.
2. Fluctuations in voltage at the power distribution points will inject noise into a circuit, potentially causing a functional failure or performance problem. Figure 24.2 shows a signal net with driver and receiver gates connected to the power and ground networks. We first examine the case where the input of the driver is high, and hence its output is intended to be at a stable low value. Due to currents drawn by other devices in the circuit, the voltages at the source terminals of transistors  $T1$  and  $T2$  will fluctuate. Since transistor  $T1$  is conducting, the voltage of the signal net will follow the voltage at  $V_{ss1}$ . If  $V_{ss1}$  increases while  $V_{ss2}$  simultaneously decreases, the effective gate-to-source voltage ( $V_{gs}$ ) of  $T2$  is increased by the difference between  $V_{ss1}$  and  $V_{ss2}$ . This can cause the output of the receiver to switch. Even if the receiver regains its correct state, the glitch in its output may propagate to a latch altering the processor state and causing a functional failure. Likewise, if the voltage shift across  $V_{ss1}$  and  $V_{ss2}$  occurs when the signal is switching high, the propagation delay to the output of



**Figure 24.2** Schematic view of noise injection from the power grid.

the receiver gate will decrease, possibly resulting in a hold time violation and a functional failure of the circuit. In the case where the signal is switching high while  $V_{ss1}$  decreases and  $V_{ss2}$  simultaneously increases, the gate delay will be increased, causing a performance failure.

The power supply voltage difference between the receiver and driver gates typically increases with the distance between driver and receiver gates on the die. Therefore, nets that transmit signals across the chip will have a higher variation in supply voltage between the driver and receiver gate and thus are most susceptible to power grid noise. Noise injected by the power grid can combine with other noise sources, such as noise from cross-coupling capacitances, and must be taken into account during the noise analysis of the chip [23], [28], [29].

3. High average current densities in a power distribution network lead to excessive wear on the metal wires due to electromigration [2]. Over an extended period of time, this can cause the wires to fail, thereby increasing the power grid voltage drop or causing a complete power loss at certain points. Since electromigration is particularly severe with unidirectional currents, power distribution networks are prone to electromigration problems.
4. The resonance of the package inductance and on-chip decoupling capacitance can result in an oscillation in the supply voltages, leading to periodic above-nominal supply voltages. These higher than intended supply voltages cause hot-carrier-injection in the connected devices, corrupting the gate oxide and degrading the device characteristics. As presented in Section 24.2.5, such voltage oscillations must be carefully controlled to prevent such negative effects.

### 24.2.2 Power Distribution Design Styles

Depending on the available process technology and the power requirements of a processor, designers have used a number of different power distribution network topologies. Power distribution networks can be broadly divided into three classes: routed power networks, power grids, and power planes. High-power processors implemented in advanced processes with six or more layers of metal commonly use either dense power grids or power planes. Smaller processors, which have moderate power requirements and are implemented in technologies with three layers of metal, tend to use routed power distribution networks. Each of the power distribution topologies is explained in detail below.

1. **Routed power distribution networks**—A typical routed power distribution network consists of routed power trunks from pads at the periphery of a chip to each block, which consist of a two layer mesh that distributes power to all gates in the block. Typically, power trunks are routed on two layers, one above the other, with one layer carrying  $Vdd$  and the other  $Gnd$ . In Fig. 24.3, the routed power distribution network for a DSP processor implemented in three layers of metal is shown. Typically, the voltage drop is worst near the center of the chip, the area farthest from any supply pad.

The advantage of routed power grids is that they require only a small portion of the total routing resources available on the chip. This is important for technologies with a limited number of metal layers, where designs tend to be routing limited. The disadvantage is that a routed topology has very little redundancy, since only a few power trunks carry all the current to circuit blocks. If a power trunk is too

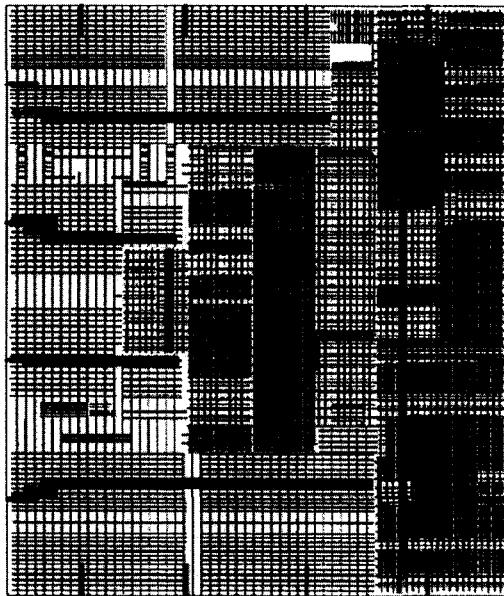


**Figure 24.3** Voltage drops of a routed power grid for a DSP processor.

narrow, current is not easily drawn through other power trunks. Thus, a single undersized trunk segment is detrimental to the integrity of a large portion of the design. Because of this, a high-quality routed power distribution network is difficult to design and requires extensive and careful analysis.

2. **Power grids**—In order to design a power distribution network that is more robust, designers often use power grid topologies for high-performance microprocessors. In a power grid, each layer of metal consists of a number of equal-width power tracks spaced at a constant interval (pitch). The metal layers alternate between horizontal and vertical power tracks, forming a finely meshed grid. Typically, the width and pitch of the power tracks are larger for the higher layers of metal and smaller for the lower layers. This creates a coarse grid for the upper layers where the current is supplied and a fine mesh near the delivery points. A power grid is especially advantageous if the chip is packaged using bump array pads. This package technology allows power pads to be placed at virtually any point on the die. Bump array pads are normally distributed evenly across the die in a matrix configuration. Figure 24.4 shows a high-performance microprocessor design using a power grid.

Power grids have a number of advantages. First, they provide a large number of redundant paths from power supply points to distribution points. This makes them very robust and less sensitive to changes in current distribution, the widths of individual power tracks, or failure of an individual power track segment. Second, since the power grid is inserted between the signal lines on all layers of metal, the coupling capacitance between signal nets is reduced, thereby reducing the cross-coupled noise in the circuit. Finally, a power grid provides a better (shorter) current return path for switching signals. This decreases the inductance of the signal nets, an important issue for high-performance processors. The disadvantage of a power grid is that it consumes a significant portion of the available



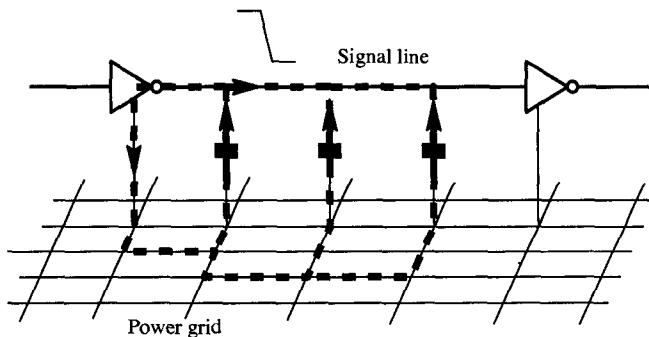
**Figure 24.4** Power grid topology of the PPC 750 processor.

routing resources on the chip. In today's processors, it is not uncommon to use 25% to 40% of all routing tracks for power and ground distribution.

3. **Power planes**—Power planes have at times been used for processors with very high current demands [30]. In a power plane topology, two entire layers of metal are dedicated to power distribution. Typically, one layer is used for  $Vdd$  and one for  $Gnd$ , and the two power planes are separated by one or more layers of signal nets. Holes are created in the power planes to place vias that connect signal nets above the power plane to signal nets below the power plane.

The clear advantage of power planes is that they allow very good current distribution across a die. Especially if a high-performance processor is implemented in a package that supplies power only to the periphery of the chip, this trait is essential. Power planes also significantly reduce the inductance of signal nets, since a return current can take a path directly through the power plane. Naturally, power planes are expensive, as they require two dedicated layers of metal which could otherwise be used for routing.

The topology of a power distribution network has a significant impact on the inductance of signal nets in the chip. When a signal net switches, a current loop is generated with current flowing out from the driver through the signal wire, across to the power grid through capacitive coupling between the wire and the power grid, and returning to the driver gate through the power distribution network. Figure 24.5 shows a schematic view of the current loop. The effective inductance of this current loop is a direct function of the area of the loop and is given by  $L = k \cdot A$ , where  $A$  is the area of the loop and  $k$  is a constant determined by the magnetic permeability and geometry of the loop. The power grid design style impacts the current distribution in the power grid and therefore determines the size of the current loop. In general, routed power networks generate the largest current loops, since there are often no power trunks that are routed close to the signal net.



**Figure 24.5** Schematic of inductive current loop formed by signal net and power grid.

Power grids are considerably better, since each signal net is within no more than a few tracks from the nearest power wire, allowing for much smaller current loops. In a design with power planes, current loops are further reduced since each signal wire is directly above or below a power plane and the separation distance between metal layers is very small. In addition to current loops, other current paths emerge in the power grid when a gate switches, such as the path for short-circuit current of a gate. The inductance of these current paths and their impact on the behavior of signal nets must also be considered in the design of the power grid. As the frequency of microprocessors increases, the inductance of signal nets and its reduction through proper power grid design is becoming an increasing concern.

For simplicity, we will specifically discuss power grid topologies in the remainder of this chapter. The principles presented, however, can be applied to all three power distribution network topologies discussed above.

### 24.2.3 Design Methodology for Power Distribution Networks

When designing a high-performance microprocessor, it is essential that the power distribution network is considered early in the design process. A power distribution network that was poorly designed in the early phases will have numerous violations and will prove impossible to fix with small changes in the final stages before tapeout. A power grid that was correctly architected and designed in the planning stages of the chip will have only a few minor problems during final verification, easily corrected with small local changes to the grid.

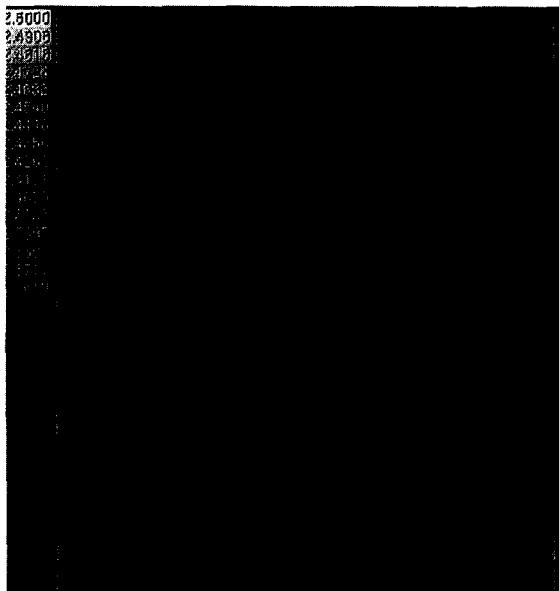
We distinguish three phases in the design of a power grid: early analysis, floorplan-based analysis, and layout-based final verification [13]. These phases are used in succession as the design gradually becomes more defined. In each subsequent phase, the models for the power grid interconnect and for the current drawn by the devices are more refined. Designing the power grid in successive stages ensures that, at each stage, any required changes are manageable. We will discuss the three phases of analysis in more detail below.

#### 24.2.3.1 Early Analysis

The early analysis of a power grid should be started before the floorplan of the design has been finalized. The objective of early analysis is to determine the basic track

widths and pitches for the power grid at each layer of metal. Since the actual circuit implementation is not available, the current distribution is assumed to be uniform across the die and the peak current magnitude is estimated as  $5\times\text{--}8\times$  the average current dissipation projected for the part.

Using the specified metal widths and pitches, a completely regular power grid is constructed. The designer will also specify the expected locations of the bump array pads or peripheral power pads and the package inductance. The designer can make essential trade-offs between the pitches and widths of the power grid at different layers of metal, and the number and location of the needed power pads. Small changes in pitches and widths can significantly reduce the voltage drop of the grid. In one industrial design [13], reducing the pitch and width of metal 3 power tracks by half reduced the total voltage drop by 40%. This change in power grid topology utilized the same percentage of metal for the power grid, although the number of available routing tracks was slightly reduced. Figure 24.6 shows an example of an early analysis power grid for the PPC 750 processor. The worst voltage drop in this analysis was 156 mV, using a peak current of  $5\times$  the projected average current.



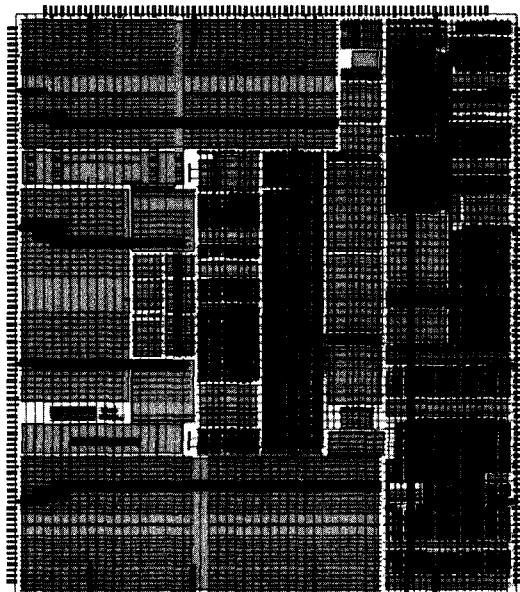
**Figure 24.6** Early analysis voltage drop of the PPC 750 with worst drop of 156 mV.

An AC analysis is also essential in the early design phase to evaluate the effect of package inductance and the adequacy of intrinsic decoupling capacitance. Such analysis will include package inductance models, decoupling capacitances, and synthesized current waveforms [10].

#### 24.2.3.2 Floorplan-Based Analysis

When the floorplan of a design is complete, an estimate of the peak current is made for each block, based either on the experience of the designer, the projected gate count, or on a transistor-level simulation of the block using a fast simulator such as PowerMill [14].

Since the layout of the block is not yet available, the current is assumed to be distributed uniformly across the area of the block. The regular grid defined in early analysis is refined in the floorplan design phase to address the nonuniform power consumption of individual blocks. In Fig. 24.7, a floorplan-based power grid analysis of the PPC 750 is shown with a drop of 172 mV, which compares well with the 156 mV drop in early analysis.



**Figure 24.7** Floorplan-based analysis of the PPC 750 with worst drop of 172 mV.

#### 24.2.3.3 Layout-Based Verification

After the layout of the design is complete, a final analysis verifies that the power grid has an acceptable voltage drop. An accurate RC model is extracted from the layout. The current profile for each individual gate is determined using a fast transistor-level simulator, and is modeled using a time-varying current source. In addition to the resistance of the network, the interconnect and device capacitances as well as the inductance of the package can be extracted and modeled.

Layout-based verification provides the designer with time-varying voltages and currents of all points in a power grid. Based on this information, weak spots or disconnects can be detected. If the grid was properly designed using early and floorplan-based analysis, problems which occur are localized and can be easily fixed. In Fig. 24.8 the layout-based analysis for the PPC 750 is shown at the worst time point in the simulation. The worst voltage drop in layout-based analysis is 170 mV, which corresponds well with the drops in floorplan analysis and early analysis. However, the location of the worst voltage drop is different in each analysis phase, since progressively more detailed current and grid information is used.

#### 24.2.4 Voltage Drop Mitigation

When a problem in the power grid is uncovered, a number of options for reducing voltage drop are available to the designer. If the design suffers from a large  $IR$ -voltage

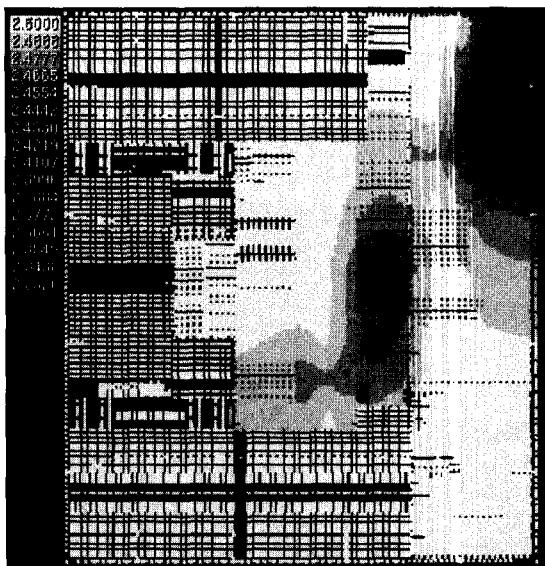


Figure 24.8 Layout-based analysis of the PPC 750 with worst drop of 170 mV.

drop at a particular location, the simplest option is to widen existing wires or add new wires to the power grid. Wire segments with high current densities and large voltage drops are good candidates for adding additional metallization. Some methods have been proposed to automatically determine which wires must be widened [5], [8]. In practice, however, this process is usually performed manually by the designer.

If the voltage drop is primarily in the package inductance, additional decoupling capacitances must be added to the grid [22], [9]. Decoupling capacitances act as local temporary power supplies, dramatically reducing the  $dI/dt$  seen by the package inductance. The metal wires connecting decoupling capacitances to high-current devices may need to be widened to ensure that the RC constants are small. Adding decoupling capacitances, also reduces the resistive voltage drop. Intentionally placed decoupling capacitances may require additional layout area and increase die size. A careful trade-off must be made between power grid integrity and die size.

In a design where voltage drop is dominated by the drop across package inductance, care must be taken when adding more metal. Additional metal reduces the resistance between the package inductance and the on-chip capacitance and can result in larger oscillations in the supply voltage due to resonance. Adding additional decoupling capacitance lowers the resonance frequency of the power grid, and the designer should ensure that the power grid does not start to resonate with the system clock.

In addition to metal and decoupling capacitances, power pads can be added. Additional power pads improve the  $dI/dt$  voltage drop, since the current handled by each individual power pad is reduced. The  $IR$  voltage drop is also improved, since the resistance from the distribution points to the nearest pad location is reduced. Placing bump array pads above areas of high current consumption mitigates voltage fluctuations particularly well.

The process of correcting a power grid is typically an iterative one. In each iteration, a problem in the grid is detected and the power grid is modified to fix the problem. Since this modification will affect voltages in the entire grid, the analysis of the entire grid is repeated in each iteration.

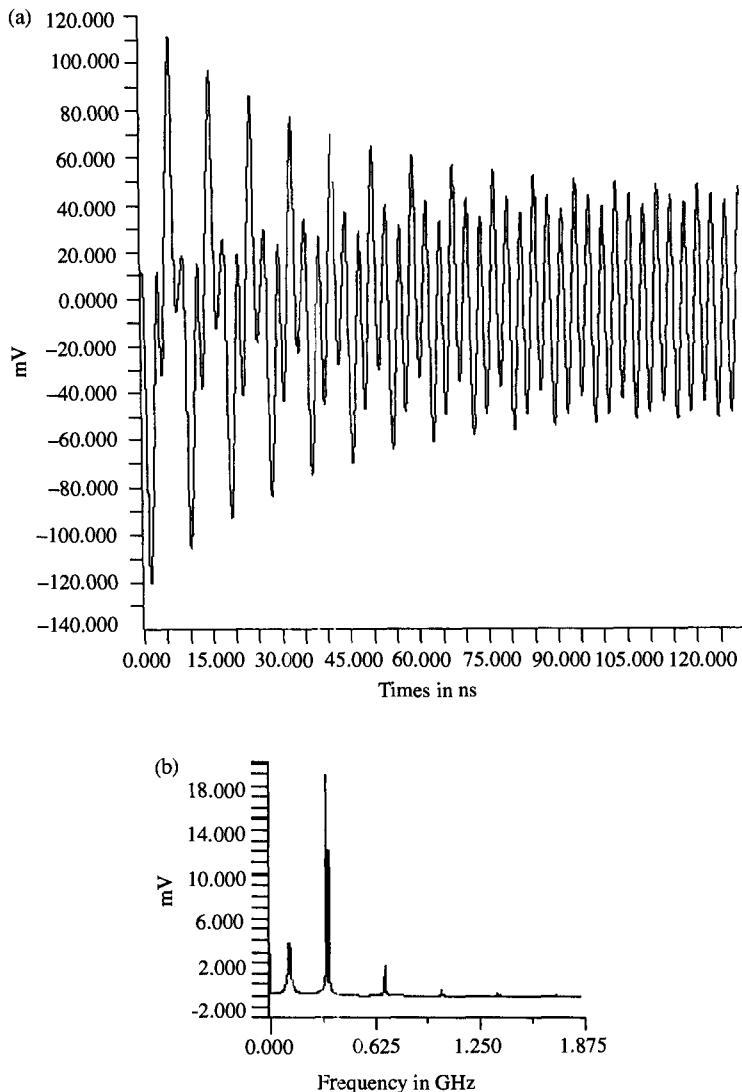
### 24.2.5 Power Grid Resonance

The R, L, C parasitics of the power grid, package, and decoupling capacitances form a multiple resonant system. Of the various resonances, the most significant is that arising from the package inductance  $L$  combined with the decoupling capacitance  $C$  with a resonance frequency of  $f = 1/(2\pi\sqrt{L \cdot C})$ . If the power grid resonates with the processor clock, dangerous voltage fluctuations in the power supply can occur. In the past, the resonance frequency of a power grid was typically much higher than the clock frequency of the processor. Only higher harmonics of the clock frequency could cause resonance in the power grid, and voltage oscillations were relatively small. However, as semiconductor technology has scaled to smaller feature sizes, the resonance frequency of the power grid has steadily decreased. On-chip device decoupling capacitance has increased with scaling while package technology has stayed relatively constant, resulting in a higher  $L \cdot C$  product and lower resonance frequency. At the same time, processor clock frequency has dramatically increased. It is now common for the resonance frequency of a power grid to lie well below the clock frequency of its processor.

It is difficult or impossible for a designer to increase the resonance frequency, since this requires either decreasing decoupling capacitance, which is bounded by the innate device decoupling capacitance, or decreasing package inductance, which is limited by the package technology. There has not been a significant decrease in the package inductance since processors switched to bump array package technology. In Fig. 24.9(a) the transient analysis of a power grid model with package inductance and on-chip decoupling capacitance models is shown for a 330 MHz processor. The analysis shows that the oscillations reach a steady state after approximately 50 cycles. In Fig. 24.9(b), the frequency content of the transient analysis is shown. The spikes at 330, 660, and 990 MHz represent the clock frequency, its first harmonic, and its second harmonic, respectively. The spike at 120 MHz represents the resonance frequency of the power grid.

The fact that the resonance frequency is now less than the clock frequency has introduced two new problems to the power grid designer. First, when the processor switches from a low to a high power state (or vice versa), the power grid will ring for several clock cycles. This ringing generates significant voltage fluctuations and must be considered in the design of the grid. Second, it is possible for the processor to produce a repetitive current pulse with a frequency less than the clock frequency by repeating a sequence of several low-power instructions (such as no-ops) followed by high-power instructions. If a processor happens to execute such a sequence, it could generate a current pulse with exactly the same frequency as the resonance frequency of the power grid. This situation results in large voltage fluctuations and must be examined by the designer of the power grid.

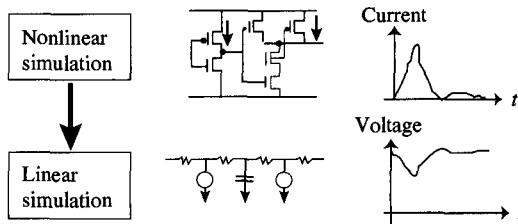
Voltage fluctuations generated by power grid resonance can be reduced by increasing the power grid resistance. Power grid resistance dissipates energy transferred between the inductor and capacitor during oscillations and dampens voltage fluctuations. For this reason, decoupling capacitors are often designed in series with a controlled amount of resistance. However, increased power grid resistance also results in an increased  $IR$ -drop. A careful trade-off must be made between  $IR$  voltage drop and voltage drop due to resonance in the design of a large microprocessor power grid.



**Figure 24.9** Voltage plot of a large RLC power grid and associated frequency content.

### 24.3 POWER DISTRIBUTION ANALYSIS

The major difficulty in analyzing the power grid of a high-performance processor is one of scale. Power grid analysis is a global problem, meaning that the current at one location can affect the voltage at all other locations in the design. The entire power grid must be analyzed simultaneously, and the problem cannot be partitioned without introducing significant error. For a moderately sized processor with 5 million devices, the power grid can easily consist of 10 million or more nodes. For large processors with on-chip memory, the number of nodes in the grid can easily exceed 25 million.



**Figure 24.10** Two-phase approach to power grid analysis.

Furthermore, in order to analyze the voltage drop, the power grid must be solved together with all transistor devices which draw current from it. This would require the simulation of a multimillion element network using a nonlinear simulator, such as SPICE. Since this is clearly infeasible, the common approach is to partition the problem into two parts: nonlinear device simulation, and linear interconnect simulation. This two-phase approach is illustrated in Fig. 24.10. First, a fast nonlinear simulation is used to simulate the devices assuming supply voltages of  $Vdd$  and  $Gnd$ . While simulating, the current drawn by each individual device is recorded. Second, a linear simulation is performed of the power distribution network using a fast linear solver. The linear network consists of resistances, capacitances, and inductances that model the power grid. The currents drawn by the devices are represented by time-varying current sources. The linear simulation calculates the voltage at all tap points of the power grid. The advantage of this two-phase method is that the linear interconnect network can be solved using specialized and very fast linear solvers. Once the nonlinear devices have been separated from the power grid, they are easily partitioned into blocks, and can be simulated in parallel using an efficient nonlinear simulator.

This method of analysis makes the simplifying assumption that the devices in a design are supplied by undiminished  $Vdd$  and  $Gnd$  supplies. In an actual design, a device will see a lesser rail-to-rail voltage, resulting in a lower current. This simplifying assumption is conservative since it results in an overestimation of current, and thus reports a larger than actual voltage drop. If the power grid is well designed and the voltage drop is 10% of  $Vdd$  or less, then the device currents will be overestimated by a similar amount, adding an acceptable amount of conservatism to the analysis. It is possible to iterate on the two-phase analysis using the voltage from the linear simulation of one iteration in the nonlinear simulation of devices in the next iteration. In practice, however, such iteration is rarely performed, as the error in question is small. Also, such iterations are not guaranteed to converge due to the unpredictable impact of rail-to-rail voltage on the timing of the overall circuit.

### 24.3.1 Fast Linear Solvers for Power Grid Analysis

The large size of a power grid places very stringent demands on the run time and memory efficiency of the solver. For large microprocessors, the power distribution network is often designed using a multilayer grid that cannot be reduced significantly using tree-link type transformations. Simulating the power grid requires solving the following system of differential equations, which are formed in a typical approach such as the *Modified Nodal Analysis* (MNA) [26] approach:

$$G \cdot x(t) + C \cdot x'(t) = b(t)$$

where  $G$  is a conductance matrix;  $C$  is the matrix resulting from capacitive and inductive elements;  $x(t)$  is the time-varying vector of voltages at the nodes, currents through

inductors, and voltage sources; and  $b(t)$  is the vector of time-varying current sources. This differential system is very efficiently solved by reducing it to a linear algebraic system  $(G + C/h) \cdot x(t) = b(t) + C/h \cdot x(t - h)$ , using the *Backward Euler* (BE) technique with a fixed time step,  $h$ . The BE reduction with fixed time stepping is advantageous for transient simulation since the left hand side (LHS) coefficient matrix is rendered stationary, allowing either preprocessing or factoring of it for a one-time cost and reusing it efficiently to solve the system at successive time points.

The type of linear solution technique that is most appropriate for a given situation is determined by the size, sparsity, and structure of the LHS matrix, model of the grid (RC or RLC), and the type of analysis (AC or DC). There are two general classes of linear solvers: *direct* [26] and *iterative* [15]. Direct techniques rely on factorizing the LHS matrix once and then using these factors repeatedly in a simple backward and forward substitution procedure to solve the system at every time step. The iterative techniques, on the other hand, rely on efficient convergence techniques to steer the iterations from an initial guess to the final solution. Generally speaking, the direct techniques are extremely attractive for lengthy AC analysis, owing to the inexpensive substitution procedure, provided the factors do not outgrow the size of memory available for computation. The iterative techniques are best suited to solve very large systems using limited memory resources.

The linear system,  $A \cdot x = b$ , of a power network is very sparse ( $10^{-4}$ – $10^{-6}\%$  sparsity). In MNA, matrix  $A$  is also symmetric and positive definite for an RC power grid model. Two special solution techniques, *Cholesky factorization* (direct method), and *conjugate gradients* (iterative method), can provide greater efficiency in such cases, as these methods exploit the symmetry and/or positive definitivity of matrix  $A$ . These methods can be used also for solving RLC grid models in the MNA approach, but with some modifications to address the fact that the resultant matrix  $A$  in that case is indefinite. Alternatively, an RLC model can be formulated as a symmetric positive definite system, albeit with some extra processing, using simple nodal formulation or mesh current formulation and solved directly using the above special techniques.

When the system is extremely sparse, as in the case of a network with abundant tree-like structures, the Cholesky factors remain small making this method an excellent choice. For denser and larger networks, the conjugate gradient technique with preconditioning of the LHS matrix works well. The preconditioner helps to converge to the solution faster. A preconditioner commonly used with conjugate gradients is an incomplete Cholesky factor. Besides the external conditioning, matrix  $A$  is inherently better conditioned when the network contains a lot of capacitance, such as the intrinsic and intentionally added decoupling capacitors. The capacitance entries in a BE formulation make the diagonals heavy and thus the iterations more stable.

### 24.3.2 Simulation Vector Generation

The results of power grid analysis strongly depend on the vectors used during nonlinear device simulation. The goal of a good power grid analysis vector set is to generate the worst realistically possible voltage drop at all locations in the grid. This requires maximizing the instantaneous peak current at all locations. If the grid model includes decoupling capacitance, the peak current should be sustained long enough to discharge these capacitances. Each vector pair should exercise a different part of the design in a worst-case manner, such that all parts of the power grid are covered by the vector set. Clearly, generating a vector set that addresses all these requirements is a very difficult

problem. Furthermore, there are currently no systematic methods for determining the quality of a vector set for power grid analysis.

In practice, a number of different approaches have been used for generating a power grid analysis simulation vector:

1. **User-generated hot-loops**—Generating power grid analysis vectors manually remains the most common approach. The designer constructs a so-called “hot loop” which aims to simultaneously activate as much of the design as possible. He bases these vectors on intuition and knowledge of the design. However, the worst power grid voltage drop requires a large number of gates switching within a very small time window, typically between 1/10 to 1/50 of a clock cycle. It is nearly impossible to have adequate understanding at such a detailed timing level when constructing chip-level vectors and user-generated vectors often leaves large portions of the design inactive. Although there is currently no viable industrial alternative, manually generating high-quality vectors is a very labor-intensive and error-prone process.
2. **Automatic peak current vector generation**—Significant research has been directed toward methods for the automatic generation of high-quality or optimal instantaneous peak current vectors [27], [16], [11]. The application of these methods is typically limited to smaller blocks, making them difficult to use for chip-level analysis. Another disadvantage is that such methods maximize the total current for a block summed across all locations of the block, when the preferred solution would maximize the current of each local area of the grid in at least one vector pair. The advantage of these methods is that they can be used in conjunction with user-generated hot loops to enhance the quality of the power grid analysis.
3. **Static peak current generation**—A number of methods have been developed to statically generate a peak current profile for a circuit block [19], [3], [25]. Instead of dynamically simulating the design with simulation vectors, the circuit is analyzed with a static tool that directly generates the time-varying currents for all power network tap points. There is no need to simulate vectors with a device simulation, making this approach very fast. Most of these methods calculate time windows during which each gate in the circuit can switch [18]. Each gate is assumed to consume its maximum current for the entire duration of its switching window. Since this approach ignores the logical and temporal correlations between gate switchings, it will overestimate current in the circuit. A number of enhancements to this basic method which partially account for such correlations at the cost of additional run time have been proposed [20]. While promising, the overestimation of current using these techniques remains quite high.

Since a high-quality simulation vector set requires a large number of vectors, the run time of the analysis is a significant problem despite the use of fast linear solvers. Vectors vary between several hundred to a few thousand clock cycles, with 20 to 100 time points per clock cycle. For a large processor, each solution point can easily take 5 minutes or more, meaning that a full simulation requires weeks or months of run time. A critical issue is how to compress simulation time points such that a reasonable run time is obtained with acceptable loss in accuracy.

The most common approach is the so-called “single cycle current envelope” compression technique. This approach simulates the full set of input vectors during nonlinear device simulation. Each of the resulting time-varying current signatures is then

compressed into a waveform of one clock cycle. For each time point in the clock period, the pointwise maximum current is taken across all clock cycles, as shown below:

$$I_{\text{envelope}}(t) = \text{Max}(i_{\text{orig}}(t + kP)), \quad 0 \leq k \leq N - 1, \quad 0 \leq t \leq P$$

where  $P$  is the number of time points in the clock period and  $N$  is the number of clock cycles. Since this method does not preserve the correlation between device currents within each cycle, it yields a pessimistic current waveform and overestimates the total current. In some cases, it can increase the reported voltage drop by more than 50% compared to full simulation. Some methods have been proposed which compress currents based on temporal and cyclic correlations, instead of blindly compressing multiple clock cycles into one clock cycle [6], [12]. These methods result in significantly lower overestimation as well as shortened simulation vectors.

In addition to the above-mentioned AC analysis modes, most tools provide a DC analysis mode. In this mode, a DC current distribution is estimated by setting the current for each current tap point in proportion to the size of the connected transistor stack. The switching factor of the device at the tap point can also be accounted for. An advantage of DC analysis is that it does not require the simulation of devices with a nonlinear device simulator and requires only one solution with the fast linear simulator, making it very efficient. DC analysis is useful for detecting gross interconnect errors, but it cannot be used to analyze the effect of package inductance or on-chip decoupling capacitance, which has significant impact on high-performance microprocessor designs.

## 24.4 POWER GRID MODELS

The power distribution network can be modeled in a number of ways, depending on the required analysis. For a DC analysis, only the resistance of the grid needs to be modeled. For transient analysis, the current from devices is represented with time-varying current sources, and the capacitances associated with the power grid must also be modeled. If the designer is interested in analyzing the  $dI/dt$  noise and resonance from the package inductance, the resistance, capacitance, and inductance must all be modeled for the chip and the package. Each of the three modeling aspects (R, C, and L) are presented in more detail below.

### 24.4.1 Resistance Models for Power Networks

The resistance of a power grid consists of the parasitic resistance of the supply pad connections, the power routing wires, the via connections between metal layers, and the contacts at transistor terminals. The resistances of supply pad connections, vias, and contacts are typically provided as part of process characterization. Due to their regular topologies, the resistance of metal wires can be calculated with reasonable accuracy using the sheet resistance at the operating temperature in the following equation:

$$R_{\text{wire}} = (\rho \cdot L)/W$$

where  $\rho$  is the sheet resistance in ohms/square, and  $L$  and  $W$  are units of length and width. If a more accurate resistance calculation is needed for modeling current crowding effects and nonrectangular shapes, a commercial 2D extraction tool can be used.

## 24.4.2 Capacitance Models for Power Networks

A detailed model including the capacitances associated with a power network is often used when the analysis includes time-varying currents. Four capacitance sources need to be considered: (1) the parasitic wire capacitance between power wires and ground wires, substrate, or signal nets; (2) the parasitic capacitance of transistors; (3) the parasitic capacitance of N-well regions; and (4) decoupling capacitors that were intentionally placed between supply lines.

### 24.4.2.1 Interconnect Decoupling Capacitance

Parasitic wire capacitances can be extracted by using approximate precalibrated equations, which define the capacitance as a function of wire width and spacing, or by using more accurate commercial 2D and 3D extraction tools. When considering the coupling between a power wire and a signal net, it should be noted that the signal net may be switching at the time of interest during transient analysis, making it difficult to determine its effect on the power grid. If the signal net is switching high, it can reduce the voltage drop on the power net and, if the signal net is switching low, it can increase it. Even if the signal net is not switching, the influence of the coupling capacitance will depend on the state of the signal net. Coupling from a power (ground) network to a signal net that is high (low) simply couples the power (ground) network to itself, with little or no effect on voltage drop.

It is computationally infeasible to accurately model the state and switching times of all signal nets during power grid analysis. For this reason, a statistical approach is typically used to model the coupling capacitance between the power grid and signal nets. The switching activity is determined by calculating the average percentage of signal nets that switch in a clock cycle. Since all signal nets have equal low and high switching probabilities, it is reasonable to assume that their effects cancel each other out at any given time, and that switching nets can be removed from the analysis. Of the remaining nets, half their parasitic capacitance is assumed to be to the power grid and half to the ground grid. As mentioned above, when a signal net is in a high state only the coupling capacitance to the ground grid acts as decoupling capacitance for the power grid, and vice versa. Therefore, only half of the parasitic capacitance of non-switching nets should be counted as decoupling capacitance. Each coupling capacitance to a signal net is replaced by an effective decoupling capacitor in series with a resistor. The value of the effective coupling capacitor is

$$C_{eff} = \frac{1}{2} C_{coupling} \cdot (1 - P_{active})$$

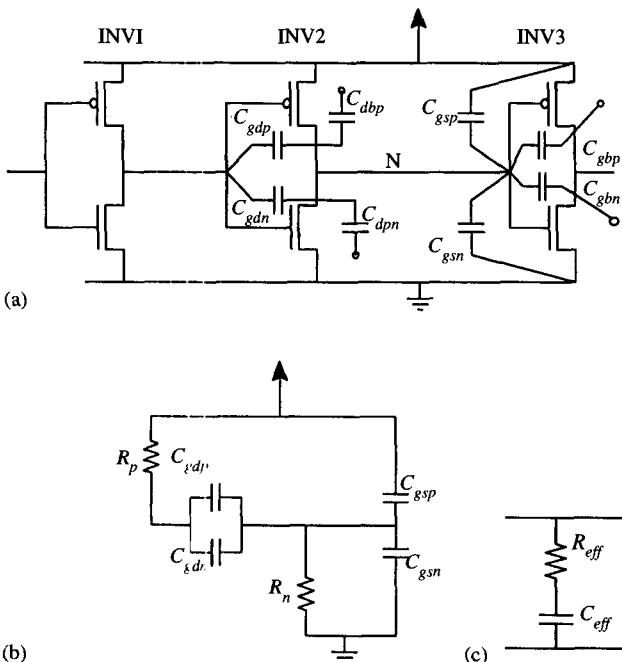
where  $P_{active}$  is the average switching activity. The resistor models the effective resistance of the gate holding the signal net in its stable state, and can be either an average value over all gates or a specific value for each particular gate.

### 24.4.2.2 Device Decoupling Capacitance

Parasitic device capacitances decouple a power grid and are typically much larger than the parasitic interconnect capacitances of a power network. As with interconnect capacitances, the effect of parasitic device capacitances on a power grid depends on the signal state, and is best modeled statistically. Each transistor (pmos and nmos) has five

device capacitances:  $C_{sb}$  (source to bulk),  $C_{db}$  (drain to bulk),  $C_{gs}$  (gate to source),  $C_{gd}$  (gate to drain), and  $C_{gb}$  (gate to bulk). Of these five, the  $C_{sb}$  can be ignored, since the source of the pmos (nmos) is connected to  $Vdd$  ( $Gnd$ ), and the bulk of the pmos (nmos) is also tied to  $Vdd$  ( $Gnd$ ) through a well tie (substrate tie). This means that the  $C_{sb}$  couples each power grid back to itself and does not affect the voltage drop. In Fig. 24.11(a), the four remaining device capacitances are shown for an inverter. Although the capacitances are shown for only one inverter, they are arranged across three inverters to make the analysis more convenient. If the gate is switching, the device capacitances contribute to the current drawn from the power grid, which is already modeled by a time-varying current source in the power grid analysis. Therefore, only the device capacitances of nonswitching gates must be modeled.

We first examine the case where net N in Fig. 24.11(a) is in a low state. The device capacitances can be modeled with an equivalent RC circuit, shown in Fig. 24.11(b). The resistance  $R_p$  and  $R_n$  are the small signal resistance of the pmos and nmos transistors of inverter 1 and 2, respectively, biased at  $V_{ds} = 0$  and  $V_{gs} = Vdd$  for the nmos, and  $V_{gs} = -Vdd$  for the pmos. Since net N is low, capacitances  $C_{dbn}$ ,  $C_{gsn}$ , and  $C_{gbn}$  are discharged and do not contribute to decoupling capacitance. Furthermore, the well resistance is relatively high and since  $C_{dbp}$  and  $C_{dbn}$  are small, they can be ignored without significant loss in accuracy. To further simplify the circuit, we note that the impedance of  $C_{gsn}$  is much larger than  $R_n$  at the frequency of interest, and hence  $C_{gsn}$  can be ignored. Finally, we lump  $C_{gdp}$  with  $C_{gdp}$  and  $C_{gdn}$  with a small loss in accuracy. The circuit in Fig. 24.11(b) is therefore modeled with the simple model in Fig. 24.11(c), where  $C_{eff} = C_{gdn} + C_{gdp} + C_{gsp}$  and  $R_{eff} = R_p + R_n$ .



**Figure 24.11** Device decoupling capacitances with their RC and decoupling cap models.

An analogous analysis can be made for the decoupling capacitances when the state of signal N is high. The state of node N is assumed to have equal probability of being either high or low, although a different ratio could easily be incorporated into the analysis. The effective decoupling capacitance accounting for the state of N is the sum of the effective high and low decoupling capacitances weighted by the probability of the gate being in either state:

$$C_{eff} = (1 - P_{active})(C_{gdp} + C_{gdh}) + \frac{1}{2}(1 - P_{active})(C_{gsp} + C_{gsn})$$

The proposed model is also extensible to stacks of multiple transistors in series by adding the capacitance of the intermediate nodes of the stack to  $C_{eff}$ .

#### **24.4.2.3 N-well Capacitance**

The parasitic capacitance between the N-well and substrate acts as a decoupling capacitance for the power grid. In series with this capacitance is the resistance of the well from the well tie to the well/substrate junction. The N-well decoupling capacitance can be approximated with a model similar to that shown in Fig. 24.11(c), where  $C_{eff}$  is the capacitance of the N-well junction and  $R_{eff}$  is the effective resistance from the well tie to the well junction. The value of  $C_{eff}$  and  $R_{eff}$  are a function of the well size, well depth, the number well ties, and the doping profile.

#### **24.4.2.4 Decoupling Capacitance Structures**

In order to increase the total decoupling capacitance of the power grid, designers often add specific decoupling capacitance structures on the chip. These structures consist of transistors with very large active areas where the source and drain are connected to one power grid and the gate to the other. Two factors need to be considered when designing decoupling capacitance structures. First, since the gate oxide area of the decoupling capacitance is very large, it is susceptible to manufacturing defects that short the power and ground grids and yield inoperable parts. To avoid a negative impact on the yield of a part, it is necessary to place a fuse or shut-off transistor in series with the decoupling transistor to allow the part to recover from a gate-oxide breakdown in the decoupling structure. When modeling the decoupling capacitance, the series resistance of this structure must also be modeled. Second, as explained further in Section 24.2.5, adding additional decoupling capacitance lowers the resonance frequency of the part, which may cause undesirable oscillations in the power supply. Increasing the resistance of the fuse or shut-off transistor will help dampen these oscillations. However, increasing the resistance too much renders the decoupling capacitance ineffective in reducing  $dI/dt$  noise. Therefore, the decoupling capacitance structure must be carefully designed and tuned to reduce  $dI/dt$  noise while not creating large oscillations [21].

### **24.4.3 Inductance Models for Power Networks**

For high-performance microprocessors, the change in current ( $dI/dt$ ) at pad locations can be quite high, causing significant voltage fluctuations. Hence, it is important to include an accurate model of the package in the power grid analysis. Figure 24.12(a) shows the cross section of a typical bump-array ceramic package. The power pads on the chip are connected with a solder ball and via to the power planes in the ceramic package.

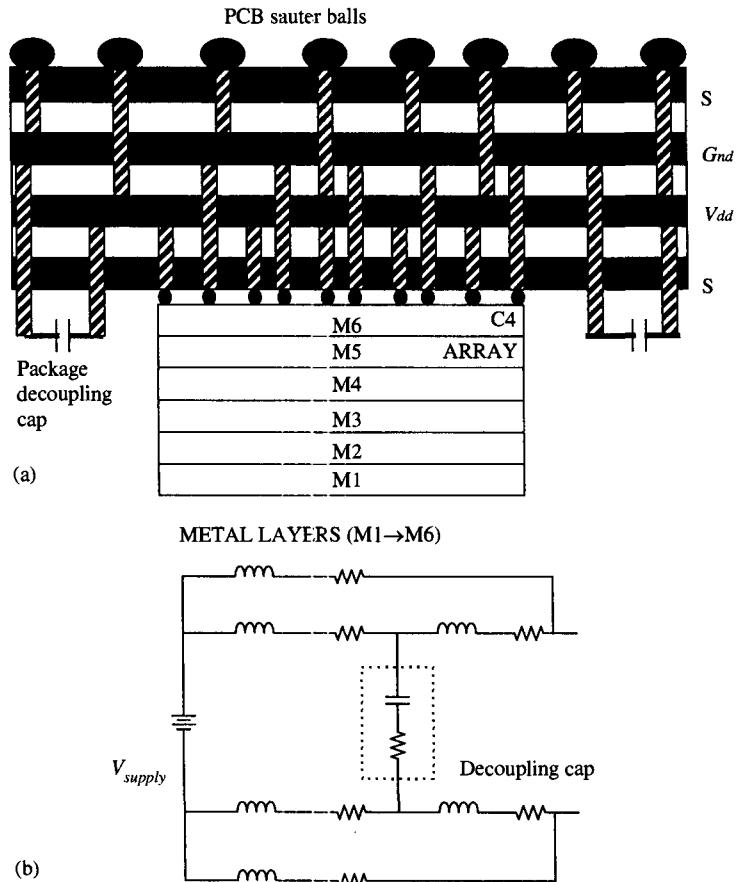


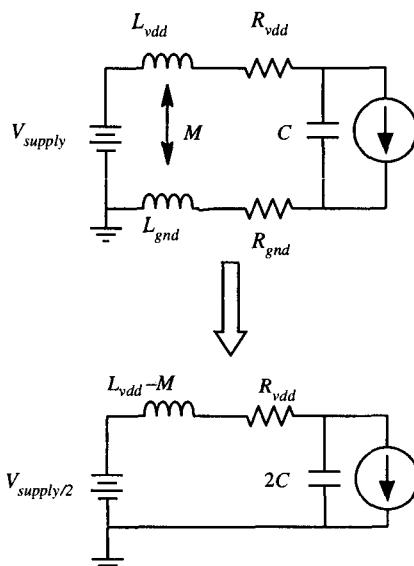
Figure 24.12 Ceramic package cross section and circuit model.

A second via connects the power plane to the printed circuit board at a number of locations. The power planes of the ceramic package are typically connected to package decoupling capacitors. Figure 24.12(b) shows a simplified package model that can be used for power grid analysis. If the package includes dedicated decoupling capacitances, these must be modeled as well. It is important to include the intrinsic series inductance and resistance of the package decoupling capacitor, which limits their effectiveness at high frequencies. Typically, package decoupling capacitances are not effective above 500 MHz, increasing the importance of on-chip decoupling capacitances.

Since the number of power connections in a bump array is typically high (several hundred), the package model is quite large. Each lead from the package has a resistance, a self-inductance, and inductive coupling with other leads. The package model forms a dense (partial) inductance matrix [4] in which there is a mutual inductance between each pair of entries. To reduce computational complexity a package leads can be grouped into equivalent leads, each providing current to a group of pads within close proximity. This significantly reduces the size of the partial inductance matrix to be solved.

To correctly analyze voltage drops at the devices, the current through the  $V_{dd}$  and  $Gnd$  leads must be simultaneously modeled, since  $V_{dd}$  and  $Gnd$  power leads are

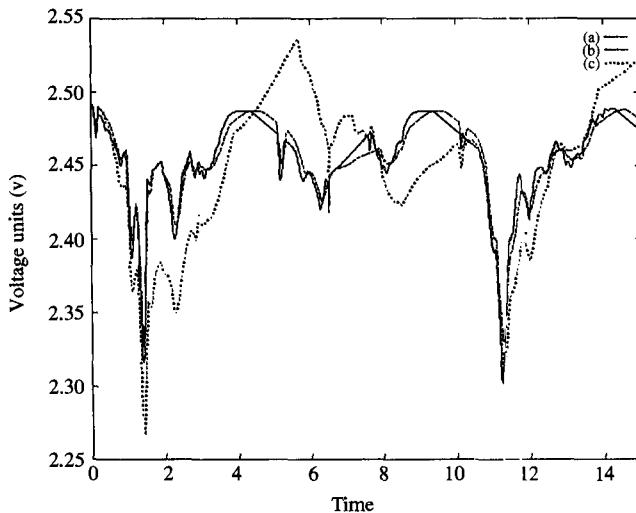
inductively coupled. Modeling only one type of grid ( $V_{dd}$  or  $Gnd$ ) ignores the inductive coupling between the  $V_{dd}$  leads to the  $Gnd$  leads and will lead to erroneous results. Analyzing only one grid in isolation also reduces the charge provided by on-chip decoupling capacitances, since instead of seeing the sum of the  $V_{dd}$  and  $Gnd$  voltage drops, only the drop from one of the two grids is seen. However, modeling both grids simultaneously requires the analysis of a network that is twice as large as is traditionally analyzed. An alternative is to make the simplifying assumption that the two power grids are identical and that the current distribution in each grid is identical. Based on this symmetry, the two coupled power grids can be modeled with two (identical) isolated grids, as shown in Fig. 24.13. In the isolated grids, the decoupling capacitance is doubled and the mutual inductance between the package leads of the two grids is subtracted from the inductance of each package lead.



**Figure 24.13** Equivalent single grid model under symmetry assumption.

#### 24.4.4 Comparison between R, RC, and RLC Analysis

To illustrate the impact of parasitic decoupling capacitances and package inductance on power grid analysis, Fig. 24.14 shows the analysis results of three types of power grid models. The first curve (a) shows a transient analysis of the PPC 750 processor when only the resistance of the power grid is modeled. The second curve (b) shows the same transient analysis with parasitic decoupling capacitance included. The total on-chip decoupling capacitance contributed by devices and interconnect was 30 nF in this analysis. No explicitly placed decoupling capacitances were modeled in this analysis. The worst voltage drop was reduced from 199 mV to 170 mV by adding decoupling capacitances. The third curve (c) shows the voltage drop when package inductance was incorporated into the analysis. The worst voltage drop increased from 170 mV to 233 mV when the package inductance was included. Also note that pad voltages fluctuated with change in current from the devices.



**Figure 24.14** Analysis results of the PPC 750 with (a) R, (b) RC, and (c) RLC power grid models.

## 24.5 CONCLUSION

Power grid design has become a significant challenge for high-performance microprocessors. With voltage drops often exceeding 7% of  $V_{dd}$  and more significantly impacting chip performance, the power grid designer has an important impact on the frequency and robustness of a part. The analysis of power grids is complicated by their sheer size and the difficulty of accurately modeling the intrinsic device and interconnect decoupling capacitance and package inductance. The power grid also has a strong impact on signal net inductance and has a resonance frequency which is typically lower than the clock frequency for today's processors. In this chapter, we presented a design and analysis methodology for analyzing large microprocessor power distribution networks. Also, models for device and interconnect decoupling capacitance and package inductance were presented. Different techniques for simulating very large power distribution networks were discussed. Results from actual processor designs were shown to illustrate the presented approaches. With the continued scaling of semiconductor technology, power grid design and analysis is likely to become an even more challenging task.

## REFERENCES

- [1] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*. Addison-Wesley, Reading, MA, 1990.
- [2] J. R. Black, "Electromigration Failure Modes in Aluminum Metalization for Semiconductor Devices," *Proc. IEEE*, pp. 1587–1594, Sept. 1969.
- [3] R. Burch, F. Najm, P. Yang, and D. Hocevar, "Pattern-Independent Current Estimation for Reliability Analysis of CMOS Circuits," *Proc. 25th Design Automation Conference*, 1988.
- [4] P. Brennan, N. Raver, and A. Ruehli, "Three-dimensional Inductance Computations with Partial Element Equivalent Circuits," *IBM Journal on Research and Development*, vol. 23, no. 6, Nov. 1979.

- [5] S. Chowdhury and M. A. Breuer, "The Construction of Minimal Area Power and Ground Nets for VLSI Circuits," *Proc. 22nd Design Automation Conference*, 1985, pp. 794–797.
- [6] R. Chaudhry, D. Blaauw, and R. Panda, "Vector Compression for Power Grid Analysis," *Proc. IEEE Design Automation Conference*, June 2000.
- [7] H. H. Chen, "Minimizing Chip-level Simultaneous Switching Noise for High-performance Microprocessor Design," *IEEE Int. Symp on Circuits and Systems*, vol. 4, 1996.
- [8] S. Chowdhury, "Optimum Design of Reliable IC Power Networks Having General Graph Topologies," *Proc 26th Design Automation Conference*, June 1989.
- [9] Clayton R. Paul, "Effectiveness of Multiple Decoupling Capacitors," *IEEE Trans. on Electromagnetic Compatibility*, vol. 34, no. 2, May 1992.
- [10] H. Chen and D. Ling, "Power Supply Noise Analysis Methodology for Deep-submicron VLSI Chip Design," *Proc. Design Automation Conference*, 1997.
- [11] D. Ciplickas and R. Rohrer, "Expected Current Distribution for CMOS Circuits," *Proc. International Conference on Computer-Aided Design*, pp. 589–592, 1996.
- [12] R. Chaudhry, D. Blaauw, and R. Panda, "Vector Compression for IR-drop Analysis," submitted to DAC, 2000.
- [13] A. Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuiianu, and D. Bearden, "Design and Analysis of Power Distribution Networks in PowerPC Microprocessors," *Proc. ACM/ IEEE Design Automation Conference*, pp. 738–743, 1998.
- [14] *Powermill Reference Manual Epic Design Technology*, 1996.
- [15] G. Golub and C. Van Loan, *Matrix Computations*. Johns Hopkins Univ. Press: Baltimore, MD, 1989.
- [16] A. Krstic and K. T. Cheng, "Vector Generation for Maximum Instantaneous Current through Supply Lines for CMOS Circuits," *Proc. ACM/IEEE Design Automation Conference*, pp. 383–388, 1997.
- [17] G. A. Katopis, "Delta-I Noise Specification for a High-performance Computing Machine," *Proc. of the IEEE*, vol. 73, pp. 1405–1415, 1985.
- [18] H. Kriplani, F. Najm, and I. Hajj, "Maximum Current Estimation in cmos Circuits," *Proc. Design Automation Conference*, 992.
- [19] H. Kriplani, F. Najm, and I. Hajj, "Pattern Independent Maximum Current Estimation in Power and Ground Buses of CMOS VLSI Circuits," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 8, pp. 998–1012, Aug. 1995.
- [20] H. Kriplani, F. Najm, P. Yang, and I. Hajj, "Resolving Signal Correlations for Estimating Maximum Currents in CMOS Combinational Circuits," *Proc. 30th Design Automation Conference*, pp. 384–388, 1993.
- [21] P. Larsson, "Parasitic Resistance in an MOS Transistor Used as On-chip Decoupling Capacitance," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 4, April 1997.
- [22] P. Larsson, "Power Supply Noise in Future IC's: A Crystal Ball Reading," *IEEE Custom Integrated Circuits Conference*, San Diego, CA, 1999.
- [23] J. Lohstroh, "Static and Dynamic Noise Margins of Logic Circuits," *IEEE J. Solid-State Circuits*, SC-14, pp. 591–598, June 1979.
- [24] L. Miller, "Controlled Collapse Reflow Chip Joining," *IBM Journal of Research and Development*, vol. 13, no. 3, pp. 239–250, 1969.
- [25] F. Najm, R. Burch, P. Yang, and I. Hajj, "CRESTMA Current Estimator for CMOS Circuits," *Proc. International Conference on Computer-Aided Design*, pp. 204–207, 1988.
- [26] L. Pillage, R. Rohrer, and C. Visweswarah, *Electronic Circuit and System Simulation Methods*. McGraw-Hill, New York, 1994.
- [27] S. Chowdhury and J. S. Barkatullah, "Estimation of Maximum Currents in MOS IC Logic Circuits," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 6, 642–654, June 1990.
- [28] K. L. Sheppard and V. Narayanan, "Noise in Deep Submicron Digital Design," *Proc. ACM/ IEEE Design Automation Conference*, pp. 524–531, 1996.

- [29] K. L. Sheppard, V. Narayanan, P. C. Elmendorf, and G. Zheng, "Global Harmony: Coupled Noise Analysis for Full-chip RC Interconnect Networks," *Proc. Intl. Conf. Computer-Aided Design*, pp. 139–146, 1997.
- [30] D. Stark, "Analysis of Power Supply Networks in VLSI Circuits," *Research Report 91/3*, Western Research Lab., Digital Equipment Corp., April 1991.
- [31] S. Taylor, "The Challenge of Designing Global Signals in UDSM CMOS," *IEEE Custom Integrated Circuits Conference*, San Diego, CA, 1999.

# TESTING OF HIGH-PERFORMANCE PROCESSORS

Dilip K. Bhavsar  
*Compaq Computer Corporation*

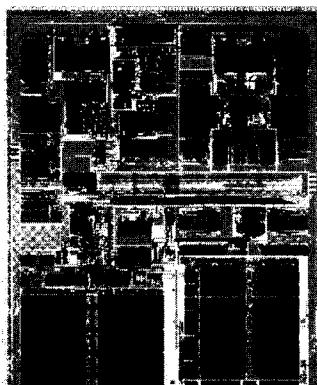
This chapter provides an overview of the principles of testing and their applications to the testing methodologies for high-performance microprocessors.

## 25.1 INTRODUCTION

*Testing is indispensable.* This is a fact of life. Our raw materials are imperfect. Our IC manufacturing processes are imperfect. We have to test an IC to ensure that it is defect free and that it conforms to its specifications. That is, it delivers the functionality and performance we promise to customers.

*Testing is eternal.* Although we design an IC only once, we test it until it retires. The testing-related activities begin in the design phase in the form of design for testability. An end user continues testing the IC in an end-product environment as IC performs useful function. Occasionally, the IC fails or malfunctions while performing its normal function. The failed IC is returned to its manufacturer for further testing, diagnosis, and failure analysis.

*Testing is a challenge.* Figure 25.1 shows a microphotograph of the most recent Alpha microprocessor designed using some of the circuit techniques discussed in the preceding chapters. It is a 15 million transistor device built in a  $0.25\text{ }\mu\text{m}$  6 metal layer CMOS process. How does one test a device like this? How does one make sure that all of the 15 million transistors are built and connected correctly and function properly? How will we test the next wave of ICs with 300 + million transistors and an operating speed of a few GHz?



**Figure 25.1** Alpha 21264 64-bit RISC microprocessor.

We address the testing challenge in this chapter. We begin with a review of some basic concepts of IC testing in Section 25.2. In Section 25.3 we introduce the concepts of testability and design for testability and provide an overview of the techniques and tools of design for testability. We summarize and conclude the chapter in Section 25.4.

Although most of the concepts in this chapter are general and applicable to any integrated circuit built in any process technology, we have focused our discussions toward high-performance microprocessors built in CMOS.

## 25.2 BASIC CONCEPTS IN TESTING

### 25.2.1 Components of Testing

Figure 25.2 shows typical test flow at the final two stages of IC fabrication. At wafer probe each die on wafer is tested using *automatic test equipment* (ATE). Modern microprocessors contain tightly packed large embedded SRAMs. These SRAMs can reduce the microprocessor yield. Therefore most microprocessors also employ some form of redundancy scheme to boost production yields. The repair is performed at the laser repair step. After repair, the IC may undergo a second wafer probe step to verify the repair as well as to complete the testing of all other functions whose testing may have been hampered due to bad on-chip RAMs. Since the packaging step may add substantial processing and material cost to the IC, it is important to weed out all defective ICs at wafer probe.

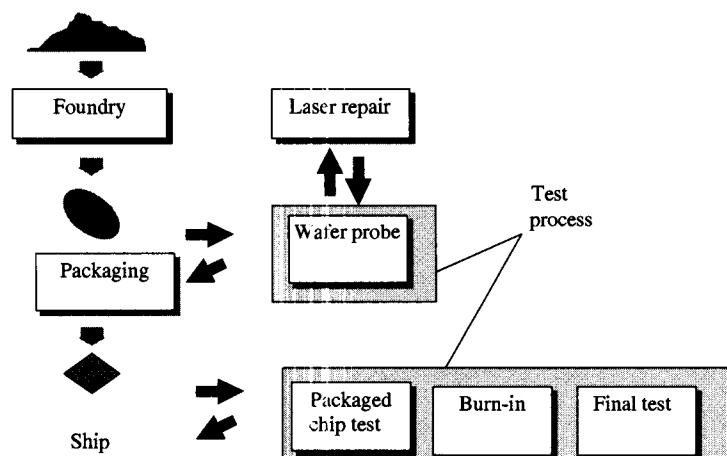


Figure 25.2 Test process in chip manufacturing.

After packaging, ICs are tested again using ATE and techniques similar to those used during wafer probe. At this step the IC's at-speed performance and various electrical parameters may be tested with precision for the first time. Such testing is generally difficult in the wafer probe environment. Burn-in step is used for weeding out ICs with infant mortality. ICs are placed in a burn-in chamber and subjected to voltage and temperature stress over an extended period of time while being tested. Finally, the post-burn-in test step repeats the pre-burn-in tests and performs the final screen before shipping.

At any stage of manufacturing, testing consists of three components: (1) fault modeling, (2) test development, and (3) applying test or the actual testing. We will look at each of these in some detail.

### 25.2.2 Fault Modeling and Testing Methods

A processed wafer may have a wide range of physical defects such as oxide pin holes, layer separation, metal misalignment, voids, shorts, metal bridge, missing contacts, parasitic p-n junctions, and so on. Although these defects are physical in nature, it is not practical to check for their presence visually in the production screen. But we can test for the effect of their presence on behavior of the individual IC. Therefore, the first step in testing is to target or select physical defects that may cause the device to malfunction. Then, we must devise an appropriate *test method* that can purposely excite the defect and induce the device to malfunction such that the deviation from the normal specified behavior can be easily measured or observed by the test process. We will call this activity *defect-based testing*.

A *test method* consists of applying stimuli to the device under test and gathering and comparing its responses with those expected from a defect-free device. All practical testing methods today use voltage levels as stimuli. They are applied at the device inputs under a varying set of parametric conditions such as ambient temperature, supply voltages, and the rate of change of stimuli while observing the electrical behavior of the device outputs.

Testing as practiced today takes only a gross view of the physical defects and uses their approximations called *fault models*. Fault models are abstractions of physical defects assumed to occur in abstract representation of design such as transistor, gate, register transfer level (RTL), or functional level. They provide a means for test development, a criterion for evaluating the effectiveness (coverage) of the selected testing method. The subject of fault modeling and testing methods is extensively treated in the literature [1], [2] and [9]–[11]. We will briefly describe the most popular among them.

#### 25.2.2.1 Stuck-At Fault Models

The stuck-at fault model targets shorts between gate input or output and power and ground lines. Consider the CMOS complex gate and its layout shown in Fig. 25.3. Bridges (shorts) between the adjacent metal routes within the metal-1 layer at sites 1, 2, and 3 cause stuck-at-1 faults on input A, and stuck-at-1 and stuck-at-0 faults on output Z, respectively. Of course, several other bridges are possible that could cause the circuit to malfunction, but these are outside the scope of the stuck-at fault model.

Most test development tools assume that faulty circuits can have only one stuck-at fault at a time. This is called *classical single stuck-at fault model*. It is customary for these tools to assume that all faults are equally likely and to develop tests to detect inputs B and C stuck-at-1 and input A stuck-at-0 faults, although from the layout geometry these faults are unlikely to occur.

#### 25.2.2.2 Delay Fault Models

The delay fault model targets those defects that cause malfunction at the IC's specified clock frequency but not at lower frequencies. Such malfunction may result

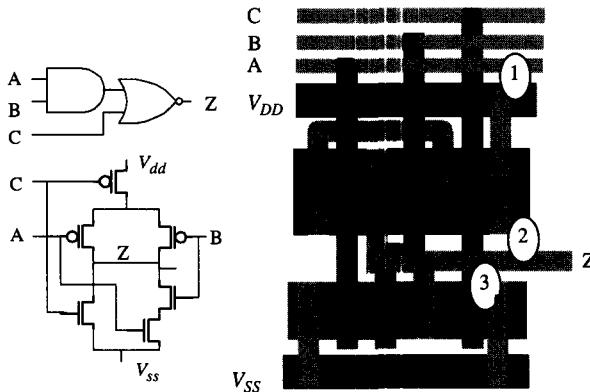


Figure 25.3 Stuck-at fault model.

from the random point defects or from the gross process variations. Two delay fault models are currently popular: the gate delay and the path delay fault model.

The gate delay fault model assumes that the delay faults are lumped at a faulty gate, causing increased propagation delay through the gate. The model affects all paths passing through the affected gate. This model is also known as transition fault model. The path delay fault model assumes that the delay defect is spread over the gates along a path. This model targets delay defects induced by process variations which affects many gates.

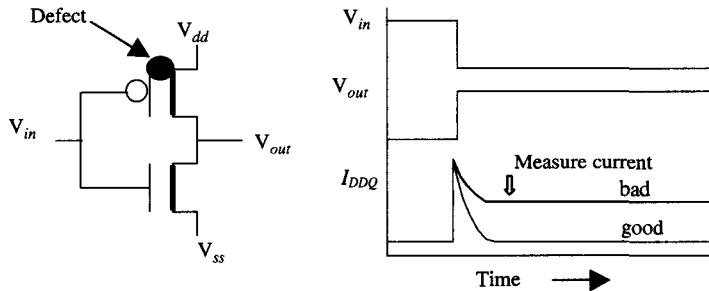
#### 25.2.2.3 CMOS Leakage Fault Models and $I_{DDQ}$ Testing

Many physical defects, such as gate oxide shorts, power supply rail shorts, signal node bridges, leaky p-n junctions, punch-through, parasitic transistors, open drain and source, and open gate, elevate  $I_{DDQ}$ , the quiescent power supply current drawn by a CMOS IC. The leakage fault model targets these defects. The  $I_{DDQ}$  testing method ([6]) consists of supplying a test vector that enables leakage paths postulated by the leakage fault model and making a current measurement after all switching transients have subsided. The device is considered bad if the measured current exceeds the established threshold. Figure 25.4 illustrates the basic principle of the  $I_{DDQ}$  testing method.

#### 25.2.2.4 Functional Fault Models

The models we discussed so far postulate faults on the structure of logic circuits. The functional fault models on the other hand postulate faults at the functional behavior level. For example, in a microprocessor ([7]), functional fault modeling may assume that faulty instruction decode logic may cause a wrong instruction or no instruction to be issued. Likewise faulty register file decode logic may select a wrong register or no register at all.

Although the study in [7] claimed correlation between test coverage of functional fault models and stuck-at fault model for a microprocessor, it cannot be considered adequate for complexities found in today's microprocessors. A serious test development effort rarely relies on this approach. Functional fault modeling, however, is widely used for testing random access memory [4], [8]. This approach tests for faults such as: storage

Figure 25.4  $I_{DDQ}$  testing principle.

cell stuck-at zero or one, certain address location cannot be read or written, two address locations are addressed simultaneously, and so on. Functional fault model is very effective in testing simple, regular structures with a limited number of operations. It requires small test development effort.

Despite its narrow focus on one type of defect, stuck-at fault model is widely used in the industry. Test development tools handle it efficiently and automatically. A high stuck-at coverage is a prerequisite for obtaining high delay fault coverage and studies have found it to correlate with multiple stuck-at and bridging fault coverage.

Several studies have tried to compare relative strengths and weaknesses of various fault models and test methods ([9]–[11]). In 1997 a study sponsored by SEMATECH [11] compared four test methods on production runs of an ASIC ( $0.45\mu\text{m}$  CMOS device with approximately 500 K transistors). Figure 25.5 shows the pass/fail count of the 2892 *delta devices* (devices failing at least one but not all of the test methods) at the packaged part testing step. The highlighted entries show the exclusive fails for a test method. Although the functional test used in the study had poor stuck-at fault coverage (52%), the result indicates that no single test method is foolproof in detecting all defects and that in order to weed out all defects a production screen may need to combine all test methods.

		$I_{DDQ}(5 \mu\text{A} \text{ limit})$			
		Pass	Pass	Fail	Fail
Scan stuck-at	Pass	6	1463	7	
	Pass	14	0	34	1
	Fail	6	1	13	8
	Fail	52	36	1251	
		Pass	Fail	Pass	Fail
Functional test (52% coverage)					
		Pass	Fail	Pass	Fail
Scan - delay test					

Figure 25.5 Comparison of test methods (the SEMATECH experiments).

### 25.2.3 Test Development

The main objective of test development activity is to create tests that excite faults in the chosen fault model and make their effect visible at primary outputs or power supply

inputs (if measuring current). The output of this activity is a set of tests or test vectors consisting of input stimuli and corresponding expected fault-free responses. Test vectors are applied sequentially, one test vector at a time, at the chosen frequency. The test vector generation effort for a microprocessor may employ methods ranging from completely manual or handcrafted test development to completely automatic test vector generation.

Over the years a number of algorithm and heuristics have been developed for the automatic test pattern generation (ATPG) [1], [2]. A typical ATPG algorithm picks a fault from the fault list and searches a test vector (input combinations) to excite it and propagate its effect to an observable output. Once a test is found, it is evaluated to check for other faults it may detect. All detected faults are removed from the fault list and the process repeats until either the fault list is empty or some exit criteria is reached.

The fault model of choice for these algorithms is the classical single stuck-at fault model, although more recently, delay and  $I_{DDQ}$  fault models have also been successfully used. Since these algorithms work with a structural description such as gate level or transistor level model of device, the test vectors generated are called *structural tests*.

Automatic test pattern generation for an arbitrary sequential circuit is impractical today. Therefore, complex devices such as microprocessors must employ drastic circuit modification techniques (Design for Test) that transform it into a combinational circuit during testing and then obtain test vectors using ATPG tools. When circuit modifications are not feasible for reasons of chip area or chip performance, test development consists of creating *functional tests*.

Unlike the structural tests, the generation of functional tests is based on a higher level of abstraction, such as an RTL or ISP model, and tests are written in macro- or micro-code. The test developer takes functional units and partitions them into appropriate level of smaller components and writes tests with the objective of exhaustively exercising the functionality of each component. Tests are evaluated using fault simulation for their coverage of faults in the chosen fault model. New tests are written to remove any coverage holes.

Techniques and tools employed for functional test development overlap those for design verification. In fact most functional test vector development efforts start with evaluating the coverage of carefully selected design verification tests. The test development effort is limited to those areas inadequately covered by the design verification test set [12].

Several times we have referred to the term *fault coverage*. It is a measure of completeness of test vector set and is defined as follows:

$$\text{Fault coverage} = \frac{\text{Number of faults in fault model detected}}{\text{Total number of faults in fault model}} \times 100$$

High fault coverage is linked to higher quality of product [15], [16]. A fault simulator is used for evaluating the fault coverage of a test vector set. Figure 25.6 shows a simple fault simulator. The test vector set is simulated on two simulation machines, one representing the good machine and the other representing the faulty machine obtained by injecting an untried fault from the fault universe. A fault is declared detected if the output response at the observation points (typically the output pins) differs for at least one vector. The process is repeated until all faults in the universe have been tried.

Like ATPG, fault simulation is a compute-intensive process. For complex devices such as microprocessors that require a very large functional test vector set, the fault simulation can be time consuming, expensive, and in the critical path of releasing product to market. In order to complete test development in reasonable time, fault

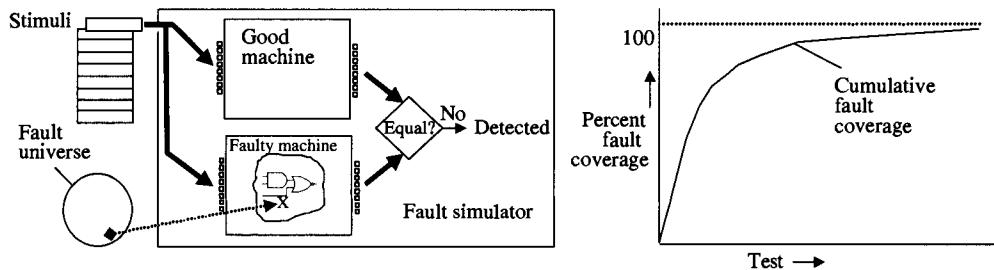


Figure 25.6 Concept of fault simulation.

simulation effort on such devices employs hardware accelerators or resorts to approximate coverage estimation techniques such as the statistical fault analysis (STAFAN) [13] or fault-sampling techniques [14].

#### 25.2.4 Testing in Production

Preparing for testing ICs in volume production involves selecting automatic test equipment (ATE or tester) and designing fixtures such as probe cards and load boards. It involves translating test vectors from the design and simulation environment into a format suitable for ATE and creating a test program to support production screening and bins. It also sets up test flows and procedures for gathering pass/fail data and statistics for troubleshooting production problems and yield improvement. This range of activity is referred to as *test engineering*.

Figure 25.7 shows a simple view of a general-purpose tester. Each signal pin of the device under test is connected to a tester channel via a suitable fixture (probe card or socket, load board). A tester channel consists of a driver, a receiver, and the associated Format and Timing Control logic. A pattern memory stores test vectors. Each test vector supplies the stimulus and the expected response data, timing, and the test program control information. When the Compare logic detects mismatches between the expected response and the received response, the Failure Capture Memory captures the failing response and the corresponding source vector on the fly. Finally, the parametric measurement unit provides the capability to measure voltages and current on pins.

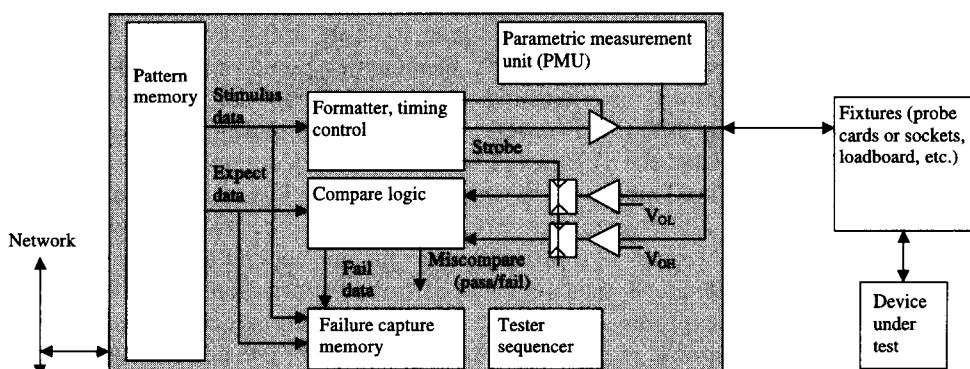


Figure 25.7 A basic tester.

The tester is the most expensive item in testing. The speed or the data rate of the tester channel, the accuracy, the flexibility of timing control, and the test vector depth contribute to the cost of tester per channel. In addition, the characteristics of the chip being tested, particularly, the presence or absence of testability features, can greatly influence the cost of the tester.

## 25.3 DESIGNING FOR TESTABILITY

*Testability is joy and pain in testing.* Researchers have tried to define testability analytically and proposed a number of metric or figure of merit for measuring testability. Since presence or absence of testability can have a significant impact on quality of product, testing costs, time-to-market and time-to-profitability, we give a qualitative measure of testability as follows:

$$\text{Testability} = f^{-1}\{\text{Customer visible quality, testing cost, time-to-market, time-to-profitability}\}$$

Stated in simpler words, testability is an attribute of design that can have significant impact on success of a product. It can positively impact customer visible quality and reduce cost of delivering a product by reducing the testing costs. It can positively impact time-to-market and time-to-profitability by aiding speedy test development and product characterization, and speedy debug and diagnosis of production and yield problems.

*Design for testability* (DFT) is a design attribute added to a product to transform the pain of testing into the joy of testing. It maximizes the above testability function by exploiting silicon for testing silicon. It systematically embeds test solutions in design that work around known limitations of test development and testing tools.

### 25.3.1 Scan Design Techniques

A scan design technique can transform an arbitrary sequential circuit into combinational circuit for the purpose of testing, thus making it possible to obtain nearly 100% fault coverage using an ATPG program.

*Full scan design* technique implements all state elements (latches and flip-flops) in a design as scannable flip-flops, referred to as *scan-flops*. A scan-flop is simply a flip-flop with two data ports. Figure 25.8 shows a generic scan-flop and a device level scan register or *scan-path*. A scan-path may load/unload the state elements directly from the device's input/output pins by following a simple shift procedure. A test generation program can, therefore, treat the state elements as pseudo-inputs and outputs of the device and concentrate on generating tests for the combinational logic left behind.

During testing, first the scan-path itself is tested by shifting a simple sequence of 1's and 0's. This is called the *shift-test*. Then the ATPG-generated test vectors are applied to test the combinational logic as shown in Fig. 25.9. The input values assigned to pseudo-inputs in a test vector are serially loaded into the state elements via the scan-path while those assigned to normal input pins are applied at the respective primary input pins. The device is then returned to normal operation mode, typically for one clock cycle. This captures the response of the combinational circuit into the scan-flops. The captured response is unloaded via the scan-path and at the same time the state element values corresponding to the next test vector are loaded in. This testing sequence is repeated until all test vectors are applied.

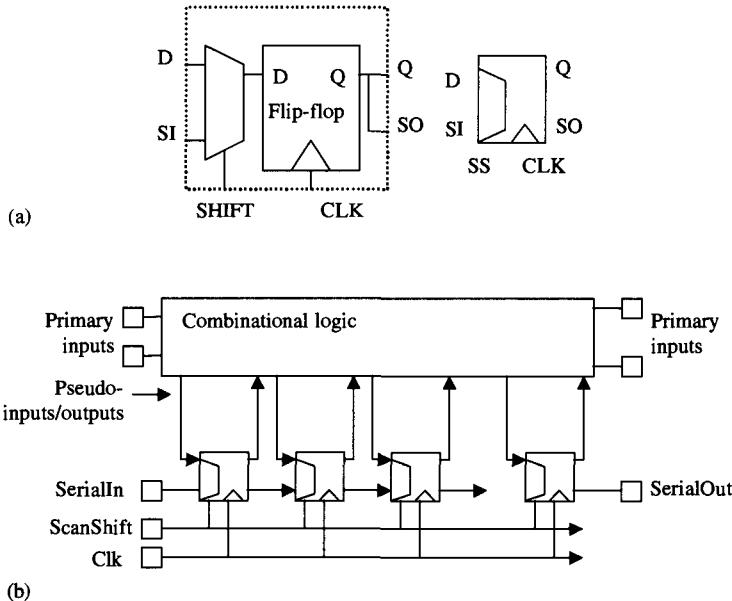


Figure 25.8 A scan-flop and a device level scan-path.

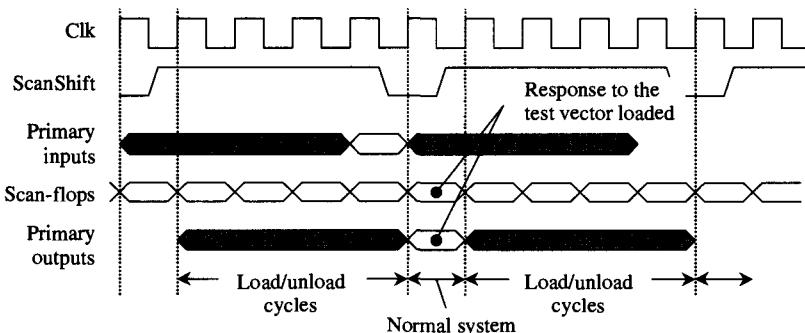


Figure 25.9 Typical scan test sequence.

### 25.3.1.1 Scan-Flop Design and Clocking

The basic scan-flop functionality can be implemented in a variety of ways. In fact, the scan-flop design and its clocking strategy are the two most important design considerations that determine the chip area and performance costs of inserting scan. The *Data-Mux* type of scan-flop shown in Fig. 25.8 introduces gate delays in the system path and therefore can impact performance. Its advantage is that both the normal operation and the scan (shift) operations use a common clock, thus requiring only one global clock to be distributed. It does require the ScanShift control signal to be routed throughout the chip, but its electrical requirements are not as stringent as distributing the additional clock that some scan-flop designs require. Furthermore, it can be piped (i.e., retimed using nonscan flip-flops) several times to ease its distribution.

The *level sensitive scan design* (LSSD) [17] introduced in 1977 implemented the dual-ported scan-port using two clocks, one for the system operation, and one for the test operation. We will refer to this as the *clock-mux* type of scan-flop. Figure 25.10 shows implementation of a scan-flop like this in the AMD-K6 [18] microprocessor. The shaded box shows the contents of the rising edge-triggered flip-flop used for the normal system operation. During scan shifting, the system clock Clk is held low and Sclk1 and Sclk2 are fed with two non-overlapping clocks. Figure 25.11(a) shows the relationship among the clocks. The flop has the advantage that there are no direct gates in the system path, and therefore the performance penalty is relatively lower than the data-mux type scan-flop. The design requires three clocks to be globally routed. However, the use of non-overlapping clocks for shift operation makes routing of the shift clock easier.

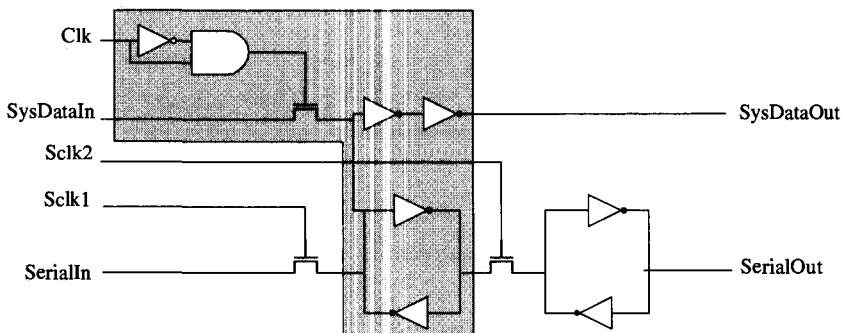
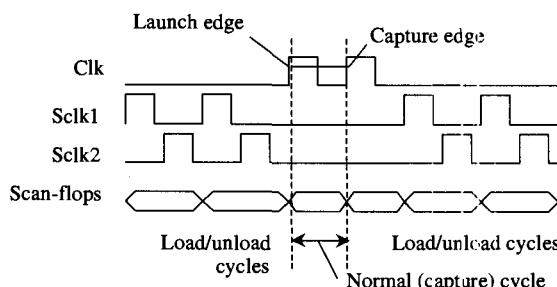
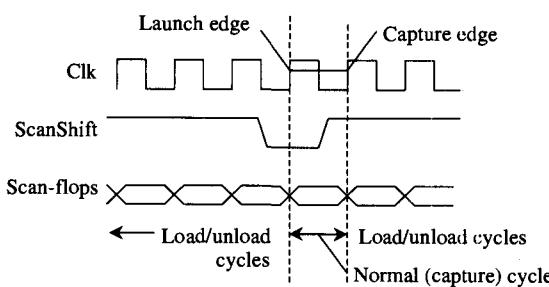


Figure 25.10 A scan-flop in an AMD-K6 microprocessor.



(a) With Clock-Mux Type Scan-flop



(b) With Data-Mux Type Scan-flop

Figure 25.11 Delay testing using scan design.

### 25.3.1.2 Scan-Based Delay Fault Testing

Although the testing sequence of load/unload and capture cycles shown in Fig. 25.9 hardly resembles the normal function of the chip or the functional testing applied at full speed, the scan technique lends itself to very effective testing for delay faults.

Let us call the clock edge on which the scan-flops capture normal/system response data the *capture edge* and the clock edge previous to the capture edge as the *launch edge*. Path delay fault testing is accomplished by controlling the distance between the capture-edge and the launch edge. For the clock-mux type scan design with two non-overlapping shift clocks, the capture-launch window is controlled by applying a burst of two capture cycles as shown in Fig. 25.11(a). The first burst serves as the initialization pattern. The second burst launches the transitions whose responses are captured and shifted out. The scan architecture with the data-mux type of scan-flop performs path delay test similarly. If the shifting is done at speed, this type of design can use the last shifted pattern as the initialization pattern as shown in Fig. 25.11(b). If the distance between the capture edge and the launch edge is equal to the clock period of the design, it accomplishes the pass/fail-testing equivalent of at-speed functional testing. Several commercial scan-based ATPG tools can obtain test patterns automatically for this type of testing.

### 25.3.1.3 Other Uses of Scan

The visibility of the internal state of design provided by the scan can be exploited for chip as well as the end-system debug. The partitioning of large circuits into smaller combinational circuits between the scan-flops makes fault isolation and failure analysis easy. A number of commercial scan design-based tools support automatic diagnosis. The ability to read and write the entire state of the machine has been exploited for diagnosing intermittent failures in field as well as for supporting system recovery on error.

### 25.3.1.4 Partial Scan Design

*Partial scan design*, as implied by the name, does not scan every flip-flop in the design. High-performance and high-volume microprocessors often resort to partial scan design when they cannot afford the performance or area penalty of full scan design. The main objective is still to add enough scan design such that an ATPG tool can generate a high coverage compact test vector set in reasonable time.

Researchers have extensively explored the partial scan design problem. The gist of their research has been to determine the best or the minimal set of flip-flops to scan given the gate-level description of an arbitrary sequential circuit. However, a simple strategy based on structural analysis is found to be the most successful and quite appropriate for microprocessors. In this strategy, flip-flops are converted into scan-flops such that the design meets the following two criteria: (1) every feedback path is broken by at least one scan-flop, and (2) the sequential depth of the acyclic network left behind is limited to a certain small number. A partial scan design meeting these criteria brings the sequential circuit in the realm of capabilities of the ATPG programs. This type of methodology was successfully used in a PowerPC microprocessor design [20].

### 25.3.2 Built-in Self-test (BiST) Techniques

Whereas the scan design techniques address the issue of automating the development of test patterns, *Built-in Self-Test* (BiST) techniques aim at eliminating it altogether. In addition, a built-in self-test can ease the requirements of tester vector depth, tester accuracy and speed, and number of tester channels. A BiST technique can make a high-performance microprocessor testable on a low-cost tester.

#### 25.3.2.1 General Model of Built-in Self-Test

An ideal self-testing circuit has its own test stimulus source and response evaluators. The only externally fed inputs are the clocks and some essential test set-up and commands. The only outputs returned are the results of the self-test, namely, the pass/fail status and perhaps some diagnostic information. Testing is done at full clock rate.

Figure 25.12 shows the general model of a self-testing circuit. The built-in stimulus source may consist of a target circuit specific algorithmic pattern generator or a more general counter or pseudo-random pattern generator. It may consist of microprogram or macrocode stored in the on-chip ROM or sometimes a special self-test macrocode loaded into the on-chip RAMs. The response evaluator may examine responses every time the circuit under test outputs a meaningful response, or it may first accumulate the responses on-chip by some compression technique and compare the final result of compression when the test ends.

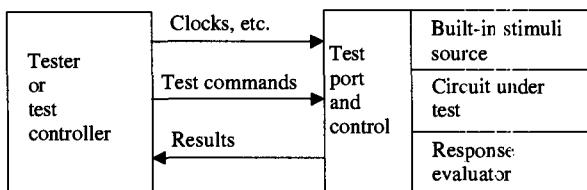


Figure 25.12 General model of self-testing circuit.

The algorithmic pattern generator and cycle-by-cycle response comparison is appropriate when the circuit under test consists of highly regular structures such as RAMs, CAMs, and register files. We will examine self-test for the RAMs in Sect. 25.3.2.6.

#### 25.3.2.2 Self-Test with Loadable Macrocode

Some may argue that a loadable code-based self-test does not constitute a built-in self-test. Regardless of how one classifies it, the technique has its place in testing microprocessors. A self-testing code loaded into the instruction cache of the microprocessor can exercise specific targets on the microprocessor, temporarily store the responses in register files and data caches, and then check them to produce the pass/fail status. Of course the approach assumes that the instruction cache and some minimum kernel are functioning properly. The test is organized such that it expands the tested kernel. Once enough of the kernel is verified to be functional, the code focuses on testing the specific test targets.

Clearly, this approach does not eliminate the test development effort. On the contrary, it requires special effort in organizing the test and making it self-checking. For microprocessors that primarily rely on functional tests for production screens, a loadable self-test pays off in testing during burn-in and life test. It can also be used for testing

high-performance microprocessors on a low capability tester. Such testers may load the instruction cache at a slower rate and then let the chip self-test itself at full speed using an on-chip phase-lock-loop. Finally, the results may be unloaded at a slower rate.

### 25.3.2.3 Self-Test with Linear Feedback Shift Registers

Apart from the special cases noted above, the most widely used self-test technique today employs built-in stimulus sources and response evaluators based on linear feedback shift registers (LFSRs).

An LFSR is a form of finite state machine composed of a shift register with the signal from its serial output fed back through exclusive-OR gates into its serial input and a few selected positions. As shown in Fig. 25.13, the cycle-by-cycle transition of states in the LFSR corresponds to the process of division by a polynomial in Galois field GF(2) [5]. After each cycle of transition, the contents of the LFSR represent the remainder, and the serial out represent the quotient of the division step [1], [21].

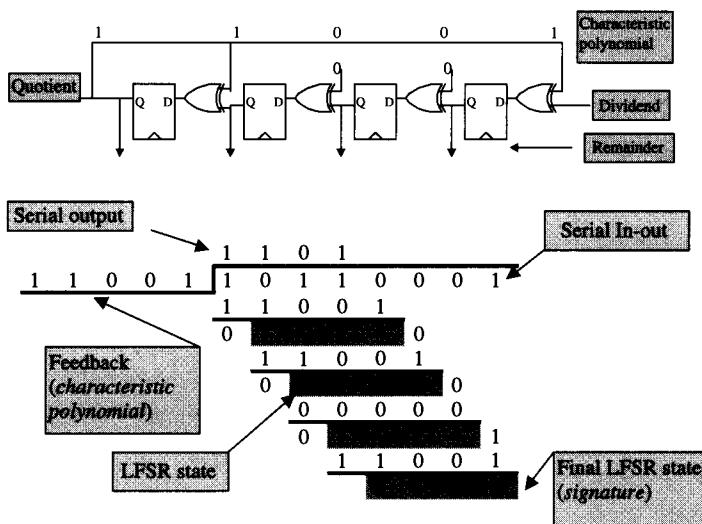


Figure 25.13 Linear feedback shift register and process of division by polynomial.

If an LFSR is initialized to a nonzero value and its serial input is tied off to logic zero, and if the feedback taps correspond to a primitive polynomial, the cycle-by-cycle transition of states goes through all possible combinations, except the all-zero state, in a random order. This is what makes it a built-in source of pseudo-random test patterns. On the other hand, if the LFSR is initialized to all zeroes (or any known state) and its serial input is tied to one of the outputs of the circuit under test, then it compresses the cycle-by-cycle responses of the circuit under test. The pass/fail decision is made by comparing the final remainder, called the *signature*, with the signature of the known good circuit under test. This use is also known as *signature analyzer*.

**25.3.2.3.1 Designing Stimulus Source and Response Evaluator.** Designing a built-in self-test scheme based on LFSRs involves determination of the size, the feedback connections, and how LFSRs will be integrated into the circuit under test. Generally, the

size of LFSR for the stimulus source is determined by the greater of the two numbers: (1) the estimate of the number of test patterns needed, and (2) the number of circuit inputs to be fed. An LFSR with  $r$  stages supplies up to  $2^r - 1$   $r$ -bit wide pseudo-random vectors.

The feedback is generally based on the primitive polynomial [5]. Generally for a given size of LFSR, there are several choices of primitive polynomials. Experience has shown that it does not matter which one is selected. So to minimize the hardware, a polynomial with the fewest taps is typically selected. The feedback itself can be applied by any of the two methods shown in Fig. 25.14. The relationship between the two forms is established by linear transformation [22]. What is important is that both feedback forms yield identical results from a test perspective. The choice, therefore, may be based on circuit considerations. For example, the internal form may be preferred for a large LFSR with many feedback taps to distribute the feedback delay. On the other hand, the external form offers a highly regular design with all stages identical. This form may be preferred for LFSRs located in data path. The feedback circuitry can be built outside the data path, as a part of the control logic.

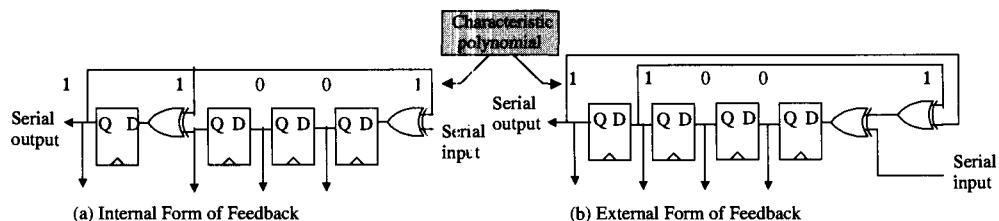


Figure 25.14 Two forms of feedback: (a) internal feedback and (b) external feedback.

The considerations in designing a signature analyzer are similar. The size, that is, the number of bits in the LFSR, is kept equal to or greater than the number of outputs of the circuit under test. All outputs are compressed simultaneously by feeding them in parallel as though exclusive-OR gates shown in Fig. 25.15. This method is referred to as *Multiple Input Signature Register* (MISR) [26].

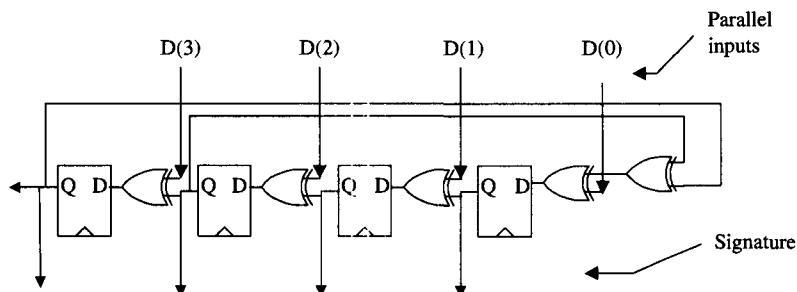


Figure 25.15 LFSR used as multiple input signature register (MISR).

Another important consideration in selecting the size of the LFSR for response compression is the problem of *aliasing*. Because of compression, several erroneous response streams may produce the same signature as the error-free response stream. Faulty circuits producing such errors cannot be distinguished from fault-free circuits.

The problem of aliasing is extensively covered in the literature [1], [3], [23], [24], [25]. If we assume that the fault-induced errors are uniformly distributed, the probability of aliasing for a long test is approximately  $1/2^t$ . For a large LFSR, as typically found in real applications, the aliasing probability is low and its impact on fault escape is marginal.

The final consideration in designing the LFSRs is integrating them with the circuit under test. This involves determining the operations that LFSRs must support. Clearly, the LFSR for stimulus source must support pseudo-random pattern generation. In addition, it must support initialization or loading of the seed value. Likewise, the LFSR for response compression must be capable of initialization and be able to off-load the signature. Since an LFSR is built from a shift register, it is natural to add a shift register mode and rely on it for both initialization and off-loading signatures.

**25.3.2.3.2 Placing LFSRs in Large Circuit under Test.** No one applies BiST with pattern source and response evaluators just at the primary input/output pins of a microprocessor. Such a test scheme will be ineffective, yielding poor coverage and have an unacceptable number of test vectors and long test time. In order to make an effective self-test for a large circuit under test such as a microprocessor, LFSRs, whether used as a stimulus source or signature analyzer, must be scattered judiciously inside the microprocessor. Appropriate locations are the natural sources and sinks of data such as buses, as well as source and destination registers in data paths. They may also be wrapped around specific blocks of logic such as PLAs and ROMs.

Another very effective location for LFSRs is at the serial input and output of scan-paths. This combination of two powerful design for test technique is known as Scan-BiST or Logic-BiST. We discuss Scan-BiST in the next section.

#### 25.3.2.4 Scan-BiST

Originally proposed for testing multichip modules consisting of several ICs with LSSD [27], the technique has helped to make arbitrarily large sequential circuit self-testable. Figure 25.16 shows a typical Scan-BiST architecture. The scan-path inside the chip is split into several parallel scan-paths of roughly equal length. The source LFSR serially pumps the pseudo-random vectors into each scan-path during the load/unload cycles. Simultaneously, the sink LFSR compresses the response data streaming out at serial outputs of scan-paths. The on-chip BiST controller generates the clocks and control needed for repeatedly stepping the load/unload and capture cycle sequences

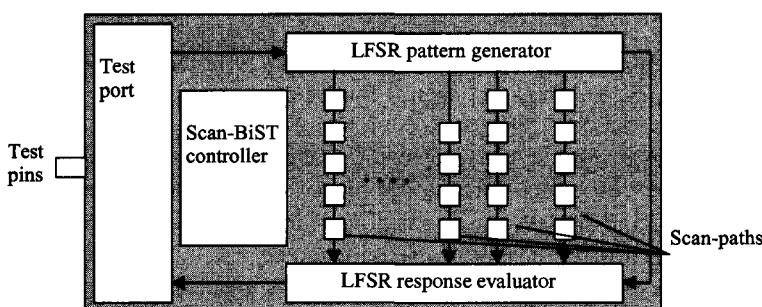
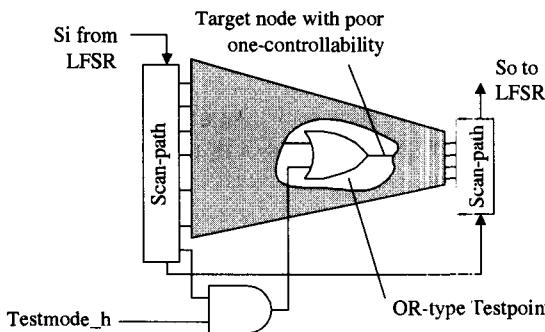


Figure 25.16 Scan-BiST architecture.

shown in Fig. 25.9. The final signature is shifted out or compared on-chip to complete the self-test.

In order to contain high fault coverage in Scan-BiST, the circuit under test must be analyzed for the presence of random-pattern resistant faults and treated to remove them. A *random-pattern resistant fault* is simply a fault known to be difficult to detect by random patterns. Consider a 16-input AND gate. To detect stuck-at zero fault on the output of this gate all inputs to the AND gate must be set to ones. The probability of obtaining this pattern from a 16-bit pseudo-random pattern generator is  $1/2^{16}$ . No one builds a single gate with such a high fan-in but any general circuit cluster with high fan-in has the same problem.

Random-pattern resistance faults are removed by inserting test points. A test point may be inserted in a variety of ways. For example, to increase the probability of controlling a node to logic one, a test point using an OR gate may be added as shown in Fig. 25.17. Similarly, a test point with an AND gate can be used to enhance zero-controllability at a given point. Several algorithms and their implementation in commercial tools automatically insert test points.



**Figure 25.17** Random-pattern resistance fault removal with a test point.

Scan-BiST test can be run at full speed to obtain coverage of delay faults. This, however, requires that the circuit under test not have any state-dependent false paths or multicycle paths. Such paths, although fine in normal operation, may fail Scan-BiST. Similar to test point insertion, additional design for testing may be used to remove such paths. Scan-BiST has been effectively deployed in several ASICs and some leading-edge microprocessors [20], [28].

### 25.3.2.5 Observability LFSRs

It is common for microprocessors to employ a hybrid design for test strategy, combining functional test and BiST techniques. Such a strategy may use a functional test developed as discussed in Sect. 25.2.3 for the stimulus source and employ judiciously scattered *observability registers* (signature analyzers) on strategic internal nodes to capture responses.

The observability registers may be placed on outputs of critical functional units, on global signals, and signals crossing major logic blocks. Several thousand internal nodes may be captured without impairing performance. The observability registers detect and trap the faults excited by the functional tests focussed on thoroughly exercising various parts of the circuit under test. Several microprocessors have reported use of this technique [28] [30] [31]. An observability register architecture is described in detail in [32].

### 25.3.2.6 Embedded RAM Self-Test

A modern microprocessor invariably employs large embedded RAMs in a variety of ways to enhance its performance. The most common usage is in the form of on-chip caches. In many instances, such RAMs constitute the majority of transistors consumed on-chip and are the largest occupants of chip real estate. For example, on the Alpha 21264 [31], the cache RAMs consume approximately two thirds of the chip's 15 million transistors and take up approximately one third of the chip area.

Embedded RAMs give rise to two interesting problems during chip manufacturing. First, because of their sheer size they become the dominant yield drivers for the host chip. Second, the lack of direct access to their input/output signals not only makes their own test difficult, but also impairs testability of the other functions on the chip.

To address the yield problem, large RAMs are provided with redundant rows and/or columns. To tackle the testing problems, RAMs are provided with built-in self-test. A number of sophisticated algorithms exist for testing RAMs [4]. The most popular among them, the MARCH is easily implemented by simple test hardware. The algorithm covers cell stuck-at, coupling, and address decoder faults. The algorithm can be expressed as follows:

```
For (address = 0 to address = last) {Write data ; } // Initialize
For AddressSweep = forward to AddressSweep = reverse) {
    For (data = 0 to data = 1){
        For (address = first to address = last)
            {Read data; Write ~ data; } // Read and compare. Write complement
    }
}
```

Figure 25.18 shows a simple implementation of the above algorithm using an ordinary counter as the algorithm engine. The counter bits are organized into several fields, which supply address, data, and read-write control. In this scheme, the complemented address field produces the reverse address sweep needed in the MARCH test.

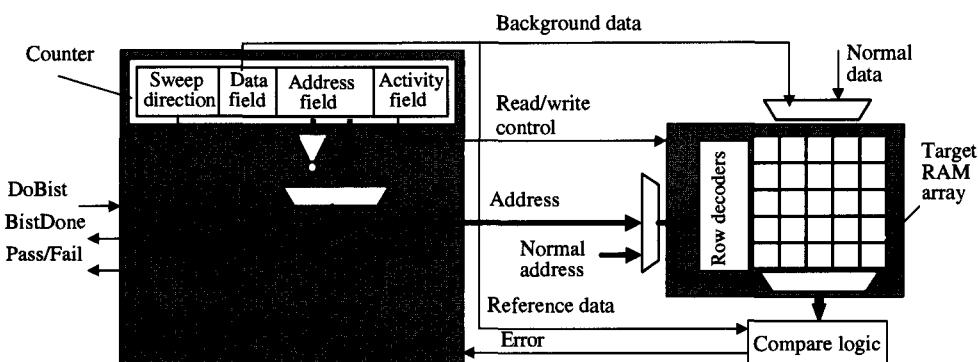


Figure 25.18 RAM self-test using MARCH algorithm.

The scheme shown above can be enhanced and tuned to cover different RAM fault models. The sequence of activities (reads and writes) performed at each address can be expanded by adding bits to the activity field. Likewise, MARCH elements with additional background data can be added by appending bits to the data field.

One disadvantage of a sequencer-based RAM BiST scheme such as the one shown above is that the RAM can be tested only with the algorithm(s) implemented by the algorithm engine. Product and test engineers find this too restrictive and fear that they may not be able to adequately test and diagnose unanticipated failure mechanisms. This concern can be addressed by implementing the BiST engine with a special processor that can be programmed to run any test algorithm [33],[34].

Built-in Self Test on RAMs opens up an opportunity for Built-in Self-Repair (BiSR). A large RAM array is provided with either redundant rows or columns or both. The general BiST-BiSR scheme works as follows. As BiST engine tests and detects bad cells, the row and column coordinates of the failing cells are stored on-chip in special memory or registers. The on-chip processor or the BiSR logic analyses the incoming failing address and maintains an up-to-date failure bitmap information. At the end of the BiST, the final failure bitmap and the associated logic allocate spares to replace bad rows and columns with spares. Among the microprocessors, the Alpha 21164 uses self-repair by spare rows [29] and the Alpha 21264 by spare rows and columns [35].

### **25.3.3 IEEE 1149.1 Standard Test Access Port and Boundary Scan Register**

IEEE Std. 1149.1 has become an indispensable part of modern microprocessor. All microprocessors implement its mandatory provisions that support testing board level interconnections. Many microprocessors rely on the standard's test access port for controlling the chip-level testability features and use the boundary scan register for performing a number of chip-level test functions. In this section we provide a brief overview of the standard and show its applications for the chip-level testing. A designer wishing to implement test access port and boundary scan register compliant with the standard must refer to the standard [36].

#### ***25.3.3.1 Overview of the Architecture***

Figure 25.19 shows the major provisions of the standard divided into two parts: the Test Access Port (TAP) and the Testability Features. The latter consists of the mandatory boundary scan register located at the chip pins and other optional testability features and test data registers which may support the chip-level test functions.

The TAP interface consists of the serial data Tdi\_H, and Tdo\_H, the test clock and mode select inputs Tck\_H and Tms\_H, and the optional test reset Trst\_L. The Instruction Register holds a test instruction that selects a test data register for serial access and controls chip's testability features. The bypass register is a one-bit shift register, providing a short shift path through TAP. It is generally useful at the board level testing. The TAP Controller is the state machine controlling the access to the instruction register and the testability features. Figure 25.19 shows the state diagram. It has two identical scan sequences, namely, Scan-IR and Scan-DR, controlling the operation of the Instruction Register and the test data register, respectively. The shift states shift the data serially. The capture and update states load and apply the test data

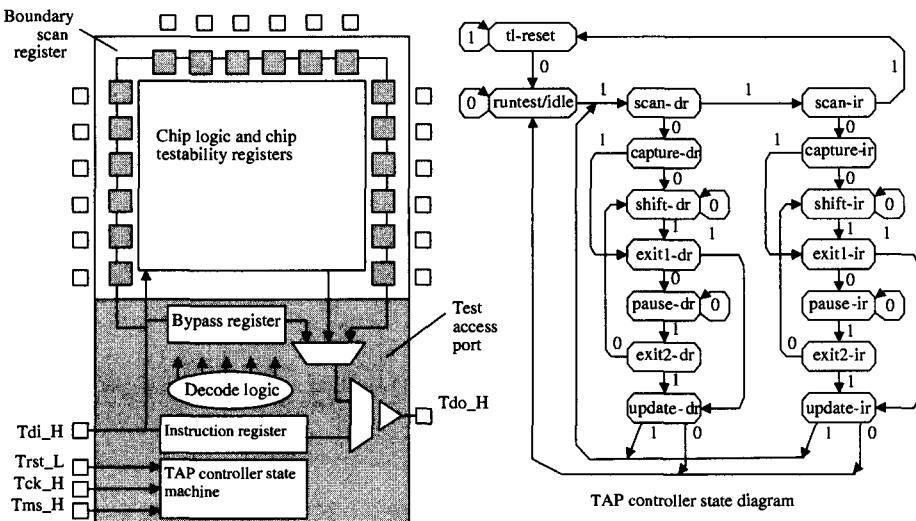


Figure 25.19 Major provisions of the IEEE Standard 1149.1.

in parallel. A newly shifted instruction becomes current in the Update-IR state. The exit and the pause states halt the shift operation. Some testers may use this to replenish data for long serial data streams. In Run/Idle state, the test logic is either idle or running the test initiated by the current instruction.

The boundary scan register (BSR) consists of boundary scan register cells located at all system pins of an IC. At minimum, the boundary scan register must support an interconnection test that is compliant with the standard.

Figure 25.20 shows a typical design of boundary scan register cells for an input-only, output-only and bidirectional pins. The cell structure consists of a shift register stage and a parallel output stage. The output stage feeds data to the pins via an intrusion multiplexer. The bidirectional pin has a control cell associated with it to control the pin direction during pin-intrusive test modes.

The cells shown in Fig. 25.20 support SAMPLE/PRELOAD and EXTEST instructions. With SAMPLE/PRELOAD instruction, all BSR cells shift in the Shift-DR state, capture system data on pins in the Capture-DR state, and parallel load the output stage in the Update-DR state. The intrusion multiplexer still passes the system data. The instruction is used to preload the boundary scan register. The instruction can also be exploited for providing a visibility for system debug, similar to a logic analyzer, without using physical probes [37].

The behavior of the BSR during the EXTEST instruction is similar to the SAMPLE/PRELOAD, except that the intrusion multiplexer is switched to pass data from the BSR cells. On the bidirectional pin, the control cell enables or disables the output driver. Board manufacturing uses this instruction to test interconnections, and the chip manufacturing uses it for DC parametric testing.

### 25.3.3.2 Applications for Chip-Level Testing

The most important application of the standard for chip-level testing is the use of TAP for accessing and controlling the chip-level testability features in a variety of ways.

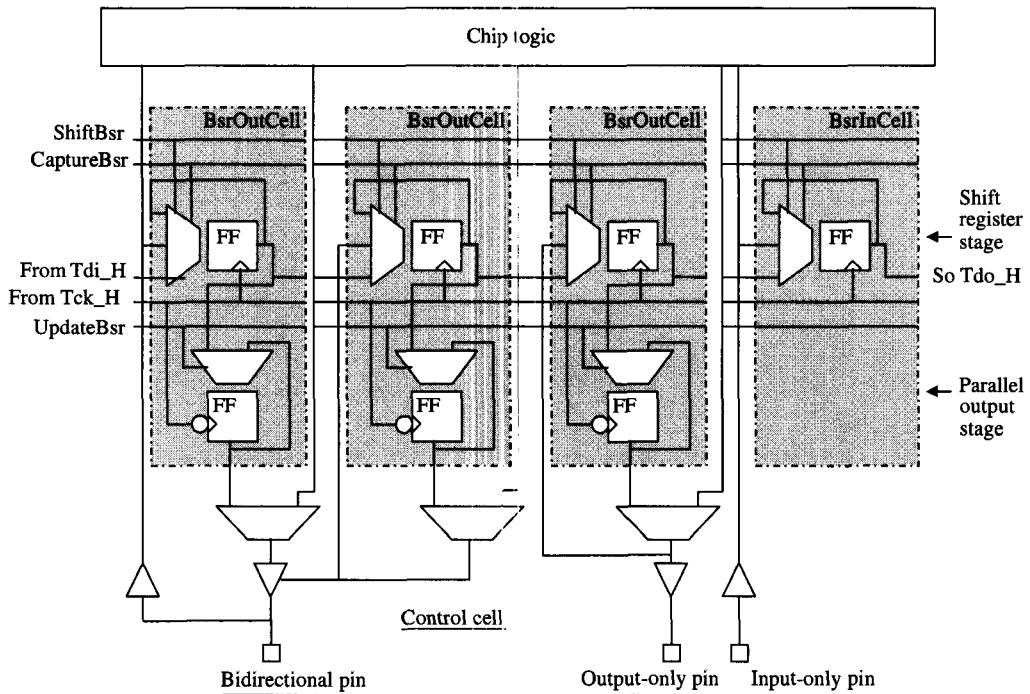


Figure 25.20 Boundary scan register cells at common pins.

The standard defines the optional RUNBIST instruction. With this instruction loaded, Shift-DR state can serially set up (initialize) various LFSR-based stimulus sources and response evaluators, the Run/Idle state may run the self-test, and when the self-test is finished, the Shift-DR state may serially shift out the signatures. The standard also defines an optional INTEST instruction that when loaded can apply the boundary scan register data to the chip's internal logic. The boundary scan operation can be integrated with the internal scan registers to test out chip logic without contacting chip's I/O pins. This type of approach is often used for testing a high-pin count chip on a tester with a limited number of tester channels [28], [20].

Many microprocessors access internal scan paths using the IEEE 1149.1 test access port [38], [39]. The Alpha 21264 microprocessor first switches the TAP to a synchronous mode to run it with the chip clock and then uses it to access all chip testability features [40], [31].

## 25.4 CONCLUSION

In this chapter we reviewed several basic concepts and techniques and tools of the testing technology and surveyed their applications to a number of microprocessors. Just as there are different paths to Nirvana, microprocessors as a group of ICs follow a wide spectrum of test strategies. At one end of the spectrum are the microprocessors [18], [19], [20], [28], [38], [39] that rely heavily on structured design for test techniques such as the full scan design and Scan-BiST. At the other end are the microprocessors

such as [29], [30], [31], which rely heavily on functional tests and a custom design for testability strategy.

In the beginning of the chapter we had asked the hypothetical question: How will we test an IC with 100 million transistors and more? We believe that this will be done by a pervasive use of design for testability. We anticipate BiST of large arrays, scan design and Scan-BiST of general logic areas, and new design for test features for the interface testing to be among the future solutions. And of course, test and design for test are areas that are ripe for further innovations.

## REFERENCES

### Books

- [1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. Revised printing, IEEE Press, Piscataway, NJ, 1991.
- [2] V. D. Agrawal and S. C. Seth, *Test Generation for VLSI Chips*. IEEE Computer Society Press, Los Alamitos, CA, 1987.
- [3] P. H. Bardell, W. H. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques*. Wiley Interscience, New York, 1987.
- [4] A. J. van de Goor, *Testing Semiconductor Memories—Theory and Practice*, John Wiley & Sons, New York, 1991.
- [5] W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes*, 2nd edition. MIT Press, Cambridge, MA, 1997. First printing 1972.

### Articles

- [6] J. M. Soden, C. F. Hawkins, R. K. Gulati, and W. Mao, “ $I_{DDQ}$  Testing: A Review,” *J. Electronic Testing: Theory and Applications (JETTA)*, vol. 3, pp. 291–303, Nov. 1992.
- [7] S. M. Thatte and J. A. Abraham, “Test Generation for Microprocessors,” *IEEE Trans. on Computers*, vol. C-29, no. 6, pp. 429–441, June 1980.
- [8] M. S. Abadir and H. K. Reghbati, “Functional Testing of Semiconductor Random Access Memories,” *ACM Computing Surveys*, vol. 15, no. 3, pp. 175–198, September 1983.
- [9] P. M. Maxwell, R. C. Aitken, V. Johansen, and I. Chiang, “The Effectiveness of  $I_{DDQ}$ , Functional and Scan Tests: How many Fault Coverages Do We Need?” *Int'l Test Conf.*, pp. 168–177, Sept. 1992.
- [10] S. C. Ma, P. Franco, and E. J. McCluskey, “An Experimental Chip to Evaluate Test Techniques: Experiment Results,” *Int'l Test Conf.*, pp. 663–672, Oct. 1995.
- [11] P. Nigh, W. Needham, K. Butler, P. Maxwell, and R. Aitken, “An Experimental Study Comparing the Effectiveness of Functional, Scan, IDDq, and Delay fault Testing,” *VLSI Test Symposium*, pp. 459–464, April 1997.
- [12] C. Stolicny, R. Davis, P. McKernan, and T. Truong, “Manufacturing Pattern Development for the ALPHA 21164 Microprocessor,” *Int'l Test Conf.*, pp. 278–285, Nov. 1997.
- [13] S. K. Jain, and V. D. Agrawal, “Statistical Fault Analysis,” *IEEE Design and Test of Computers*, vol. 2, no. 1, pp. 38–44, Feb. 1985.
- [14] V. D. Agrawal, “Sampling Techniques for Determining Fault Coverage in LSI Circuits,” *Journal of Digital Systems*, vol. 5, no. 3, pp. 189–202, 1981.
- [15] T. W. Williams and N. C. Brown, “Defect Level as a Function of Fault Coverage,” *IEEE Trans. on Computers*, vol. C-30, no. 12, pp. 987–988, Dec. 1981.

- [16] S. C. Seth and V. D. Agrawal, "Characterizing the LSI Yield Equation from Wafer Test Data," *IEEE Trans. on CAD*, vol. CAD-3, pp. 123–126, April 1984.
- [17] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *Proceedings of the 14th Design Automation Conf.*, New Orleans, pp. 462–468, 1977.
- [18] R. S. Fetherston, I. P. Shaik, and S. C. Ma, "Testability Features of AMD-K6 Microprocessor," *Proc. Int'l Test Conference*, pp. 408–413, Nov. 1997.
- [19] M. E. Levitt, S. Nori, S. Narayanan, G. P. Grewal, and L. Youngs, "Testability, Debuggability, and Manufacturability Features of the UltraSPARC-I Microprocessor," *Int'l Test Conf.*, pp. 157–166, Oct. 1995.
- [20] L. L. Day, P. A. Ganfield, D. M. Ruckert, and F. J. Ziegler, "Test Methodology for a Microprocessor with Partial Scan," *Int'l Test Conf.*, pp. 708–716, Oct. 1998.
- [21] D. K. Bhavsar and R. W. Heckelman, "Self-testing by Polynomial Division," *Int'l. Test Conf.*, pp. 208–216, 1981.
- [22] P. C. Maxwell, "Comparative Analysis of Alternative Implementations of Multiple Input Signature Analyzers," *IEEE Trans.*, C-37, pp. 1411–1414, Nov. 1988.
- [23] D. K. Bhavsar and B. Krishnamurthy, "Can We Eliminate Fault Escape in Self-testing by Polynomial Division (Signature Analysis)?," *Int'l Test Conf.*, pp. 134–139, Oct. 1984.
- [24] S. Z. Hassan and E. J. McCluskey, "Increased Fault Coverage through Multiple Signatures," *14th Annual Int'l Fault-Tolerant Computing Symp.*, pp. 354–359, June 1984.
- [25] T. W. Williams, W. Daehn, M. Gruetzer, and W. Starke, "Aliasing Errors in Signature Analysis Registers," *IEEE Design and Test of Computers*, vol. 4, pp. 39–45, April 1987.
- [26] B. Konemann, J. Mucha, and G. Zwiehoff, "Built-In Logic Block Observation Technique," *Digest of Papers 1979 Test Conf.*, pp. 37–41, Oct. 1979.
- [27] P. H. Bardell and W. H. McAnney, "Self-Testing of Multi-chip Logic Modules," *Int'l. Test Conf.*, pp. 200–204, Nov. 1982.
- [28] M. Kusko, B. Robbins, T. Snethen, P. Song, W. Huott and T. Foote, "Microprocessor Test and Test Tool Methodology for the 500 MHz IBM S/390 G5 Chip," *Int'l. Test Conf.*, pp. 717–726, Nov. 1998.
- [29] D. K. Bhavsar and J. H. Edmondson, "Testability Strategy of the Alpha AXP 21164 Microprocessor," *Int'l. Test Conf.*, pp. 50–59, Oct. 1994.
- [30] A. Carbine and D. Feltham, "Pentium-Pro Microprocessor Design for Test and Debug," *Int'l Test Conf.*, Nov. 1997.
- [31] D. K. Bhavsar, et al., "Testability Access of the High Speed Test Features in the Alpha 21264 Microprocessor," *Int'l. Test Conf.*, pp. 487–495, Oct. 1998.
- [32] D. K. Bhavsar and R. Tan, "An Observability Register Architecture for Cost-effective Production Test and Debug," *North Atlantic Test Workshop*, May 1999.
- [33] H. Koike, et al., "A 30 ns 64 Mb DRAM with Built-in Self-Test and Repair Function," *Int'l Solid State Circuits Conf.*, pp. 150–151, Feb. 1992.
- [34] J. Dreibeldis, J. Barth, H. Kalter, and R. Kho, "Processor-Based Built-in Self-Test for Embedded DRAM," *IEEE J. of Solid State Circuits*, vol. 33, no. 11, pp. 1731–1740, Nov. 1998.
- [35] D. K. Bhavsar, "An Algorithm for Row-Column Self-Repair of RAMs and Its Implementation in the Alpha 21264," *Int'l. Test Conf.*, pp. 311–318, Sep. 1999.
- [36] IEEE Std. 1149.1-1993 "A Test Access Port and Boundary Scan Architecture."
- [37] M. Lefebvre, "Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester," *Int'l. Test Conf.*, pp. 294–303, Sept. 1990.
- [38] D. D. Josephson, D. J. Dixon, and B. J. Arnold, "Test Features of the HP PA7100LC Processor," *Int'l. Test Conf.*, pp. 764–772, 1993.
- [39] R. Patel and K. Yarlagadda, "Testability Features of the SuperSPARC Microprocessor," *Int'l Test Conf.*, pp. 773–781, 1993.
- [40] D. K. Bhavsar, "A Method for Synchronizing IEEE 1149.1 Test Access Port for Chip Level Testability Access," *Eleventh Int'l Conf. on VLSI Design*, Jan. 1998.

# INDEX

- absolute distance check 214  
active regenerators 374  
active skew management 273  
adaptive body biasing (ABB) 58–61  
  effectiveness 60  
adder leakage current reduction 57  
addition 181–93  
  *see also* specific adders  
address decoders 299  
address path design 299–301  
admittance moments 346  
advanced logic styles 9  
aggression sources 176  
aggressor signals 151–2  
aliasing 536  
Alpha 21064 microprocessor 18, 193, 338  
Alpha 21164 microprocessor 9–11, 15–17, 19,  
  128, 222, 540  
Alpha 21264 microprocessor 11–12, 15–17,  
  286–90, 297  
Alpha 21364 microprocessor 22  
alpha chip design 333  
alpha particle perturbations 214  
alpha particles 307–8  
alpha ratio 122  
aluminum and aluminum-alloy films 431, 432  
aluminum-based interconnects 430–1  
AMD-K6 microprocessor 532  
AMULET3 176  
analog-to-digital converter (ADC) 419,  
  421–3  
anneal process 103  
aperture window 211  
application program interface (API) 475  
application-specific integrated  
  circuits (ASICs) 134  
  design 474  
  implementation methodology 469  
arithmetic units 181–204  
Arrhenius temperature dependence 435  
aspect ratio 352  
assertion edge 211  
associativity in caches 13–14
- Asymptotic Waveform Evaluation (AWE)  
  338–9, 342–4  
  delay and noise analysis 344–5  
asynchronous logic 131  
  timing verification 492  
atomic flux 434  
  divergence 429  
attenuation, transmission lines 400–1  
automatic peak current vector generation 513  
automatic test equipment (ATE) 524, 529  
automatic test pattern generation (ATPG) 528  
  software 177  
  tools 478
- back bias generator 316–17  
  timing diagram 317  
back-gate capacitive coupling 152–3, 156  
backward Euler (BE) technique 512  
bandwidth limitation 418  
bandwidth multiplexing 409  
beta ratio 122, 156  
bias circuit 70  
biased transistors 385  
binary tree 269, 271  
bit error rate (BER) 405  
bit line precharging 301–2, 318  
bit rate maximizing 409–10  
bit rates, future trends 418  
BJT gain 90, 91  
BJT leakage 91  
Black electromigration lifetime acceleration  
  model 444  
Blech critical threshold 444  
body effect 60, 122–3, 125, 135  
Booth Encoder 200–2  
Booth-MacSorley algorithm 198  
Booth Recoding Algorithm 198–9  
Booth Selector 200–2  
boundary scan register (BSR) 541  
branching effort 137  
bubble-limited region 170–3  
bubbles 164, 168–9, 171  
buffer chain 269

- buffer insertion 333–4, 372–3  
 Built-in Self Test (BiST) techniques 534  
 Built-in Self-Repair (BiSR) 540  
 bump-array ceramic package 517–18  
 bus interface unit (BIU) clock 486
- C4 bump packages 277  
 caches 285–308  
   associativity in 13–14  
   basic architecture 286–90  
   redundancy 306–7  
   reliability 307–8  
 capacitance  
   and delay variations 360–1  
   power dissipation 360  
   problems and solutions 359–64  
 capacitance effects, scaling 353  
 capacitance models, power grids 515–17  
 capacitance variation, potential solutions 361–4  
 capacitive charge sharing 129  
 capacitive coupling 150–3, 279, 461  
 capacitor  
   admittances 341  
   divider circuit 278  
   voltages 341  
 carrier heating process 449  
 carry-lookahead adder (CLA) 185–90  
 carry select adder (CSLA) 192  
 carry skip adder (SKA) 183–4  
 cascading dynamic structures 144–5  
 cascode non-threshold logic 125  
 cascoding 383–4  
 C-elements 163–5  
 CHANGO 442  
 channel connected region (CCR) 483  
 channel-length variation 276  
 channel profile design 36–8  
 charge leakage 146–8, 156  
 charge pumps 238, 239  
 charge redistribution effect 459  
 charge sharing 148–50, 156  
 charged device model (CDM) 387, 394–5  
 chemical mechanical planarization (CMP) 100  
 chemical mechanical polishing (CMP) 322  
 chip assembly 476  
 chip-enable-bar 66  
 chip interface 235  
 chip-level circuit analysis 477–8  
 chip-level layout verification 476–7  
 chip-level testing 541–2  
 chip-level timing analysis 477–8  
 Cholesky factorization 512
- circuit design guidelines 474  
 circuit feasibility study 469  
   tools 470–1  
 circuit reliability and hot carrier reliability 460–1  
 circuit structures 489  
 classical single stuck-at fault model 525  
 clock/clocking  
   definitions 262–5  
   gated 268, 272  
   global 208–9 272  
   overview 207  
   self-timed pipelines 162–3  
   strategy 207–9  
   systems 484  
   two-wire non-overlapping 220  
 clock  
   cycles 235, 513–14  
   delay adjustment 133  
   domains 229  
   driver cell 274  
   driver pair, circuit diagram 277  
   duty cycle 485  
   edge 262–3  
   frequency 266  
   gating 486–7  
   generation 414–18  
   jitter 209, 262, 485  
   clock-mux type scan-flop 532  
   network 208  
   nonideality 262  
   objectives 265–6  
     evaluation 266–7  
   performance 263  
   skew 208, 210, 218, 485, 486  
   tick 229  
 clock design 261, 267  
   verification 276  
 clock distribution 109, 111, 261–81  
   delay 235  
   ideal 266  
   implementation 268–73  
   networks 269  
   power supply 277–8  
   process variations 276  
   research examples 273  
   temperature variation 278–9  
   trends 273  
 clock driver  
   configuration 268  
   final-stage 268  
   layout 273–6  
   predriver network 269–71

- clock signals 208, 235, 262  
nonideal nature 209–10  
periodicity 209
- clock uncertainty 212  
specifications 485
- clocked-delayed domino 132–3
- clocked diffamp circuits 386
- clocked feedback topology 228
- clocked logic 128–31
- clocked storage elements 207–34  
applications 221–7  
characterization 229  
comparison of types 231  
dynamic logic 231–3  
functional latency 230  
performance metrics 229–30  
rules for robust design 228–9
- CMOS**  
channel doping profiles 37  
leakage fault models 526  
scaling theory 17  
technology  
implementing processor architecture 6–14  
suitability 3–4
- CMOS technology**, *see also* variable threshold-voltage CMOS (VTCMOS)
- column address buffer 310
- column decoder 310
- column multiplexing 302, 305–6
- combinatorial gates 229
- comparator 7
- complementary pass-gate logic (CPL) 126–7
- computer-aided design (CAD)  
reliability analysis 442–5  
tools 261, 439–42, 469–79, 489
- computer architecture, impact of physical technology 3–24
- conditional clocking 268
- conditional-sum addition (CNSA) 192
- conjugate gradients 512
- constant-field scaling 27–9  
rules 27–9
- constant gate delay model 185
- contact and via resistance/size 104
- contamination delay 229
- content addressable memories (CAMs) 13
- control circuitry 310
- control logic design tools 475–6
- cosmic ray particle perturbations 214
- cosmic rays 307–8
- cost/performance trade-offs 401
- Coulomb's Law 359
- coupling capacitance 331, 334, 337
- coupling of unrelated signals 215
- critical loops 15
- critical path simulation input extraction 474–5
- critical paths of CPUs 87
- crossover current 123–4, 265
- crossover power 129
- crosstalk 357, 361, 368, 399
- crowbar current 123–4, 129
- current-control feedback loop 407
- current density 437, 438, 502
- current-mode drivers 406–7
- current sensing 374
- current sources 440
- cycle time, high-performance microprocessor 14–15
- cycle-to-cycle jitter 262–3
- Dadda counter 195–6
- Dadda multiplier 196
- damping factor 248, 378
- data-dependent noise 279
- data-limited region 170–3
- Data-Mux type scan-flop 531
- data-retention modes 78
- data-to-clock delay 212
- datapath/memory design 470, 473  
tools 473–5
- datapaths 12
- decoupling capacitance 499, 502, 508  
structures 517
- decoupling capacitors 276, 277, 512
- defect based testing 525
- delay 137  
analysis, Asymptotic Waveform Evaluation (AWE) 344–5  
and noise 334–5  
computation 349  
fault models 525–6  
margin 176, 177  
noise 120  
stages 135  
variability 121–2  
variations 89–90  
capacitance 360–1  
inductance 367–8
- delay-locked loops (DLLs) 71, 74, 235–60,  
273, 414–16  
alternative structures 240  
architectures 249  
block diagram 237  
circuits 255–9

- delay-locked loops (DLLs) (*cont.*)
  - design strategy 240
  - frequency response 239–40
  - general structure 237
  - locking waveforms 238
  - loop components 237–8
  - output jitter 250–3
  - performance issues 249–54
  - self-biased techniques 259–60
  - substrate noise sensitivity 254–5
  - supply noise filters 254
  - supply noise sensitivity 253
  - supply/substrate noise response 252
  - waveforms of signals and quantities 238
- demultiplexing 413
  - timing margins 413
- deposition process 103
- design for testability (DFT) 530
- design metrics 137–8
- de-skewing element 229
- deterministic interconnect variation impact 112–14
- deterministic jitter 262
- deterministic simulators 177
- device decoupling capacitance 515–17
- die-pattern dependencies 101
- die-to-die variation 59, 71
- dielectric constant 104, 432
- dielectric thickness 104
- differential cascode voltage-switched logic (DCVSL) 9, 124–5, 136
- differential logic inputs 124
- differential signaling 411
- differential split level logic 125
- digital-to-analog converter (DAC) 420–3
- Direct RAMBUS 22
- domino circuits 57–8
- domino logic 8–9, 128–30, 142–4
- doping variations 102
- drain-induced barrier lowering (DIBL) 31, 46, 47, 49, 59–60, 82
- DRAM 12, 22, 125
  - architecture 309–10
  - core and logic interference 325
  - core operation 312–14
  - macro architecture 323–5
  - see also* embedded DRAM
- drift current 29
- driver models 346–50
  - N-port 349–50
  - RC loads 347–9
- dual-cascode voltage switch logic 161
- dual-loop architectures 416–17
- dual-monotonic pair 161
- dual-monotonic signal 174
- dual-rail dynamic logic 144
- dual-rail monotonic signals 231
- duty cycle 263, 267
  - definition 262
- dynamic circuits 134–5
- dynamic dominos 128–30
  - implementation 135
- dynamic gates 140
- dynamic logic 6–9, 140–57
  - area 145
  - basics 140–1
  - comparison to standard complementary CMOS logic 145–6
  - design issues 146–55
  - evaluation phase 141
  - general design features 145–6
  - high-performance processor designs 9–12
  - noise immunity 146
  - overview 140–6
  - power dissipation 146
  - precharge phase 141
  - speed 145
  - timing verification 490
- dynamic-to-static converter (DSC) circuit 231
- dynamic transmission gate latch 141
- edge rate 356–7
  - control 380–1
- edge-triggered clocking 216–17, 220–1
- effective capacitance 346, 348, 349
- effective current density 438
- electrical effort 137
- electromigration 275
  - damage 429, 437
  - design rule 439
  - driving force 436
  - failure times 435
  - flux 430, 436
  - materials and process effects 430–3
  - modeling 435–7
  - transport 434–5
- electromigration-induced stress gradient 436
- electromigration lifetime 432–8
  - accelerated testing 435
- electromigration reliability 429–48
  - budgeting 438–9
  - characterization 435
  - design 438–45
  - pulsed dc and ac operation 437–8
  - risk 443

- risk assessment 444
- electron-hole pairs 307
- electron wind 429
- electronic design automation (EDA) tools 350
- electrostatic discharge (ESD) 386–7
  - circuit design 377–96
  - circuit topology of protection network 388
  - double diode protection 392–3
  - models 387–8
  - protection design elements and methods 389–93
  - protection device 66
    - qualities of good protection 388–9
- Elmore delay 335–6
- embedded DRAM 309–28
  - design issues 323–6
  - device structure 324
  - features 319–20
  - future 326–7
  - gate density 323
  - low-voltage operation 325–6
  - LSIs 320
  - memory bus characteristics 319
  - memory cell array architecture 314–15
  - memory cell capacitor selection 322–3
  - memory latency reduction 320
  - process issues 321–3
  - transistor performance 323
- embedded dual- $V_t$  design 57–8
- embedded RAM self-test 539
- embedded SRAM arrays 286
- energy consumption 18
- energy-delay product 20, 63, 137
- environmental variation 267
- equi-power curves 64
- equi-speed curves 64
- error correction code (ECC) 136
- eye diagrams 403–4
  
- Failure Capture Memory 529
- false switching 123, 129
- fan-in 125
- fan-out 125, 133
- fanout-of-four (FO-4) 402–3, 409, 413, 422
- fault coverage 528
- fault models 525
- fault simulation 528–9
- final row drivers 299–300
- finite impulse response (FIR) 419, 421
- first-order models 338
- flip-chip technology packaging 21–2
- flip-flops 211–13
  - amplifier-based 223–5
  - comparison with latch-pairs 220–1
  - designs based on 487
  - edge-triggered clocking 216–17
  - latency 212, 221
  - two-way mux pulsed 227
    - see also* hybrid latch flip-flop element (HLFF)
  - floating body 84–6
  - floating well driver 410–11
  - 4:2 compressor 197
    - pass-transistor implementation 202
  - four-level pulse amplitude modulation (4-PAM) 421
  - four-way OR 134
  - fourth-order RC circuit 339
  - Fowler–Nordheim (FN) tunneling 50
  - full adder 181–2
    - pass-transistor realization 182
  - full scan design 530
  - fully depleted (FD) SOI device, design 80–3
  - functional fault models 526–7
  - functional (hold time) analysis 481–2
  - functional tests 528
  
  - gain normalization factor 242
  - gate coupling 152–3
  - gate current 450
  - gate density, embedded DRAM 323
  - gate induced drain leakage (GIDL) 47, 49–50
  - gate-isolated pseudo-static transparent high-latch (THL) 214, 215
  - gate modulation circuit 391
  - gate oxides 35–6
    - tunneling current 50
  - gate voltage 450
  - gated clocks 268, 272
  - general purpose register files (GPRs) 286
  - gigahertz processors 332
  - glitch latch 232
  - glitching power 123
  - global clock 208–9, 272
  - global net routes 476, 478
  - global *RC* delay 41–2
  - global wire issues 41–2
  - grain boundaries 434
    - microstructure 432
  - grain size 432
  - gridded clock 272
  - ground voltages 155
  
  - H tree 269, 270
  - halo doping 38

- handshake-limited region 170–3  
heat dissipation, physical limits 20  
hierarchical read paths 304–5  
high-density plasmas (HDP) 104  
high-level decisions 5  
high-level restoration 125  
high-performance microprocessor, cycle time 14–15  
high-performance processor design 23  
  dynamic logic 9–12  
high-speed electrical signaling 397–425  
high-to-low (HL) input transitions 490  
history dependence of propagation delay 87–90  
Hitachi's DPL multiplier 199  
hold-time 488  
  analysis 494  
  failure 263–4  
hot carrier 449–52  
  analysis 461–3  
  tools 460  
  damage mechanisms 452–4  
  degradation 449  
    circuit effects 457–9  
    modeling 454–6  
  design guidelines 460  
  injection 51–2  
  reliability 449–65  
    and circuit reliability 460–1  
    stress effects 378  
human body model (HBM) 387  
hybrid latch flip-flop element (HLFF) 225–7  
  timing waveforms 226  
hysteretic gate delay 91–2  
IBM 1 GHz microprocessor 132  
IDQ testing 65, 526  
IDLE model 66  
IDLE state 69  
IEEE 1149.1 standard 540–2  
  applications for chip level testing 541–2  
  architecture 540–1  
image currents 398–9  
impedance matching 384–5  
impedance mismatch 399  
inductance  
  between high-current wires 333  
  delay variations 367–8  
  models, power grids 517–19  
  potential solutions 368–70  
  problems and solutions 364–70  
  scaling effects 354–5  
  inductive impedance 333  
  Inoue's multiplier 200–2  
  input receivers 385–6  
  input vector activation 55–7  
  insulator thickness 39  
interconnect  
  analysis 335–8  
  capacitance 39, 278  
  capacitive coupling 150–2, 156  
  decoupling capacitance 515  
  delay 18  
  dimensions 352  
  driving techniques 352–76  
  effects 331–51  
    local and global 332  
  impact analysis 108–14  
  inductance 17  
  lifetime 431  
  modeling 332–4  
  power dissipation 359  
  process variation 358  
  RC delay 39–42  
  resistance 17, 39  
  scaling 39–40, 331–2  
  sensitivity analysis 111  
  technology evolution 430–1  
  wiring delays 331  
interlevel dielectrics 431–2  
  thickness 100, 111–14, 463  
internal race 211  
inverter transfer function and noise margin 121  
inverters 333  
I/O  
  bidirectional circuit 388  
  circuitry 310  
  connections 21, 22  
  diode-based pad protection 392  
  latch-up 75  
  mixed-voltage 381–4  
  performance 408  
I-RAM 320  
jitter 209, 266, 417  
  definition 262  
  *see also* clock jitter; output jitter  
*k*-factor equation 347–8  
Kirchhoff's laws 54, 365, 440  
Laplace transform 342  
latch-based designs 487–8  
latch latency 211

- latch-pair 210–11  
comparison with flip-flops 220–1  
modified Svensson 222–3
- latch-up immunity 74–5
- latched evaluate logic 130–1
- latching elements  
dynamic circuits 231–3  
inclusion of logic 230  
pseudo-static 214  
rules for robust design 213–15  
state node 228  
*see also* hybrid latch flip-flop element (HLFF)
- LATID 463
- layout-schematic equivalency checkers 475
- LLD 463
- leakage control techniques 55–61
- leakage current 46–7  
components 47–52  
*see also* subthreshold leakage current;  
threshold leakage current
- leakage current monitor (LCM) 67, 69, 70, 73
- leakage fault models 526
- leakage power reduction 46–62
- LEAN/LEAP logic 127–8
- least positive up level (LPUL) 121
- least significant bit (LSB) position 192
- least significant portion 192
- level-sensitive clocking 217–18
- level-sensitive latch 210–11
- level-sensitive scan design (LSSD) 532
- level shifting 384
- line length 432–3
- line space 104
- line width 104, 432–3
- linear feedback shift registers (LFSRs) 535–8
- Ling adder 191
- link margins 403–5
- link performance metrics 402–5
- loadable code-based self-test 534–5
- logic families 119–39
- logic functions 134–5  
circuit style selection 133  
implementing issues 133–8
- logic noise immunity 120
- logic stage 124
- logic styles, advanced 9
- logical effort metrics 137
- long-channel MOSFET 30
- long distance routing, problems and  
solutions 371–5
- long-term jitter 262
- loop filters 258–9
- loop gain normalization constant 248
- low-pass filtering 403–4
- low-swing signaling 373–4
- low-temperature CMOS 42–4
- low-to-high (LH) input transitions 490
- low-voltage low-threshold-voltage circuit  
design 63–5
- low-voltage technologies 63–79
- lumped statistics 99
- machine model (MM) 387
- macro replacement tool 475
- Manchester carry chain (MCC) 187–8
- MARCH algorithm 539–40
- master-slave flip-flop 211, 212
- max flow line 171
- max path modeling 483
- maximum potential barrier 31
- Maxwell's equation 365
- memory blocks, timing verification 492–4
- memory bus characteristics, embedded  
DRAM 319
- memory cell 309, 310–12  
1Tr/1C 311  
array architecture, embedded DRAM 314–15  
plate voltage generator 318
- memory latency reduction, embedded  
DRAM 320
- metal interconnect thickness 104
- metal resistivity 104
- metallization technology 431
- micro-architecture  
definition 3  
design tools 469–71
- microprocessor design process flow diagram 470
- Miller capacitance 152–3, 360, 361
- min path modeling 483
- minority carrier charge injection 153–5
- minority carrier injection 156
- mismatch 107–8
- mixed-voltage I/O 381–4
- model order reduction 338–45
- model parameter extraction 105
- modified Booth algorithm 198
- modified Booth Selector 201
- modified nodal analysis (MNA) 511
- moment matching 338, 342–5
- moments, calculation 340–2
- MOSFET  
channel length 36  
characteristics at low temperature 42  
constant-electric-field scaling 28

- MOSFET (*cont.*)  
 scaling theory 27–31  
*see also* n-MOSFETs; p-MOSFETs  
 subthreshold slope 42  
 transconductance 27
- most positive down level (MPDL) 121
- most significant (MS) portion 192
- mousetrap dual-monotonic latches 174
- multicycle cache arrays 289–90
- multilevel signaling 421–3
- multiple clock domains 273, 485–6
- multiple-input NMOS dynamic gate 141
- multiple-input NMOS gate 140
- multiple-input signature register (MISR) 536
- multiple-output domino logic (MODL) 143
- multiple pass-gate read/write ports 297
- multiple-threshold CMOS (MTCMOS) 57, 66, 78
- multiplexor  
 2 : 1 409–10  
 function 135
- multiplication 193–202  
 algorithm 193–5  
*see also* specific multipliers
- multipumping 175
- mutual inductances 366
- mutual resistance 366
- MUX 13–14
- NAND design 122–3
- narrow width effect 50
- negative skew 262
- nFETs 81, 85, 88, 89
- NMOS  
 domino structure 8  
 dynamic inverter 141  
 inverter 140  
 pull-down network 8  
 transistors 7, 77
- n-MOSFETs 449, 450, 452–7, 461
- noise  
 and delay 334–5  
 data-dependent 279  
 interconnect-induced 331–2  
 noise analysis 337  
 Asymptotic Waveform Evaluation (AWE)  
 344–5  
 noise bound  
 derivation 337  
 for coupled interconnect 337–8
- noise computation 334
- noise immunity 122  
 dynamic logic 146
- noise injection from power grid 501
- noise margin 121, 213  
 reduction in SOI 94
- noise on data input 213
- noise sources 213
- noise-tolerant precharge (NTP) domino logic 143
- noise vulnerability 125
- noise waveform 334
- nonclocked logic 119–28
- nonmemory custom blocks, timing verification 490–2
- nonsilicided NMOS 389–90
- nonuniform channel profile 38
- N-well capacitance 517
- observability registers 538
- off-chip driver 378  
 edge rate control 380–1  
 terminator 385
- off-state leakage current 47
- ohmic impedance 333
- on-chip electrical lengths 357–8
- on-chip induction 333
- on-chip interconnect technology 23
- open drain signaling 382–3
- output-enable-bar 66
- output jitter 250–2
- oxide electron traps 452
- oxide hole traps 452
- PA8000 processor 15–17
- package inductance resonance 502
- packaging  
 advances 21–2  
 technology 20
- parallel multipliers 195
- parasitic delay 137
- parasitic device capacitances 515–17
- parasitic wire capacitances 515
- partial product reduction tree (PPRT) 197
- partial products 193–5, 200
- partial scan design 533
- partially depleted SOI device *see* PD-SOI
- pass-gate circuitry 135–6
- pass-gate logic 125–8, 136  
 timing verification 491–2
- pass-gate multiplexor 136
- pass-gate transient leakage 90–1
- path delay 138
- path effort 137
- PD-SOI CMOS  
 circuit issues 91–5

- digital circuits 87–95
    - power versus access delay 95
    - power/stage versus delay/stage 95
  - PD-SOI device design 80–3
  - performance trade-off vs. power 33–4
  - pFET 85, 88, 89
  - phase detectors 238, 256–7
  - phase-frequency detectors 238
  - phase-locked loops (PLLs) 209, 235–60, 414–16
    - architectures 236–8, 249
    - block diagram 237
    - circuits 255–9
    - closed loop analysis 248
    - closed loop transient step response 244
    - design issues 246–7
    - design strategy 248–9
    - device parameters 247, 248
    - frequency response 242–5
    - function 240–2
    - general structure 237
    - higher order roll-off 245–6
    - loop components 237–8
    - loop parameters 246–7
    - open loop analysis 248–9
    - output jitter 250–3
    - performance issues 249–55
    - phase margin 248–9
    - self-biased techniques 259–60
    - substrate noise sensitivity 254–5
    - supply noise filters 254
    - supply noise sensitivity 253
    - supply/substrate noise response 252
  - phase noise 414–16
  - $\pi$ -model parameters 348
  - pick-2 arbiter 11
  - pick-4 problem 12
  - pipelining 158
    - see also* self-timed pipelines
  - PLA structure 9
  - PMOS
    - dynamic inverter 141
    - inverter 140
    - transistors 77
  - p-MOSFETs 449, 451–5, 457, 461
  - p-n junction reverse bias current 48
  - point-to-point structures 401
  - poles 339–40
  - polycrystalline film 432
  - polysilicon gates 275
  - porosity, clock layout 275–6
  - power, vs. performance trade-off 33–4
  - power consumption 18–21, 23, 124, 137, 500
    - ultimate impact 21
  - power control 21
  - power-delay product 63, 230
  - power dissipation 63, 64
    - capacitance 360
    - dynamic logic 146
    - effect of architectural decisions 20
    - interconnect 359
    - vs. operating frequency 77
  - power distribution networks 499–522
    - analysis 510–14
    - design 501–9
      - methodology 505–7
      - styles 502–5
    - floorplan-based analysis 506–7
    - topology 504
  - power-down scheme 66–7
  - power efficiency 230
  - power estimation 18–20
  - power grid 503–4
    - analysis 510–14
    - comparison between R, RC and RLC models 519
    - simulation vector 513
  - capacitance models 515–17
  - correcting 508
  - design style 504
  - early analysis 505–6
  - fast linear solvers 511–12
  - inductance models 517–19
  - integrity problems 501–2
  - layout-based verification 507
  - models 514–19
  - network model 500
  - noise injection from 501
  - resistance models 514
  - resonance 509
  - simulation vector generation 512–14
  - topology 504
- power pads 508
- power planes 504
- power reduction 21, 63
- power supply 31–4, 378
  - clamps 393–4, 379
  - clock distribution 277–8
  - noise 155
  - ringing 378
  - ripple 215
  - split system 378–9
  - variation 155
- PPRAM 320

- precharged gate 160
- precompensation drivers 385
- preswitching 89
- printed circuit boards (PCBs) 399, 411
- process scaling 331–2
- process variations
  - clock distribution 276
  - device electrical parameters 103
  - device geometry 102
  - device material parameters 102–3
  - environmental factors 98
  - interconnect 358
    - geometry 103–4
    - material parameters 104
  - inter-die 100–1
  - intra-die 100–1
  - methodologies used to characterize and address 105–8
  - models 98–115
  - parametric 98
  - physical factors 98
  - sources 98–9
  - SRAM cell 293–4
  - survey 102–4
  - tolerance 267
- product definition, timing design and verification
  - flow from 495
- product goals 484
- productivity, clock layout 275–6
- propagation delay, history dependence of 87–90
- propagation delay time 63
- propagation of variance 106
- proportional noise 399
- pulse shaping 486–7
- punchthrough 50
- QCRIT** 308
- race management 228, 230
- race-through failure 263–4
- race tolerance metric 230
- rail voltage offsets/biases 125
- RAMDACs 422
- RAMs 12–14, 23
- random noise 405
- random pattern resistant fault 538
- random variations 109–10
- RAPPID 174
- RC circuit 339, 341
- RC interconnect 346
- RC loads
  - driver models 347–9
- $\pi$ -model 346
- RC-matched distribution network 268, 271
- RC-matched tree 268, 270, 271
- RC switching network 440
- RC trees 335, 342
- read paths
  - design 301–5
  - hierarchical 304–5
- read-only pull-down ports 297–8
- receivers 410–13
  - design 412–13
  - equalization 419
  - input noise 410–12
- recurrence solver based adders 190
- redundancy 306–7
  - implementing 307
- reference voltage 407
- reflections, transmission lines 399–400
- register files 285–308
  - basic architecture 286–90
- register-transfer level *see* RTL
- relative distance check 214
- reliability
  - analysis, CAD 442–5
  - caches 307–8
- repeater insertion 371–2
- reset spacer 164
- resistance
  - models, power grids 514
  - problems and solutions 370–1
  - scaling effects 354
  - shielding 346
- resonance frequency 509
- response evaluator 535–7
- retrograde channel profile 44
- retrograde doping 36–7
- reverse bias 70
  - reverse bias p-n junction leakage 48
- ring performance graphs 169–71
- ring performance region edges 171–3
- ripple carry adder 183
- RISC
  - architecture 4–5
  - core 75
- RLC grid models 512
- RMS current limits 445
- routed power distribution networks 502–3
- row address buffer 310
- row decoder 310
- RTL model 494
  - design tools 470–3
  - re-partitioning tool 475

- RTL simulation 493  
sample-set differential logic (SSDL) 130–1  
scaling 18–20  
    alternative technology solutions 356  
    architectural issues 358–9  
    capacitance effects 353  
    classical 19  
        effect on circuit parameters 29  
        inductance effects 354–5  
        issues below  $0.25\text{ }\mu\text{m}$  31–8  
        local interconnect parameters 40  
        resistance effects 354  
        static timing errors 418  
        theoretical effects 19  
        trends 352–9  
scan  
    delay fault testing based on 533  
    design techniques 530–3  
    observability and controllability 229  
Scan-BiST 537–8  
scan-flops 530  
    design and clocking strategy 531–2  
scan-path 530  
schematic drawing tool 476  
self-adjusting threshold-voltage scheme (SAT) 67, 71  
self-biasing techniques 259–60  
self-inductance 333  
self-resetting CMOS (SRCMOS) 131–2  
self-substrate bias circuit (SSB) 67, 69, 70, 73, 74, 77  
self-timed dynamic logic 9  
self-timed logic 131  
    timing verification 492  
self-timed pipelines 158–80  
    applications 173–6  
    clock 162–3  
    control interconnections 164–8  
    definitions 163–4  
    individual stages 160–3  
    minimum-latency 166  
    overall latency and throughput 168–73  
    overview 158–9  
    parameter definitions 169  
    power issues 176–8  
    reverse latency 166  
    ring, performance region edges 171–3  
    testing 176–8  
self-timed ring, performance graphs 169–71  
sense amplifier 310, 373–4  
    design 302–4  
sense amplifier enable signal (SAE) 492–3  
sensitivity analysis 106  
sequential logic, timing properties of 215  
set-up time 488  
    violation 217  
set-up-time, failure 263  
SH-4 microprocessor 77  
Shannon's theorem 421  
shift-test 530  
short-channel effects 44, 59–60, 82  
short-channel MOSFET 29  
signal path resistor network 441  
signal-to-noise ratio (SNR) 418, 420  
signaling methods 401–2  
signaling system components 397  
signature analyzer 535  
silicide-blocked NMOS 389–90  
silicide formation 103  
silicided NMOS 390–2  
silicon dioxide 431–2  
    alternatives 432  
SIMIC 176  
simultaneous switching noise (SSN) 378  
single-bubble diagonal edge 172  
single-bubble line 170  
single-cycle caches 286–9  
single-cycle current envelope compression technique 513  
single-edge optimizations 161  
single-edge tuning 162  
single-ended write operation 298  
single-rail monotonic signal 231  
single-token line 170–2  
skew 266, 269, 272  
    definition 262  
    *see also* clock skew  
skew-tolerant level-sensitive clocking 217  
slack-passing 211, 219–20  
sleep mode 34  
sleep transistor 66  
slew-rate control 407–8  
soft error upsets (SEUs) 307–8  
SOI devices 80–97  
    low-power behavior 95–6  
    self-heating 80–4, 90–1  
     $V_T$  mismatch 94–5  
source-coupled transistors 224  
source follower action 135  
spatial  $\Delta L$  effects 110  
spatial variation modeling 107–8  
speed (set-up time) analysis 481–2  
SPICE 67, 347, 392, 439, 445, 458, 459, 463  
Spice Level-1 MOSFET model 105

- split row decoder scheme 299
- SRAM 12, 15, 71, 74, 92, 125, 286, 287, 492, 493, 524
  - design 286, 290–8
  - operation 290–8
  - read operation 291–4
  - sizing and process variations 293–4
  - soft error upsets 307–8
  - variants on basic 6T cells 297–8
  - write operation 294–6
- SRCMOS 162, 168
- STA program 493
- STA tools 489–91, 495–8
- stack effect 52–3
- stack leakage 55
- standby leakage control 55–7
- standby power 32
- standby power reduction scheme (SPR) 67, 70, 71, 77
- static circuit selection 134
- static combinatorial CMOS logic 120–4
- static gates 156
- static peak current generation 513
- static timing
  - analysis tools 486
  - calibration techniques 418
  - errors, scaling of 417–18
- static-to-dynamic converter (SDC) circuit 231
- statistical descriptions 99–101
- statistical device models 105–6
- statistical electromigration budgeting
  - (SEB) 439
- statistical fault analysis (STAFAN) 529
- statistical interconnect impact 111–12
- steady-state leakage model 53–5
- stimulus source 535–7
- storage element timing characterization 488–9
- stress voiding 431
- StrongARM latch 412
- structural tests 528
- stuck-at fault models 525
- stuck-at fault testing 177
- substrate bias 67–75, 78
- substrate-impedance scheme 78
- substrate noise influence 74
- subthreshold currents 44, 86
- subthreshold I–V characteristics 43
- subthreshold leakage current 52–5, 64, 67
- super cutoff CMOS (SCCMOS) 67
- super-halo 38
- supply/substrate noise 417
- supply voltage 46
- Svensson latch-pair 222–3
- switch point (SWP) 121
- symmetric loads 256
- temperature variation, clock distribution 278–9
- test patterns generation 478
- testing 523–44
  - basic concepts 524–30
  - Built-in Self Test (BiST) techniques 534
  - components of testing 524–5
  - designing for testability 530–42
  - fault modeling and testing methods 525–7
  - need for testing 523–4
  - self-timed pipelines 176–8
  - test development 527–9
  - test engineering 529
- thin-film interconnects 429
- three-dimensional optimization method
  - (TDM) 197
- threshold leakage current 46
- threshold voltage 31–4, 36–7, 42, 43, 46
  - nonscaling 32–3
- time-borrowing 211, 219–20
- time constant 53
- time-dependent dielectric breakdown
  - (TDDB) 307, 378
- timing analysis process 482
- timing circuit scaling 417–18
- timing diagram 264
- timing properties of sequential logic 215
- timing verification 480–98
  - asynchronous logic 492
  - design changes 496
  - design flow 494–7
  - design guidelines 489
  - dynamic logic 490
  - full-chip 494–7
  - future challenges 497
  - goals 480
  - key factors 484–98
  - memory blocks 492–4
  - non-memory custom blocks 490–2
  - pass-gate logic 491–2
  - self-timed logic 492
  - tokens 164, 168–70, 173
  - trade-offs 5
  - transfer function 121
  - transient leakage, pass-gate 90–1
  - transient pass-gate leakage 92–4
  - transistor performance, embedded DRAM 323
  - transistor scaling trends 463–4
  - transistor size ratio 70

- transmission-gate level-sensitive latch 223
- transmission gates 126
- transmission lines 398–402
  - attenuation 400–1
  - reflections 399–400
- transmitted variability 106
- transmitters 405–10
  - equalization 419–22
  - output drivers 405–7
- transparent high latch (THL) 210–11
  - gate-isolated pseudo-static 214
- transparent low latch (TLL) 211
- trees 125, 195–7, 269–71, 335, 342
- triplets 263
- tunability, clock layout 275–6
- tunneling current 30, 35
- twisted bit line architectures 301
- two-dimensional scale length theory 29–31
- two-stage inverter chain 440
- TX3900 75–6
- ULSI circuits 439, 446
- Ultrasparc-III 18, 19
- unity gain point (UGP) 121
- unrolled ring line 170, 172
- user-generated hot-loops 513
- variable block adder (VBA) 184–5
- variable threshold-voltage CMOS (VTCMOS) 67–9, 78
  - area penalty 74
  - circuit techniques 69–71
  - performance and penalty 71–5
  - power and area penalty 72–4
  - power current 73
  - $V_{TH}$  controllability 71–2
  - $V_{dd}$  reduction 21
  - verification, clock design 276
- victim signals 151–2
- void growth 436
- voltage-controlled delay line (VCDL) 237, 238, 255–6, 259
- voltage-controlled oscillator (VCO) 237, 238, 240–2, 245, 255–6, 258–9
- voltage down converter 318
- voltage drop 501, 506, 511, 519
  - distribution points 499
  - mitigation 507–8
- voltage fluctuations 500, 501, 509, 517
- voltage generator 315–18
  - basic configuration 315–16
  - memory cell plate 318
- voltage-mode drivers 406–7
- wafer-level variation impact 109–10
- wafer-probe testing 177
- Wallace/Dadda multiplier 195–6
- Wallace Tree 195–7
- wave pipelining 374–5
- weak inversion 49
- wire-bond connections 22
- within-die variations 59, 61
- word line hierarchies and layout 300–1
- worst-case analysis 106–7
- write amplifier design 305
- write path design 305–6
- X tree 269–71
- XOR function 9
- yield, clock driver layout 276
- zero detection circuit 7
- zero-overhead line 171–3
- zeros 339–40

# DESIGN OF HIGH-PERFORMANCE MICROPROCESSOR CIRCUITS

This book covers the design of next generation microprocessors in deep submicron CMOS technologies. The chapters in *Design of High-Performance Microprocessor Circuits* were written by some of the world's leading technologists, designers, and researchers. All levels of system abstraction are covered, but the emphasis rests squarely on circuit design. Examples are drawn from processors designed at AMD, Digital/Compaq, IBM, Intel, MIPS, Mitsubishi, Motorola, and Toshiba.

Each topic of this invaluable reference stands alone so the chapters can be read in any order. The following topics are covered in depth:

- Architectural constraints of CMOS VLSI design
- Technology scaling, low-power devices, SOI, and process variations
- Contemporary design styles including a survey of logic families, robust dynamic circuits, asynchronous logic, self-timed pipelines, and fast arithmetic units
- Latches, clocks and clock distribution, phase-locked and delay-locked loops
- Register file, cache memory, and embedded DRAM design
- High-speed signaling techniques and I/O design
- ESD, electromigration, and hot-carrier reliability
- CAD tools, including timing verification and the analysis of power distribution schemes
- Test and testability

*Design of High-Performance Microprocessor Circuits* assumes a basic knowledge of digital circuit design and device operation, and covers a broad range of circuit styles and VLSI design techniques. Packed with practical know-how, it is an indispensable reference for practicing circuit designers, architects, system designers, CAD tool developers, process technologists, and researchers. It is also an essential text for VLSI design courses.

## About the Editors

**Anantha Chandrakasan** is an associate professor of electrical engineering and computer science at the Massachusetts Institute of Technology. Dr. Chandrakasan has received numerous awards and has served on the technical program committees of various IEEE and ACM conferences. His research interests include the energy efficient implementation of DSPs, wireless microsensor networks, and CAD tools for VLSI.

**William J. Bowhill** is a principal member of technical staff in Compaq Computer Corporation's Alpha Development Group (formerly Digital Equipment Corp.). He has contributed to many VAX and Alpha microprocessor designs. Prior to joining Digital in 1985, he designed telecommunication chips for Standard Telecommunication Research Laboratories in England. He has a B.Eng. from Liverpool University.

**Frank Fox** (Thomas F. Fox) has been a vice president at Rambus Inc., since 1998. Previously, he worked for Digital Equipment Corporation, where he was a principal member of the Alpha processor design team. During his fourteen years at Digital, he contributed to the development of the Alpha processor technology and CAD tools. Dr. Fox has a B.Sc. from the University of Dublin, Trinity College, Ireland and a Ph.D. from Trinity College, Dublin.

IEEE Press  
445 Hoes Lane  
P.O. Box 1331  
Piscataway, NJ 08855-1331 U.S.A.  
+1 800 678 IEEE (Toll-free in U.S.A. and Canada)  
or +1 732 981 0060

Shop at the IEEE store! Visit <http://www.ieee.org>

