

Contents

CHAPTER 1: BACKGROUND

| | |
|--|----|
| 1. Introduction | 1 |
| 2. Definitions | 4 |
| 2.1 Faults, errors and failures | 4 |
| 2.2 A taxonomy of faults | 6 |
| 2.3 Classifying the effects of faults | 7 |
| 2.4 Dependability and its attributes | 9 |
| 3. Error models for hardware and software components | 10 |
| 3.1 Error models for hardware components | 10 |
| 3.1.1 Hardware-level error models | 12 |
| 3.1.2 System-level error models | 13 |
| 3.1.3 Hardware-level errors vs. system-level errors | 15 |
| 3.2 Error models for software components | 19 |
| 3.2.1 Error models at the source-code level | 20 |
| 3.2.2 Error models at the executable-code level | 21 |
| 4. Origin of single-event effects | 22 |
| 4.1 Sources of highly energized particles | 22 |
| 4.1.1 Space radiation environment | 22 |
| 4.1.2 Atmospheric radiation environment | 23 |
| 4.1.3 Ground radiation environment | 23 |
| 4.2 Physical origin of single-event effects | 24 |
| 4.2.1 Direct ionization | 24 |
| 4.2.2 Indirect ionization | 25 |
| 4.3 Single-event effects in memory circuits | 25 |
| 4.4 SEU mechanisms in DRAMs | 25 |
| 4.5 SEU mechanisms in SRAMs | 27 |

| | | |
|-----|--|----|
| 4.6 | Single-event effects in logic circuits | 28 |
| 4.7 | Propagating and latching of SETs | 30 |
| 5. | Redundancy techniques | 30 |
| 5.1 | Hardware redundancy | 31 |
| 5.2 | Information redundancy | 32 |
| 5.3 | Time redundancy | 33 |
| 5.4 | Software redundancy | 34 |
| 6. | References | 35 |

CHAPTER 2: HARDENING THE DATA

| | | |
|-------|--|----|
| 1. | Introduction | 37 |
| 2. | Computation Duplication | 38 |
| 2.1 | Methods based on instruction-level duplication | 38 |
| 2.1.1 | High-level instruction duplication | 38 |
| 2.1.2 | Selective instruction duplication | 42 |
| 2.1.3 | Assembly-Level Instruction Duplication | 45 |
| 2.2 | Procedure-level duplication | 49 |
| 2.2.1 | Selective procedure call | 49 |
| 2.3 | Program-level duplication | 54 |
| 2.3.1 | Time redundancy | 54 |
| 2.3.2 | Simultaneous multithreading | 56 |
| 2.3.3 | Data diversity | 57 |
| 3. | Executable assertions | 59 |
| 4. | References | 61 |

CHAPTER 3: HARDENING THE CONTROL FLOW

| | | |
|-----|---|----|
| 1. | Introduction | 63 |
| 2. | Background | 63 |
| 3. | Path identification | 70 |
| 3.1 | The approach | 70 |
| 3.2 | Experimental results | 73 |
| 3.3 | Advantages and limitations | 73 |
| 4. | CFE detection in sequential and parallel programs | 74 |
| 4.1 | The approach | 74 |
| 4.2 | Experimental results | 75 |
| 4.3 | Advantages and limitations | 75 |
| 5. | BEEC and ECI | 76 |
| 5.1 | The approach | 76 |
| 5.2 | BEEC | 76 |
| 5.3 | ECI | 78 |
| 5.4 | Experimental results | 78 |
| 5.5 | Advantages and limitations | 79 |

| | |
|--|-----|
| 6. Exploiting instruction level parallelism: ARC technique | 79 |
| 6.1 The approach | 79 |
| 6.2 Experimental results | 81 |
| 6.3 Advantages and limitations | 81 |
| 7. VASC | 82 |
| 7.1 The approach | 82 |
| 7.2 Experimental results | 83 |
| 7.3 Advantages and limitations | 85 |
| 8. ECCA | 85 |
| 8.1 The approach | 85 |
| 8.2 ECCA-HL | 86 |
| 8.3 ECCA-IL | 88 |
| 8.4 Experimental results | 89 |
| 8.5 Advantages and limitations | 90 |
| 9. Plain inter-block errors detection | 91 |
| 9.1 The approach | 91 |
| 9.2 Experimental results | 92 |
| 9.3 Advantages and limitations | 92 |
| 10. CFC via regular expressions resorting to IPC | 93 |
| 10.1 The approach | 93 |
| 10.2 Experimental results | 94 |
| 10.3 Advantages and limitations | 94 |
| 11. CFCSS | 95 |
| 11.1 The approach | 95 |
| 11.2 Experimental results | 98 |
| 11.3 Advantages and limitations | 98 |
| 12. ACFC | 99 |
| 12.1 The approach | 99 |
| 12.2 Experimental results | 101 |
| 12.3 Advantages and limitations | 102 |
| 13. YACCA | 103 |
| 13.1 The approach | 103 |
| 13.2 Experimental results | 105 |
| 13.3 Advantages and limitations | 107 |
| 14. SIED and its enhancements | 108 |
| 14.1 The approach | 108 |
| 14.1.1 Intra-block detection | 108 |
| 14.1.2 Inter-block detection | 111 |
| 14.2 Experimental results | 113 |
| 14.3 Advantages and limitations | 114 |
| 15. References | 114 |

| | |
|---|-----|
| CHAPTER 4: ACHIEVING FAULT TOLERANCE | |
| 1. Introduction | 117 |
| 2. Design diversity | 117 |
| 2.1 N-version programming | 119 |
| 2.1.1 Time redundancy | 121 |
| 2.2 Recovery Block | 122 |
| 2.2.1 Distributed recovery block | 125 |
| 3. Checkpointing | 130 |
| 4. Algorithm-based fault tolerance (ABFT) | 132 |
| 4.1 Basic technique | 132 |
| 4.2 Matrix multiplication | 132 |
| 4.2.1 Method description | 132 |
| 4.2.2 Comments | 138 |
| 4.3 FFT | 138 |
| 4.3.1 Method description | 139 |
| 4.4 Final comments | 141 |
| 5. Duplication | 142 |
| 5.1 Duplication and checksum | 142 |
| 5.1.1 Detecting and correcting transient faults affecting data | 142 |
| 5.1.2 Detecting and correcting transient faults affecting the code | 144 |
| 5.1.3 Results | 145 |
| 5.2 Duplication and hamming code | 147 |
| CHAPTER 5: HYBRID TECHNIQUES | |
| 1. Introduction | 153 |
| 2. Control flow checking | 154 |
| 2.1 Assigned run-time signature control-flow checking | 157 |
| 2.1.1 Structural integrity checking (SIC) | 157 |
| 2.2 Derived run-time signature control-flow checking | 159 |
| 2.2.1 Embedded signature monitoring | 159 |
| 2.2.2 Stored reference | 165 |
| 2.2.3 Reference program | 167 |
| 3. Memory access checking | 169 |
| 4. Reasonableness checking | 171 |
| 4.1 Watchdog methods for special purpose applications | 172 |
| 4.2 Watchdog methods for general purpose applications | 172 |
| 5. Combined techniques | 173 |
| 5.1 Duplication and watchdog | 174 |
| 5.2 Infrastructure-IP | 176 |
| 5.2.1 Support for control flow checking | 178 |

| | |
|---|-----|
| 5.2.2 Support for data checking | 180 |
| 5.2.3 Error detection capabilities | 183 |
| 5.2.4 Experimental results | 187 |
| 5.2.4.1 Analysis of fault detection capabilities | 188 |
| 5.2.4.2 Faults affecting the code | 188 |
| 5.2.4.3 Faults affecting the data | 189 |
| 5.2.4.4 Faults affecting the processor memory elements | 190 |
| 5.2.4.5 Overhead analysis | 190 |
| CHAPTER 6: FAULT INJECTION TECHNIQUES | |
| 1. Introduction | 199 |
| 2. The FARM Model | 199 |
| 2.1 Fault injection requirements | 200 |
| 2.2 Intrusiveness | 200 |
| 2.3 Speed | 201 |
| 2.3.1 Speeding-up the single fault-injection experiment | 201 |
| 2.3.2 Reducing the fault list size | 201 |
| 2.4 Cost | 202 |
| 3. Assumptions | 202 |
| 3.1 Set F | 202 |
| 3.2 Set A | 204 |
| 3.3 Set R | 205 |
| 3.4 Set M | 205 |
| 4. The fault injection environments | 207 |
| 4.1 Simulation-based fault injection | 207 |
| 4.1.1 Golden run execution | 209 |
| 4.1.2 Static fault analysis | 210 |
| 4.1.3 Dynamic fault analysis | 210 |
| 4.1.4 Checkpoint-based optimizations | 212 |
| 4.2 Software-implemented fault injection | 213 |
| 4.2.1 Fault injection manager | 214 |
| 4.2.2 Implementation issues | 216 |
| 4.3 Hybrid fault injection | 218 |
| 4.3.1 The fault injection interface | 219 |
| 4.3.2 Injecting faults | 220 |
| 4.3.3 Memory blocks | 221 |
| 4.3.4 Applying stimuli and observing the system behavior | 222 |
| 4.3.5 The FI process | 223 |
| 5. References | 223 |