

ARCHITECTURE DESIGN FOR SOFT ERRORS

Shubu Mukherjee



ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann Publishers is an imprint of Elsevier



MORGAN KAUFMANN PUBLISHERS

Contents

Foreword	xiii
Preface	xvii

1 Introduction 1

1.1 Overview	1
1.1.1 Evidence of Soft Errors	2
1.1.2 Types of Soft Errors	3
1.1.3 Cost-Effective Solutions to Mitigate the Impact of Soft Errors	4
1.2 Faults	6
1.3 Errors	7
1.4 Metrics	9
1.5 Dependability Models	11
1.5.1 Reliability	12
1.5.2 Availability	13
1.5.3 Miscellaneous Models	13
1.6 Permanent Faults in Complementary Metal Oxide Semiconductor Technology	14
1.6.1 Metal Failure Modes	15
1.6.2 Gate Oxide Failure Modes	17
1.7 Radiation-Induced Transient Faults in CMOS Transistors	20
1.7.1 The Alpha Particle	20
1.7.2 The Neutron	21
1.7.3 Interaction of Alpha Particles and Neutrons with Silicon Crystals	26
1.8 Architectural Fault Models for Alpha Particle and Neutron Strikes	30
1.9 Silent Data Corruption and Detected Unrecoverable Error	32
1.9.1 Basic Definitions: SDC and DUE	32
1.9.2 SDC and DUE Budgets	34

1.10	Soft Error Scaling Trends	36
1.10.1	SRAM and Latch Scaling Trends	36
1.10.2	DRAM Scaling Trends	37
1.11	Summary	38
1.12	Historical Anecdote	39
	References	40
2	Device- and Circuit-Level Modeling, Measurement, and Mitigation	43
2.1	Overview	43
2.2	Modeling Circuit-Level SERs	44
2.2.1	Impact of Alpha Particle or Neutron on Circuit Elements	45
2.2.2	Critical Charge (Qcrit)	46
2.2.3	Timing Vulnerability Factor	50
2.2.4	Masking Effects in Combinatorial Logic Gates	52
2.2.5	Vulnerability of Clock Circuits	59
2.3	Measurement	60
2.3.1	Field Data Collection	62
2.3.2	Accelerated Alpha Particle Tests	62
2.3.3	Accelerated Neutron Tests	63
2.4	Mitigation Techniques	67
2.4.1	Device Enhancements	67
2.4.2	Circuit Enhancements	68
2.5	Summary	74
2.6	Historical Anecdote	76
	References	76
3	Architectural Vulnerability Analysis	79
3.1	Overview	79
3.2	AVF Basics	80
3.3	Does a Bit Matter?	81
3.4	SDC and DUE Equations	82
3.4.1	Bit-Level SDC and DUE FIT Equations	83
3.4.2	Chip-Level SDC and DUE FIT Equations	84
3.4.3	False DUE AVF	86
3.4.4	Case Study: False DUE from Lockstepped Checkers	87
3.4.5	Process-Kill versus System-Kill DUE AVF	89
3.5	ACE Principles	90
3.5.1	Types of ACE and Un-ACE Bits	90
3.5.2	Point-of-Strike Model versus Propagated Fault Model	91
3.6	Microarchitectural Un-ACE Bits	93
3.6.1	Idle or Invalid State	93
3.6.2	Misspeculated State	93
3.6.3	Predictor Structures	93
3.6.4	Ex-ACE State	93

3.7	Architectural Un-ACE Bits	94
3.7.1	NOP Instructions	94
3.7.2	Performance-Enhancing Operations	94
3.7.3	Predicated False Instructions	95
3.7.4	Dynamically Dead Instructions	95
3.7.5	Logical Masking	96
3.8	AVF Equations for a Hardware Structure	96
3.9	Computing AVF with Little's Law	98
3.9.1	Implications of Little's Law for AVF Computation	101
3.10	Computing AVF with a Performance Model	101
3.10.1	Limitations of AVF Analysis with Performance Models	103
3.11	ACE Analysis Using the Point-of-Strike Fault Model	106
3.11.1	AVF Results from an Itanium [®] 2 Performance Model	107
3.12	ACE Analysis Using the Propagated Fault Model	114
3.13	Summary	118
3.14	Historical Anecdote	118
	References	119
4	Advanced Architectural Vulnerability Analysis	121
4.1	Overview	121
4.2	Lifetime Analysis of RAM Arrays	123
4.2.1	Basic Idea of Lifetime Analysis	123
4.2.2	Accounting for Structural Differences in Lifetime Analysis	125
4.2.3	Impact of Working Set Size for Lifetime Analysis	129
4.2.4	Granularity of Lifetime Analysis	130
4.2.5	Computing the DUE AVF	131
4.3	Lifetime Analysis of CAM Arrays	134
4.3.1	Handling False-Positive Matches in a CAM Array	135
4.3.2	Handling False-Negative Matches in a CAM Array	137
4.4	Effect of Cooldown in Lifetime Analysis	138
4.5	AVF Results for Cache, Data Translation Buffer, and Store Buffer	140
4.5.1	Unknown Components	140
4.5.2	RAM Arrays	142
4.5.3	CAM Arrays	145
4.5.4	DUE AVF	146
4.6	Computing AVFs Using SFI into an RTL Model	146
4.6.1	Comparison of Fault Injection and ACE Analyses	147
4.6.2	Random Sampling in SFI	149
4.6.3	Determining if an Injected Fault Will Result in an Error	151
4.7	Case Study of SFI	152
4.7.1	The Illinois SFI Study	152
4.7.2	SFI Methodology	152
4.7.3	Transient Faults in Pipeline State	154
4.7.4	Transient Faults in Logic Blocks	156

4.8	Summary	158
4.9	Historical Anecdote	159
	References	160
5	Error Coding Techniques	161
5.1	Overview	161
5.2	Fault Detection and ECC for State Bits	162
5.2.1	Basics of Error Coding	162
5.2.2	Error Detection Using Parity Codes	168
5.2.3	Single-Error Correction Codes	170
5.2.4	Single-Error Correct Double-Error Detect Code	174
5.2.5	Double-Error Correct Triple-Error Detect Code	176
5.2.6	Cyclic Redundancy Check	178
5.3	Error Detection Codes for Execution Units	181
5.3.1	AN Codes	182
5.3.2	Residue Codes	183
5.3.3	Parity Prediction Circuits	185
5.4	Implementation Overhead of Error Detection and Correction Codes	187
5.4.1	Number of Logic Levels	187
5.4.2	Overhead in Area	189
5.5	Scrubbing Analysis	190
5.5.1	DUE FIT from Temporal Double-Bit Error with No Scrubbing	191
5.5.2	DUE Rate from Temporal Double-Bit Error with Fixed-Interval Scrubbing	193
5.6	Detecting False Errors	194
5.6.1	Sources of False DUE Events in a Microprocessor Pipeline	195
5.6.2	Mechanism to Propagate Error Information	197
5.6.3	Distinguishing False Errors from True Errors	198
5.7	Hardware Assertions	200
5.8	Machine Check Architecture	202
5.8.1	Informing the OS of an Error	202
5.8.2	Recording Information about the Error	203
5.8.3	Isolating the Error	203
5.9	Summary	203
5.10	Historical Anecdote	205
	References	205
6	Fault Detection via Redundant Execution	207
6.1	Overview	207
6.2	Sphere of Replication	208
6.2.1	Components of the Sphere of Replication	208
6.2.2	The Size of Sphere of Replication	209
6.2.3	Output Comparison and Input Replication	211

6.3	Fault Detection via Cycle-by-Cycle Lockstepping	212
6.3.1	Advantages of Lockstepping	213
6.3.2	Disadvantages of Lockstepping	213
6.3.3	Lockstepping in the Stratus ftServer	216
6.4	Lockstepping in the Hewlett-Packard NonStop Himalaya Architecture	218
6.5	Lockstepping in the IBM Z-series Processors	220
6.6	Fault Detection via RMT	222
6.7	RMT in the Marathon Endurance Server	223
6.8	RMT in the Hewlett-Packard NonStop® Advanced Architecture	225
6.9	RMT Within a Single-Processor Core	227
6.9.1	A Simultaneous Multithreaded Processor	228
6.9.2	Design Space for SMT in a Single Core	229
6.9.3	Output Comparison in an SRT Processor	230
6.9.4	Input Replication in an SRT Processor	232
6.9.5	Input Replication of Cached Load Data	234
6.9.6	Two Techniques to Enhance Performance of an SRT Processor	236
6.9.7	Performance Evaluation of an SRT Processor	238
6.9.8	Alternate Single-Core RMT Implementation	239
6.10	RMT in a Multicore Architecture	240
6.11	DIVA: RMT Using Specialized Checker Processor	241
6.12	RMT Enhancements	244
6.12.1	Relaxed Input Replication	244
6.12.2	Relaxed Output Comparison	245
6.12.3	Partial RMT	245
6.13	Summary	247
6.14	Historical Anecdote	248
	References	250

7 Hardware Error Recovery 253

7.1	Overview	253
7.2	Classification of Hardware Error Recovery Schemes	254
7.2.1	Reboot	255
7.2.2	Forward Error Recovery	255
7.2.3	Backward Error Recovery	256
7.3	Forward Error Recovery	258
7.3.1	Fail-Over Systems	258
7.3.2	DMR with Recovery	259
7.3.3	Triple Modular Redundancy	260
7.3.4	Pair-and-Spare	262
7.4	Backward Error Recovery with Fault Detection Before Register Commit	263
7.4.1	Fujitsu SPARC64 V: Parity with Retry	264
7.4.2	IBM Z-Series: Lockstepping with Retry	265

7.4.3	Simultaneous and Redundantly Threaded Processor with Recovery	266	
7.4.4	Chip-Level Redundantly Threaded Processor with Recovery (CRTR)	269	
7.4.5	Exposure Reduction via Pipeline Squash	270	
7.4.6	Fault Screening with Pipeline Squash and Re-execution	273	
7.5	Backward Error Recovery with Fault Detection before Memory Commit	277	
7.5.1	Incremental Checkpointing Using a History Buffer	278	
7.5.2	Periodic Checkpointing with Fingerprinting	280	
7.6	Backward Error Recovery with Fault Detection before I/O Commit	283	
7.6.1	LVQ-Based Recovery in an SRT Processor	284	
7.6.2	ReVive: Backward Error Recovery Using Global Checkpoints	288	
7.6.3	SafetyNet: Backward Error Recovery Using Local Checkpoints	290	
7.7	Backward Error Recovery with Fault Detection after I/O Commit	292	
7.8	Summary	292	
7.9	Historical Anecdote	294	
	References	294	

8 Software Detection and Recovery 297

8.1	Overview	297	
8.2	Fault Detection Using SIS	299	
8.3	Fault Detection Using Software RMT	301	
8.3.1	Error Detection by Duplicated Instructions	303	
8.3.2	Software-Implemented Fault Tolerance	305	
8.3.3	Configurable Transient Fault Detection via Dynamic Binary Translation	306	
8.4	Fault Detection Using Hybrid RMT	309	
8.4.1	CRAFT: A Hybrid RMT Implementation	310	
8.4.2	CRAFT Evaluation	311	
8.5	Fault Detection Using RVMS	313	
8.6	Application-Level Recovery	315	
8.6.1	Forward Error Recovery Using Software RMT and AN Codes for Fault Detection	315	
8.6.2	Log-Based Backward Error Recovery in Database Systems	317	
8.6.3	Checkpoint-Based Backward Error Recovery for Shared-Memory Programs	319	
8.7	OS-Level and VMM-Level Recoveries	322	
8.8	Summary	323	
	References	324	

Index	327
-------	-----