



Guideline for Single-Event Effect (SEE) Testing of System on a Chip (SOC) Devices

NASA Electronic Parts and Packaging (NEPP) Program
Office of Safety and Mission Assurance

Steven M. Guertin
Radiation Effects Group
Jet Propulsion Laboratory
California Institute of Technology

February 1, 2018

JPL Pub 18-2, (replaces document only released as an unnumbered NEPP review copy)
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California



Guideline for Single-Event Effect (SEE) Testing of System on a Chip (SOC) Devices

NASA Electronic Parts and Packaging (NEPP) Program
Office of Safety and Mission Assurance

Steven M. Guertin
Radiation Effects Group
Jet Propulsion Laboratory
California Institute of Technology

JPL Pub 18-2, (replaces document only released as an unnumbered NEPP review copy)

NASA WBS: 724297.40.49
JPL Project Number: 104593
Task Number: 40.49.03.04

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

<http://nepp.nasa.gov>

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the National Aeronautics and Space Administration Electronic Parts and Packaging (NEPP) Program.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

©2018 California Institute of Technology. Government sponsorship acknowledged.

Revision History

Revision	Date	Description
Original	4/2/2014	Limited release of an unnumbered NEPP document for NASA to NASA reviewers. This document was not widely distributed and never formally released.
A	2/13/2018	<p>However, a number of review copies have been in the radiation effects community for more than 2 years. Therefore, we are releasing this iteration as a new document, JPL Pub 18-2, to avoid any possible confusion between this iteration and the earlier version.</p> <p>Minor changes were made to the original review document throughout document per review comments. Added clarifying equations in section 2.1. Added a limited set of references to newer published documents. Clarified laser testing options. Clarified device thinning in section 3. Clarified speed impact due to board selection. Clarified test procedures in section 4. Clarified all figure permissions. Defined some missing acronyms.</p>

Foreword and Acknowledgement

This guideline is presented to help test engineers and program planners to create and execute useful radiation tests of modern, complex system-on-a-chip (SOC) devices. It provides a lot of background material on radiation testing, planning radiation tests, performing those radiation tests, and analyzing the collected data. It is our hope the reader may find it useful for its intended purpose.

This guideline was written with the intention of capturing the state of radiation testing of SOC devices under the loose rationale that an SOC is a device that combines many of the standard elements of a computer. This term SOC has become more generic in recent years, dropping the formal requirement of having the subject device actually approximate an entire system. It is commonly used now to designate devices where many, but not necessarily all devices in a system are incorporated in the SOC, and usually the differentiation occurs based on cost and speed benefit of including the additional devices, as opposed to having them external to the SOC. To simplify pronouncing the document name, it is called a guideline for single event effects (SEE) testing of SOC devices, which will hopefully not cause any confusion.

This guideline targets the type of devices that include memory peripherals, Ethernet, and similar services on the chip. By necessity it was not possible to cover all types of modern SOC's, and a large portion of the SOC devices, such as those in cell phones, are only approximately handled in this guideline.

The author acknowledges the significant contribution of many reviewers who have helped ensure that the material herein is accurate and properly treats topics well, as far as they are developed. And the reviewers have raised significant concerns regarding things that are not included. We appreciate the thoughtful reviews, inputs, and comments from the following people: Heather Quinn, Lewis Cohn, Rocky Koga, Charles Barnes, Allan Johnston, Ken LaBel, Craig Hafer, Brian Wie, Lawrence Clark, James Howard, Keith

Avery, Ethan Cannon, Dale McMorro, Gary Lum, Ian Troxel, and Stephen Buchner. We are exceedingly grateful to those reviewers who provided extensive reviews of the entire document.

Some topics that were indicated by reviewers as lacking could not be addressed fully in this version of the guideline. These topics are as follows.

- Laser evaluation of single events in complex systems can be very helpful for understanding how the system responds, and for manufacturers can help identify things to improve. Lasers are not handled in a complete way, though we have endeavored to respond to useful comments and suggestions from reviewers.
- Testing of SOCs in vacuum is not discussed in the present version but is an important topic.
- For commercial devices, and many test boards, worst case conditions cannot be set, users may be required to establish a safe operating area (SOA) that can be used during radiation testing, but the details of handling this are outside our scope here.
- Simulation of processors for the intention of injecting errors into the simulation was outside the scope of this document.
- Modeling of test software and how to compare it to flight software is not covered in detail. The impact of high levels of total ionizing dose (TID) on SEE response is only discussed as it affects worst-case test conditions for low TID situations.
- Detailed evaluation of angular response and methods to perform such testing are largely excluded, and detailed discussion and evaluation of the effectiveness of built-in-self-test and hardware debugger test vectors were not explored significantly.
- A list of facilities for radiation testing is provided in this guideline, but as of the writing of this Foreword, it is out of date. It should be updated to include multiple proton facilities and updated heavy ion facility information in a future revision. For the present, some discussion of testing at the now-defunct proton facility in Bloomington, Indiana is still included, but that discussion can be applied to most other proton test facilities with energy in the range of 200 MeV.

TABLE OF CONTENTS

Summary.....	1
1.0 General Overview	2
1.1 Introduction	2
1.2 Background Information	2
1.2.1 Why This Guideline	2
1.2.2 Target SOCs	3
1.3 Key Issues.....	4
1.3.1 Device Performance.....	4
1.3.2 Setup and Operation	4
1.3.3 Test Approaches	5
1.3.4 Packaging	5
1.3.5 Thermal Issues.....	5
1.3.6 Operating Frequency and SET.....	5
1.4 Background Information	7
1.4.1 Existing Test Guidelines.....	8
1.4.2 SEE Test Approach on SOCs	8
1.4.3 Boeing Maestro ITC	8
1.4.4 Cobham Gaisler UT699.....	10
1.4.5 Freescale	10
1.4.6 Other Devices.....	11
1.5 Seven-Phase-Approach Guideline Development.....	12
1.5.1 Manufacturer and User Collaboration	12
1.5.2 Peripheral Evaluation Approach.....	13
1.5.3 Impact of Fault Tolerance on SEE Characterization	13
1.5.4 Impact of Radiation Hardening by Design on Characterization.....	13
1.5.5 Impact of Multicore Design on Characterization.....	13
1.5.6 Use of General Test Methods	13
1.5.7 Investigation and Analysis of Prior Sample Testing	13
2.0 Radiation Characterization Needs	15
2.1 Standard Testing Needs	15
2.1.1 Data Needed	15
2.1.2 Heavy Ion Test Data.....	16
2.1.3 Proton Test Data	16
2.1.4 Alternate Exposure Types	18
2.1.5 Worst-Case Conditions	19
2.2 SOC Characterization	20
2.2.1 Time-Model of Test Beam and Targets.....	20
2.2.2 Selecting Test Software and Hardware	22
2.2.3 Event Normalization	22
2.2.4 Estimating Angular Response.....	23
2.2.5 Higher Beam Energies	23
2.2.6 Beam Flux	24
2.3 Software Characterization.....	26

2.4	Summary of Recommendations	27
3.0	Test Preparation.....	29
3.1	Introduction	29
3.2	Beams and Test Facilities	29
3.2.1	Beam Penetration.....	29
3.2.2	Proton Beams.....	30
3.2.3	Heavy Ion Beams.....	30
3.2.4	Laser Sources and Other Beams	30
3.2.5	Facility Review	30
3.3	Device Preparation.....	31
3.3.1	Board Selection	32
3.3.2	Physical Manipulation.....	32
3.3.3	Thermal Control.....	36
3.4	Detection of Errors	37
3.4.1	Expected IO Patterns	37
3.4.2	Hardware Interrogation.....	38
3.4.3	Software Detection	38
3.5	General Approach to Testing	38
3.5.1	Test and Validation.....	38
3.5.2	Customer-Based Testing.....	40
3.5.3	Preparing for Unexpected Errors.....	40
3.6	Radiation Test Approaches for SOCs	40
3.6.1	General Algorithms.....	41
3.6.2	Software to Operate SOCs.....	42
3.6.3	Beam Delivery Structure	46
3.7	Summary of Recommendations	48
3.7.1	Board Selection	48
3.7.2	Device Preparation.....	48
3.7.3	Test Software and Algorithms	48
4.0	Testing.....	50
4.1	General Testing.....	50
4.2	Beam Diagnostics and Data Quality.....	50
4.3	Effective Sensitivity	50
4.3.1	Definition of Effective Sensitivity	51
4.3.2	Determination of Effective Sensitivity	51
4.3.3	Improving Effective Sensitivity.....	53
4.3.4	Recommendations for Effective Sensitivity	54
4.4	Testing Problems	54
4.4.1	Multiple Event Sensitivity.....	55
4.4.2	Flawed Analysis Due to Incorrect Assumptions	55
4.4.3	Flux Dependence	56
4.4.4	Anomalies	57
4.5	Debugging Tools.....	58
4.5.1	How Tools Can Be Used	58
4.5.2	Manufacturer Tools	59
4.5.3	In-Code Debugging	59

4.5.4	Adequate Test Output Files.....	60
4.6	Recommendations	60
5.0	Analysis and Reporting	62
5.1	Pre Irradiation Measurements and Observation.....	62
5.2	Test Log Requirements	63
5.3	Data Analysis	64
5.3.1	Effective Sensitivity	64
5.3.2	Static Counting.....	64
5.3.3	Functional Codes	65
5.3.4	Analysis of I/O Test Data.....	65
5.3.5	Analysis of Upsets in EDAC Protected Systems.....	65
5.4	Test Report Requirements	65
5.4.1	Description of Test Approach	66
5.4.2	Reporting of Results.....	66
5.5	Recommendations	66
6.0	Sample Results.....	67
6.1	Examples Layout.....	67
6.2	Example from UT699 Testing.....	67
6.2.1	Seven Work Points for UT699.....	67
6.2.2	Test Setup.....	69
6.2.3	Response of Caches and Registers.....	70
6.2.4	Initial Anomaly Results	72
6.2.5	Register Partial Reset	72
6.2.6	SpaceWire Testing.....	74
6.2.7	Watchdog Testing	74
6.2.8	UT699 Conclusion.....	74
6.3	Example from Maestro ITC Testing.....	75
6.3.1	Seven Work Points.....	75
6.3.2	Background	76
6.3.3	Test Board Info.....	77
6.3.4	Device Preparation.....	78
6.3.5	Algorithm Architecture.....	79
6.3.6	Test Sensitivity	81
6.3.7	Cache Testing—RHBD vs. Non-RHBD.....	82
6.3.8	Debugging with BTK.....	82
6.3.9	Data Presentation.....	82
6.3.10	Maestro ITC Conclusion.....	84
6.4	Example from Freescale Testing.....	86
6.4.1	Seven Work Points.....	86
6.4.2	Background	87
6.4.3	Commercial Issues.....	89
6.4.4	Test Goals.....	89
6.4.5	Hardware Setup	90
6.4.6	Data Presentation.....	94
6.4.7	Freescale Conclusion.....	98
6.5	Other SOC Test Efforts	99

6.5.1	PowerQUICC III with Whetstone	99
6.5.2	STMicroelectronics 8051-Based SOC.....	99
6.5.3	Using DfT Circuits	100
6.5.4	Other General Microprocessor Testing	100
6.5.5	Intel/AMD Testing.....	102
6.5.6	Testing of FPGAs.....	103
6.5.7	Simulating SEE to Obtain System Error Rates.....	105
6.6	Recommendations	106
7.0	Conclusion	107
8.0	References.....	108
9.0	Acronyms	114

Figures

Figure 1-1.	Diagram showing how increased frequency can contribute to higher rates for upset in digital circuits. What is shown is the basic operation of a clock-edge triggered latch. The data must be valid where the upper portion is high which is only when the clock is transitioning. If an SET occurs between these regions it does not affect anything. However, if we were running at twice the clock rate then the SET would have disrupted the valid data during a window where it must be valid. Doubling the frequency effectively doubles the SET sensitivity.....	6
Figure 1-2.	Sensitivity of 0.18 and 0.25 μm DICE latches to SETs as a function of frequency. Note that DICE latches in these feature sizes are not expected to be sensitive to direct upset of the cells in this cross section range, after [23]......	7
Figure 1-3.	Critical transient width as a function of feature size suggests that submicron technology nodes, such as 90 nm, 45 nm, and below, are expected to have significant SET sensitivity, after [24]......	7
Figure 1-4.	The block diagram of the Maestro 49-core many core microprocessor. Image used with permission. The center of the diagram shows the 49-core tile array, which is surrounded by various peripheral systems. The tiles are supported by five on-chip networks. DDR2 = double data rate 2, XAUI = X [for 10 Gigabit] Attachment User Interface, GPIO = general purpose input/output, RMGII = Reduced Gigabit Media Independent Interface, CLK = clock, I2C (-h/-s) = Inter-Integrated Circuit interface high-speed and standard, HPI = Hardware Platform Interface, UART = Universal Asynchronous Receiver/Transmitter, SPI = Serial Peripheral Interface (bus), PWR = power, MAC = Media Access Controller, SERDES = serializer/deserializer [7]......	9
Figure 1-5.	Block diagram of the UT699 (AHB = Advanced High-performance Bus, APB = Advanced Peripheral Bus, IrqCtrl = interrupt controller, CAN = Controller Area network).....	10
Figure 1-6.	Block diagram of the P2020 dual core processor [8]. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.....	11
Figure 1-7.	Block diagram of the P5020 dual core e5500 processor [11]. It has significantly more features than the P2020 and provides more opportunities to explore test methods. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.....	12
Figure 1-8.	Example of RHBD compared to non-RHBD SEE sensitivity curves (example; in reality it may be impossible to extract the individual mechanisms without test structures, however they may have distinct signatures).	14
Figure 2-1.	Cross section vs. effective LET example for the L1 cache of the P2020 SOC [53].	17
Figure 2-2.	Example of proton SEE response in testing the Freescale MC7447A [17].	17
Figure 2-3.	SEU cross section vs. proton energy for IBM 45-nm and 65-nm SRAMs [68]. Image used with permission. It is unclear why the peaks do not line up. However, the general observation that the proton cross section can increase with decreasing proton energy is the key finding for this guideline.	18

Figure 2-4. The time structure of different types of targets in a device. The top, unhandled target, structure is expected for any part of the device not returned to a known state periodically.....	21
Figure 2-5. The range versus LET plot for 40 MeV/amu beams at TAMU [77] Image used with permission.....	24
Figure 2-6. System error in ideal fault-tolerant system as a function of flux (similar for beam delivered over an integration window for an algorithm) [27]. Credit to L. Edmonds at JPL for figure.....	25
Figure 2-7. Similar to Figure 2-6, this shows how the real device response differs from the ideal situation when an underlying error upset mode exists in a fault-tolerant system [28]. Image used with permission.....	26
Figure 3-1. i.MX6Q processor. Note that the cover is a heat spreader, which ensures good contact between the flip-chip die internal to the package and the external heat sink [113]. Image used with permission.....	34
Figure 3-2. A drawing of the mechanical structure of a flip-chip IC with a heat spreader. The sensitive region is the portion of the IC that is closest to the mechanical mount.....	34
Figure 3-3. The ion beams available at TAMU with 15 MeV/amu. Note that from the poly (p-phenylene terephthalamide)-aramid (Aramica) window the higher LET ions will be near the top of the Bragg Peak after 80 μm of Si [77]. Image used with permission.....	35
Figure 3-4. A device with heat spreader milled while on the board. The copper ring shows the heat spreader that remains. Photo used with Jet Propulsion Laboratory permission.....	36
Figure 3-5. Flow chart for the "static soak" test type. A periodic cycle of setting up static elements and later checking them for upsets makes up the main part of the test loop. Irradiation should be carried out during the inner loop.....	43
Figure 3-6. The actual algorithm executed for an FFT operation. Note that the inner loop (over n) includes four operations: complex loads (2) complex multiply-add (1), counter increment (1), and a compare/branch (1).....	45
Figure 3-7. Time structures for two types of tests performed on the Maestro ITC device. The black lines refer to the beam structure (which is the same structure in each case, zoomed out bottom) while the lower dual-colored strip in each region refers to the test algorithm duration [30]. Image used with author's permission.....	47
Figure 4-2. Example console display of the contents of register window 2 on a LEON processor [96]. The upper set of registers are the in, out, and local register sets (8 registers each) for register window 2 (specified by window invalid mask (wim) in the lower set), and the 8 global registers. The processor state register (PSR), trap base register (TBR), and condition register (Y) are also reported. This provides an immediate snapshot of the status of the running processor core.....	59
Figure 6-1. The block diagram of the UT699 (IrqCtrl = interrupt controller). (This is not the physical layout.) Image used with permission.....	68
Figure 6-2. The UT699 ready for testing. It is mounted on the GR-CPCI-UT699 evaluation board and the DUT is exposed by removing the lid. Photo used with permission.....	70
Figure 6-3. The cross section for SRAM cells was measured and compared to earlier data presented in [51]. Upsets in these bits will be corrected before they can impact the operation of the processor. Figure used with author's permission.....	71
Figure 6-4. Device cross section for register partial reset events on the UT699. Also shows that lower flux and lower fluence did not change the cross section [34]. Image used with permission.....	73
Figure 6-5. The block layout of the Maestro device. The main structure is the 49-tile array. It is surrounded by four DDR2 controllers, four XAUI ports, and a handful of additional peripherals. The tiles and peripherals are connected with five on-chip networks. Image used with permission [C. Rogers et al., "The Maestro Flight Experiment...", 2016].....	77
Figure 6-6. The FTB setup at TAMU. The beam pipe is directed at the DUT. The position of the DUT on the FTB is close to the center, and it almost does not fit in the beam line at TAMU. Image used with permission of Jet Propulsion Laboratories.....	78
Figure 6-7. Unthinned (top) and thinned (bottom) Maestro ITC test devices. The final thickness of devices is approximately 80 μm . Photos of backthinned devices used with permission of JPL.....	79
Figure 6-8. The general algorithm for test code used on Maestro. A single tile is chosen as the test master that monitors the test reporting from the other tiles. Photo used with permission of JPL.....	80

Figure 6-9. Effective sensitivity for detection of SEEs during Maestro ITC testing. Sensitivity is limited by the fluence that when delivered to the DUT results in approximately 50% of test tiles crashing [30]. (This is also Figure 4-1, duplicated to assist reading. Figure used with authors permission.).....	82
Figure 6-10. L1 data and L2 cache bit sensitivity of the Maestro tile processors. There is considerable spread in the data (especially L2) but these data are consistent with the SRAM sensitivity reported [44]. Figure used with author's permission.....	83
Figure 6-11. The register dynamic clobber cross section compared to the test sensitivity shows a response that is separated from the noise floor. Figure used with author's permission.	84
Figure 6-12. P2020 dual e500 core SOC block diagram [8]. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.....	88
Figure 6-13. Block diagram of the P5020 SOC showing significantly more complexity and more peripherals for examination as compared to the P2020 [11]. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.....	89
Figure 6-14. The general test setup for operation of the P2020 and P5020 for SEE testing.....	90
Figure 6-15. A P2020 with the heat spreader opened over the die. Photo used with JPL permission.	91
Figure 6-16. A P5020 with only a portion of the heat spreader removed over the die. This photo used with permission and provided by Jet Propulsion Laboratories.....	92
Figure 6-17. A P2020RDB test board at TAMU is shown. Note that when it is moved closer to the beam opening, the dry nitrogen line (blue/orange) blows directly on the opening in the heat spreader. This photo used with permission and provided by Jet Propulsion Laboratories.....	92
Figure 6-18. P5020 demonstration system. It can easily be seen that this is a general purpose circuit card enabling many different end uses. Image used with permission from Jet Propulsion Laboratory.	93
Figure 6-19. P2020 L1 cache results from [103].	94
Figure 6-20. Results for L2 testing on the P2020 [19].	95
Figure 6-21. P5020 L2 SEE sensitivity, showing similar sensitivity for '0' to '1' and '1' to '0' errors [53].	96
Figure 6-22. The register bit SEE sensitivity for e500 GPRs in the P2020 processor [103]. Testing was performed on both cores.	97
Figure 6-23. Crash sensitivity of the e500 cores in the P2020 processor (includes SEFIs and hangs, since both result in loss of operation of the core) [103].	98
Figure 6-24. Hang results on PowerPC and Intel tests for Power PC, MMX, and Pentium II processors. Note that testing performed on Intel devices used the Windows NT operating system, which is very complex and requires more of the device to function properly than the JPL PowerPC test software [1], with Heimstra data from [39]. Image used with permission.	100
Figure 6-25. Cache bit SEE sensitivity for three PowerPC processors with different feature sizes [101].	101
Figure 6-26. Cache results for more PowerPC processors [1]. Note that RAD6000 processors are 0.50 μm or larger.	101
Figure 6-27. Comparison of SEU cross section for the PowerPC7448 operated at 500, 1066, and 1600 MHz, some difference is observed with frequency, but it is limited to about 30% [1].	102
Figure 6-28. Cross section for upsets of data cache bits in Intel Pentium III microprocessors. Note that the lower line corresponds to the L1 Data & L2 100% case [40].	103
Figure 6-29. Cross section for SEFIs from various test algorithms [40].	104
Figure 6-30. The SEE response, device cross section vs. LET, of Leon3 (with FT, and with various amounts of SEE protection) showing that the unhardened setup saturates at about $5 \times 10^{-5} \text{ cm}^2$. [108]	105
Figure 6-31. Cross section for bit errors in an MGT channel on the Xilinx Virtex 5-QV [109].	106

Summary

The use of complex single and multicore processors with significant cache memory, on-chip peripherals, memory controllers, and high speed input/output (IO) that integrate many of the parts of a traditional computer system is becoming more common in space applications. Such devices are often referred to as system on a chip devices (SOCs), even though the term is used somewhat inaccurately due to the lack of analog and mixed signal subcircuits.¹ These devices are complex combinations of single- or multi-core processors with memory controllers, high-speed input/output (IO), and other peripheral structures that formerly would have been handled by off-chip resources. In the past the processors were tested for single event effects (SEE) separately, and the peripherals were often put into custom application-specific integrated circuits (ASICs) along with other resources required by the user. Performance and cost pressures have pushed commercial devices to incorporate many of the functional blocks into a single chip, an SOC. The NASA Electronic Parts and Packaging Program (NEPP) has been examining ways to perform SEE radiation hardness assurance (RHA) testing of these processor-centric SOCs to achieve reasonable understanding of their performance in space missions.

This document collects many of the findings of the NEPP efforts to perform SEE testing of SOCs into a guideline that can be used in evaluation of such devices. The goals are the following:

1. Identify the appropriate environments for general and specific SEE testing;
2. Explore and develop test methods for SOCs based on existing methods used in microprocessor and peripheral SEE testing;
3. Put the information into a useful format that provides relatively easy access to information that can assist test groups working on SOC SEE tests.

This guideline focuses on several distinct points, one of which is collaboration. Efforts that contributed directly to this guideline include testing of Cobham Gaisler components with direct support from Cobham Gaisler. Testing of Freescale SOCs included collaboration with BAE Systems (BAE). Testing of the Maestro Interim Test Chip (ITC) was supported by a collaborative test effort from several aerospace organizations including Boeing and Integrity Applications Incorporated. Many of the test methods discussed in detail come from test reports from Jet Propulsion Laboratory (JPL) and other radiation testing groups, including the test methods in the JPL Ground Based Microprocessor Test Guideline [1]. This guideline also benefits from collaborative review by several aerospace organizations.

This guideline begins with a general overview of SOCs, relevant radiation effects, and context with other testing. We then develop the need for radiation characterization testing, covering the types of test facilities that can be used and how test data are employed to calculate space event rates. The guideline then presents details about test preparation (focusing on hardware and software development) and approaches for ensuring that worst-case device conditions with respect to determining SEE sensitivity are invoked. We then cover actual testing and recommendations on how to successfully perform testing on SOCs at remote facilities. Analysis of test data and reported results are then discussed. Finally, we present results from testing conducted by NEPP, as well as review several key test reports on SOCs or microprocessors that concern topics relevant to this guideline.

¹ Traditionally, and by strict definition, an SOC is a device that needs only a power connection and perhaps a few IOs (such as a clock) in order to function, and usually include some amount of analog and mixed-signal circuitry. Modern devices that are called SOCs typically only include processors, bus switches, memory controllers, and high- and low-speed IO controllers. They often lack actual on-chip memory, physical layer IO support, and other functionality typically associated with a full SOC. Of particular note, almost all cell phone processors are called SOCs, but only fit the looser definition, even though in some cases they have extremely complex mixed-signal subsystems (on-chip wireless-fidelity [Wi-Fi]-radios). We use the looser definition in this document.

1.0 GENERAL OVERVIEW

Significant background and context material exists for this guideline. We will review many of these sources in this section to establish the basis for this guideline.

1.1 Introduction

This is a guideline for SEE radiation hardness assurance (RHA) testing of SOC. SOC use in spacecraft for NASA and other government organizations has expanded in recent years. This expansion is expected to continue as the technology transitions from commercial manufacturers to space users and as development of commercial devices into special radiation hardened versions proceeds. SEE RHA of SOC in this guideline focuses on terrestrial testing of SOC. An alternative approach of doing SEE RHA testing of the target device is to perform RHA by analysis with data on test structures used to feed the analysis, but it is expected that at some point in development a full test of the SOC will be required.

This document is intended to provide relevant information to highlight questions that must be answered regarding SEE testing of SOC. The primary goal is to provide a comprehensive study of relevant SEE impacts on modern SOC architectures, focused on how SEE testing should address them. The study of SEE focuses on the circuit impact, rather than the fundamental mechanisms. This guideline provides specific information on key elements of SEE testing of these complex devices, including determination of relevant beams and beam parameters, appropriate SEE sensitizing approaches, and presentation of test data and results.

This guideline is divided into motivation, approach, test methods, and sample data. The approach is defined by seven phases that are designed to ensure both a cost- and technically-effective test campaign that includes: manufacturer and user collaboration, the role and impact of peripheral circuits on overall SEE response, the impact of circuit and subsystem fault tolerance, the role and impact of radiation hardening by design (RHBD), multicore processor unique issues, the use of available general test methods, and sample test results. Test methods are discussed in detail, covering hardware selection and preparation, structure of algorithms, recommended tools, and more. The sample data provided covers several types of SOC (with each example covering different combinations of the seven phases) and shows some examples of reported test data. In each section key recommendations are gathered into a list for quick reference of specific findings.

This guideline was developed under the NEPP program. Because the SOC field is changing rapidly, and the SEE community is actively identifying topics of concern to the recommendations and findings in this guideline, readers are encouraged to also have wider knowledge of the general SEE community. Furthermore, we appreciate any feedback or collaborative opportunities to expand or update the material in this guideline as new information is developed. Please direct any inquiries or feedback to the author or the NEPP program.

1.2 Background Information

This guideline is an extension of NEPP efforts to test SOC devices and the existing NEPP guideline for ground based testing of microprocessors [1]. In this section we will discuss many of the influences that led to the need for an SOC-specific version and the relevant recent SEE test information for this work.

1.2.1 Why This Guideline

Many modern designers of spacecraft and landers desire increased processing capability on board. The Mars Science Laboratory (or Curiosity rover) uses the RAD750 processor [2]. Curiosity requires somewhat autonomous operations due to the light delay for commanding the rover (i.e., it is not real-time), with such algorithms using computational power to increase the performance of the rover. When executing the Drystone 2.1 benchmark, the RAD750 generally runs approximately 400 million instructions per second (MIPS) [3] (see Section 1.3.1 for some comparisons). However, new algorithms have been developed for

autonomous operations that require considerably higher performance than this. An example is the Rockster application, which requires on the order of a minute to analyze images of rocks with the RAD750, while implementation on a modern multicore processor can reduce this time down to a few seconds [4]. Rockster was designed to run on systems with closer to 10,000 MIPS.

Modern processors are increasingly resorting to complex architectures—including multicore processors and incorporating many of the high-speed peripherals on the processor die, where the goal is not so much to produce an entire system on a chip as to reduce issues due to latency and input/output (IO) operations. Furthermore, multicore architectures are allowing more capability on a chip without requiring increased clock speed or pipeline complexity. This is a trend that is becoming more mainstream as manufacturers move to 2.5D and 3D computer chips. Aerospace component manufacturers lag behind the commercial market but are now transitioning into this regime as the technology feeds into the development pipeline for RHBD and custom aerospace devices.

This guideline is also intended to provide a framework to enable collaboration with various industry manufacturers and various testing groups. Collaboration can bring forward alternate solutions for the challenging problems confronting SOC testing.

By presenting this guideline, standard methods can be developed and their benefit judged for potential use in testing of SOCs for user programs.

1.2.2 Target SOCs

As discussed above, we will denote these complex processor/memory controller/peripheral devices as SOCs (in-line with cell phone and modern laptop and computer processors), where multiple circuits are put onto the same IC in order to reduce latency and complexity of interconnects between devices while leveraging decreased feature sizes, which allows a considerable amount of circuitry on a single IC. These SOCs include one or more processor cores, memory controllers, and high- and low-speed IO circuitry. We are only concerned with devices that are entirely digital and do not cover some types of SOCs where the full system is incorporated on a single IC. We also deviate from cell phone processors somewhat here because they include various analog circuits.

For this guideline we are focused on two particular types of SOCs. These types are not necessarily complementary, and do not lend themselves to identical test methods. They are joined here because they are the two types of the most interest for space system developers. The first is the RHBD, high-reliability-type SOCs that are currently being produced for aerospace applications such as the Atmel 697E, the Cobham Gaisler UT699, and the Maestro 49-core processor devices [5,6,7]. The other type is commercial high performance SOCs that may be selected for space programs that can accept fairly high risk in order to have high performance. Example devices of relevance are the Freescale P2020 and the P5020. The P2020 has been used as the basis of a flight computer made by Space Micro, the Proton 400k-L™ [8,9,10]. The Freescale P5020 is a higher capability processor than the P2020, with 64-bit processing cores [11]. This device is built from library elements that are likely to be used in future space RHBD microprocessors from BAE Systems [12].

The aerospace application devices (e.g., Maestro) are interesting to examine for radiation effects because they show many of the key features being discussed in the SEE community, such as single-event transient (SET) sensitivity and angular structure of response of RHBD circuit elements. The commercial devices are also interesting to examine, but in their case the reason is because the hardware preparation is more challenging and the devices have significantly higher processing capabilities and IO speed, making for more interesting test algorithms. Many aspects of test development and performance are similar between these devices, but their differences also make things interesting and mean that some types of testing or development are more appropriate to one type than to the other.

1.3 Key Issues

There are some key research areas that apply to SOCs. Some of these will be covered in more detail later in this background section. We briefly discuss each of the topics here to introduce these key research areas.

1.3.1 Device Performance

A large number of current space systems use the R600 or some form of the RAD750 family that can provide between 266 and 400 Dhrystone 2.1 MIPS [3] depending on the host technology. Although the Dhrystone MIPS synthetic computing benchmark program is not necessarily a good way to evaluate absolute performance of microprocessors, we can use this information to perform a general comparison. Modern microprocessors can perform at upwards of 200,000 MIPS [13]. Although the latter require power that is not feasible on a spacecraft (more than 100 W) they do highlight a significant gap. This is further shown by considering the Freescale P2020 processor, which can operate on only a few watts of power, a level that is viable for space use. The P2020 can achieve about 4,000 MIPS [14], which is about ten times more than the RAD750. The Apple A5 processor in the iPhone 4S [15] achieves about 5,000 MIPS.

Modern commercial devices, especially those with low-power capability, are available primarily as SOCs. This architecture choice enables the devices to achieve higher performance at low power. As a result, these devices are generally more interesting to aerospace applications.

This guideline explores several types of SOC, which span different performance regimes to include the Cobham Gaisler UT699, which provides about 90 MIPS and the Maestro microprocessor that can provide 45 giga-operations per second (GOPS). (Note that instructions per second is not as useful as operations per second, since modern processors often include instructions that perform multiple operations. Thus, GOPS and million operations per second (MOPS) are better terms, but MIPS is still commonly used for historical reasons.)

Raw device performance must also be balanced with power consumption. Spacecraft are on limited power budgets, with some types of missions restricted to as little as 10 W of power. Another useful parameter to explore for such systems is operations per watt (GOPS/W or MOPS/W). Although it is outside of our scope to discuss the ranges appropriate for missions, in general larger values of GOPS/W are better as the system can perform more operations on a given power budget.

SOCs directly studied in this guideline involve feature sizes down to 45 nm. This guideline is primarily focused on the effects known to impact devices at this feature size and larger. In particular we are interested in the impact of multi-bit effects, angular effects on RHBD and non-RHBD circuits, sensitivity to transient signals, and scaling of critical charge.

1.3.2 Setup and Operation

Setup and operation of test equipment are key issues for SEE testing of SOCs. In this guideline we explore hardware and software setup. For hardware, we cover test board selection, hardware setup at SEE test facilities (including general problems with biasing of boards), and methods for interfacing with test boards. For software setup we discuss several topics. The first is the available hardware and software tools for interfacing to a device under test (DUT) through support interfaces. We also discuss manufacturer tools for interacting with test boards. Finally, we discuss in detail the features required of test software for extraction of SEE performance characteristics.

Setup details also include selection of appropriate beam facilities and beam parameters. This includes examining how the available hardware mates to the available beam facilities to enable collection of the key data necessary to enable reliable rate calculations for SEE on the DUT in application environments.

1.3.3 Test Approaches

Several different approaches can be taken to ensure that a complex device is operating in a SEE sensitive mode and to facilitate characterization and ensure collection of statistically significant SEE data. The methods break into the following: utilize the resources of the device to sensitize the device; monitor the operation of the device at the pin level and look for deviations from expected IO patterns; and utilize test hardware and debugging interfaces to verify internal operation.

The second method is very similar to the golden chip method that has been known to the SEE community for a while. A basic description of this type of test can be found in Koga et al. [16]. Unfortunately, this type of test is of limited benefit when the circuit becomes complex and due to either fault tolerance (FT) or other mechanisms becomes non-deterministic. The cache operations and parity protection alone render this method essentially impossible to reconcile.

The remaining two methods are both advocated here. Later in this document we provide significant depth regarding methods to allow a microprocessor to test itself. These are based largely on Irom [1], and Guertin and Irom [17,18]. We provide some specific examples of how test hardware and debugging interfaces can be used to explore complex SOCs. This is done via hardware used to operate a port defined by the Joint Test Action Group (JTAG), Debug Support Units, and other interfaces such as Maestro's protocol mode [19,20]. Having a device test itself can be very useful for exploring detailed nuances of device operation. However it is also very tedious and potentially difficult to get the test code to work properly. The alternative of using built in hardware debugging support is a very powerful approach that is becoming easier to perform as devices become more complex and manufacturers put increasing amounts of on-chip self-test hardware in their devices.

1.3.4 Packaging

Preparation of test devices for heavy ion exposure generally includes modification of the DUT to enable heavy ions to penetrate to the sensitive part of a DUT. In this guideline we explore packaging problems and preparation of devices to enable testing.

The devices explored generally fall into three categories. The first type of package is one that can simply be opened up to expose the top surface of the die. The second is one where the device die is directly exposed but the device is a flip chip, which requires some thinning of the die or use of only long-range ions because the active devices are on the bottom. The third type of package is one with a heat spreader which must be removed in order to expose the IC.

The packaging problem is further complicated by the mounting of the DUT. We examine how DUT mounting hardware can reduce the cost of test devices but also limits the angular capabilities of the test system due to shadowing by the mounting hardware.

1.3.5 Thermal Issues

Modification of device packaging changes the thermal behavior of the DUT. We examine the difficulties involved in preparing devices including modern heat spreaders. We also review methods used in earlier work involving thermal issues in microprocessors.

1.3.6 Operating Frequency and SET

Digital combinatorial circuits, for the most part, have been demonstrated to be increasingly prone to the generation of single event transients (SET) as clock frequency increases, due to the increased probability of latching transients (SETs) on an active clock edge. There are a few studies of how this affects technology, latches, and microprocessors [21,22,23]. The general concept is shown in Figure 1-1. Here we have three traces, one showing when data must be valid for a clock-edge triggered latch, the next showing our example clock, and the final one showing what an SET pulse might look like. Because the SET pulse is short relative

to the clock period, it only registers during the setup and/or hold period of a latch. Thus, under the clock that is shown, there would be no effect. However, if the clock is doubled (as indicated by the dashed pulse), then the shown SET pulse would be able to affect the system.

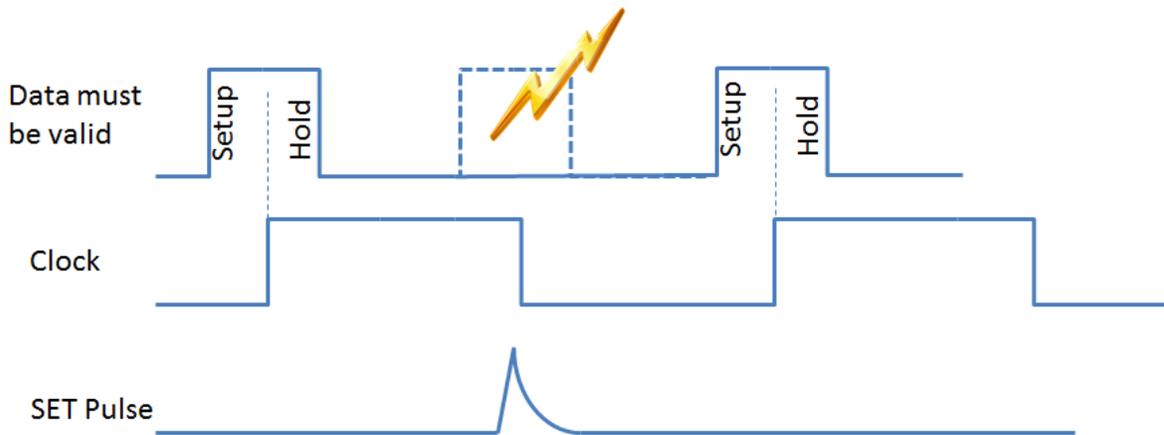


Figure 1-1. Diagram showing how increased frequency can contribute to higher rates for upset in digital circuits. What is shown is the basic operation of a clock-edge triggered latch. The data must be valid where the upper portion is high which is only when the clock is transitioning. If an SET occurs between these regions it does not affect anything. However, if we were running at twice the clock rate then the SET would have disrupted the valid data during a window where it must be valid. Doubling the frequency effectively doubles the SET sensitivity.

Clear evidence that SET sensitivity increases with clock rate can be found in Benedetto et al., [23] and is shown in Figure 1-2. The behavior in this figure is observable because the basic sensitivity of storage cells (in this case the Dual Interlocked Cell, DICE) is much lower than the SET sensitivity of the circuit being tested. Thus, the cells themselves are not upsetting, only the SETs are contributing, and as discussed in Figure 1-1 the sensitivity to SETs is directly proportional to frequency. In this case it may be an SET on the input to the DICE latch, or on the clock circuit that leads to the observed sensitivity.

The frequency dependence observed in commercial devices has largely been of lower level than the sensitivity of latches and memory cells. This has given rise to frequency increases on the order of 30–50% of the low frequency sensitivity when applied to frequency increases of a factor of three or more. Because of this behavior, we do not expect significant frequency effects in evaluation of commercial devices throughout this document. However, it is clear that frequency dependence may be important in RHBD devices such as the Maestro microprocessor and the Xilinx V-5QV field programmable gate array (FPGA) [24,25,26,27,28,29,30] and thus it is an important variable in test planning, though it remains elusive to find SOCs affected by this. Further, it is possible that frequency effects in commercial devices may become more important as the critical transient width prediction in Figure 1-3 clearly should be impacting circuits [20].

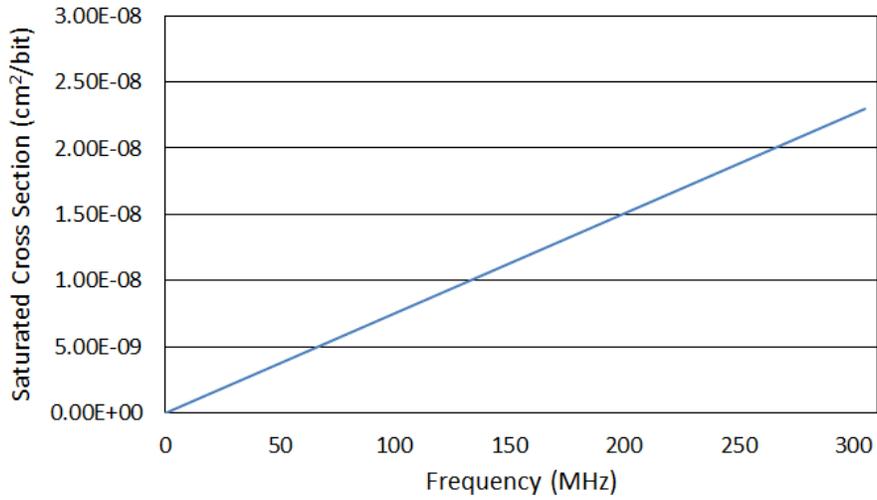


Figure 1-2. Sensitivity of 0.18 and 0.25 μm DICE latches to SETs as a function of frequency. Note that DICE latches in these feature sizes are not expected to be sensitive to direct upset of the cells in this cross section range, after [23].

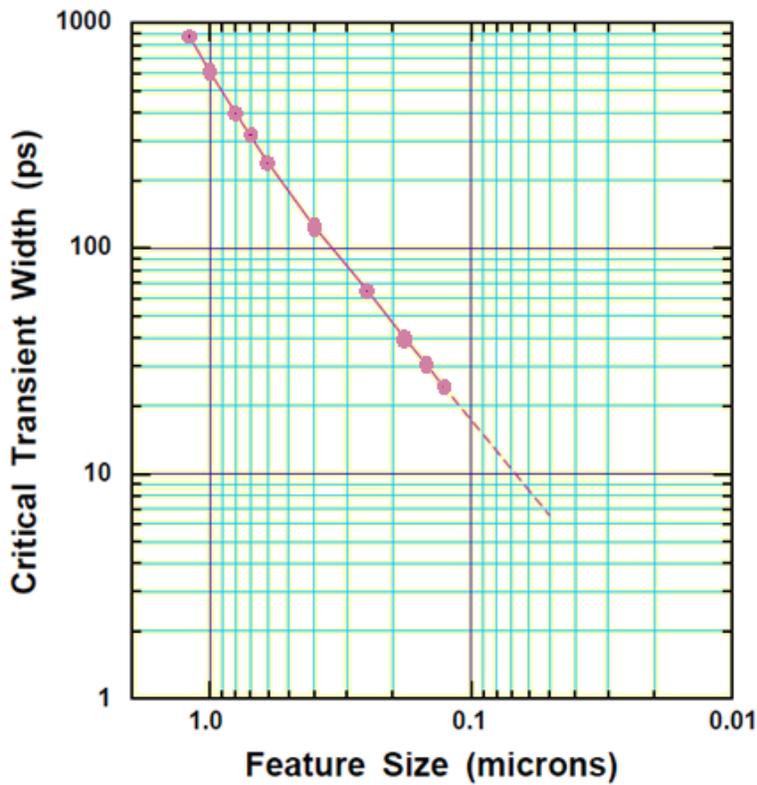


Figure 1-3. Critical transient width as a function of feature size suggests that submicron technology nodes, such as 90 nm, 45 nm, and below, are expected to have significant SET sensitivity, after [24].

1.4 Background Information

This guideline can be thought of as an extension of the existing SEE test standards and microprocessor test guidelines. These include the ASTM International (see Acronyms) guide for heavy ion single event

phenomena (SEP) [31], and the JEDEC Solid State Technology Council (see Acronyms) guideline for SEE from heavy ions [32]. A guideline for SEE testing of microprocessor devices is also available [1]. In addition to this background literature, actual reports on test efforts provide a very important resource for developing methods for SEE testing.

In this section we examine the existing literature as is relevant to the goals of this guideline. We focus on developments targeting microprocessors, complex devices, and some high-speed IO. The material discussed here is only for background purposes for the development of the guideline. The particular results of the discussed efforts, if relevant to test methods and data presentation will be discussed further in the radiation characterization needs (Section 2), or in the sample results (section 5).

1.4.1 Existing Test Guidelines

The ASTM International and JEDEC guidelines are useful for general SEE test considerations [31,32]. A general microprocessor testing guideline from Jet Propulsion Laboratory (JPL) is discussed in Irom (2008) [1]. There is also an SEE testing guideline from Sandia [33]. These guidelines must be extended for SOCs because of difficulties specific to SOCs. In general, following the guidelines listed will result in a good general test, but will likely produce data sets of limited value for SOCs. We extend the guidelines by showing how to expand the microprocessor test methods to SOC architectures, and by providing a review of the microprocessor test guideline in the light of changes that are explicit to these advanced devices, their modern architectures, and their process nodes.

1.4.2 SEE Test Approach on SOCs

The test approach on SOCs examined in this guideline is consistent with several publications by the JPL radiation effects group [17,18,19,20], [30,34,35,36]. We explore the presented approach in the context of other approaches developed by Koga [37], Bezerra [38], Hiemstra [39], Howard [40], Czajkowski [41], Rech [42], and Quinn [43].

The test approach derives from microprocessor studies, but it is analyzed in the context of the difficulties involved in testing SOCs. This includes hardware selection, IO methods, on-board software, external and internal debugging tools, and standard data analysis considerations.

1.4.3 Boeing Maestro ITC

Here we provide support information on the Maestro microprocessor. The material in this section can also be found in Guertin et al., 2012 [30]. The Onboard Processing Expandable Reconfigurable Architecture (OPERA) program has produced a 49-core multicore processor based on the Tiler architecture, named Maestro [7,20]. The Maestro ITC (Interim Test Chip) microprocessor, designed by Boeing Solid State Electronics Development (SSED) and fabricated in the IBM 90-nm bulk technology (9SF), is designed for space applications and was supported by the joint Defense Threat Reduction Agency/Defense Advanced Research Projects Agency (DTRA)/(DARPA) Radiation Hardening by Design (RHBD) program [44]. Intellectual Property (including the Maestro processor) from the OPERA program is available to all United States Government (USG) agencies for space use.

The Maestro ITC 49-core many-core microprocessor is shown in Figure 1-4. This figure is a rough functional block layout indicating the 7-by-7 array of tile processors and the various support peripheral interfaces. The cores (or “tiles”) are connected by a grid of five on-chip networks. These are the Memory Dynamic Network (MDN), the Tile Dynamic Network (TDN), the User Dynamic Network (UDN), the Input/Output Dynamic Network (IDN), and the Static Network (STN). These provide communication resources whereby tiles communicate with neighbors. The control mechanisms for these networks enable messages to be passed to the appropriate targets, including peripheral controllers.

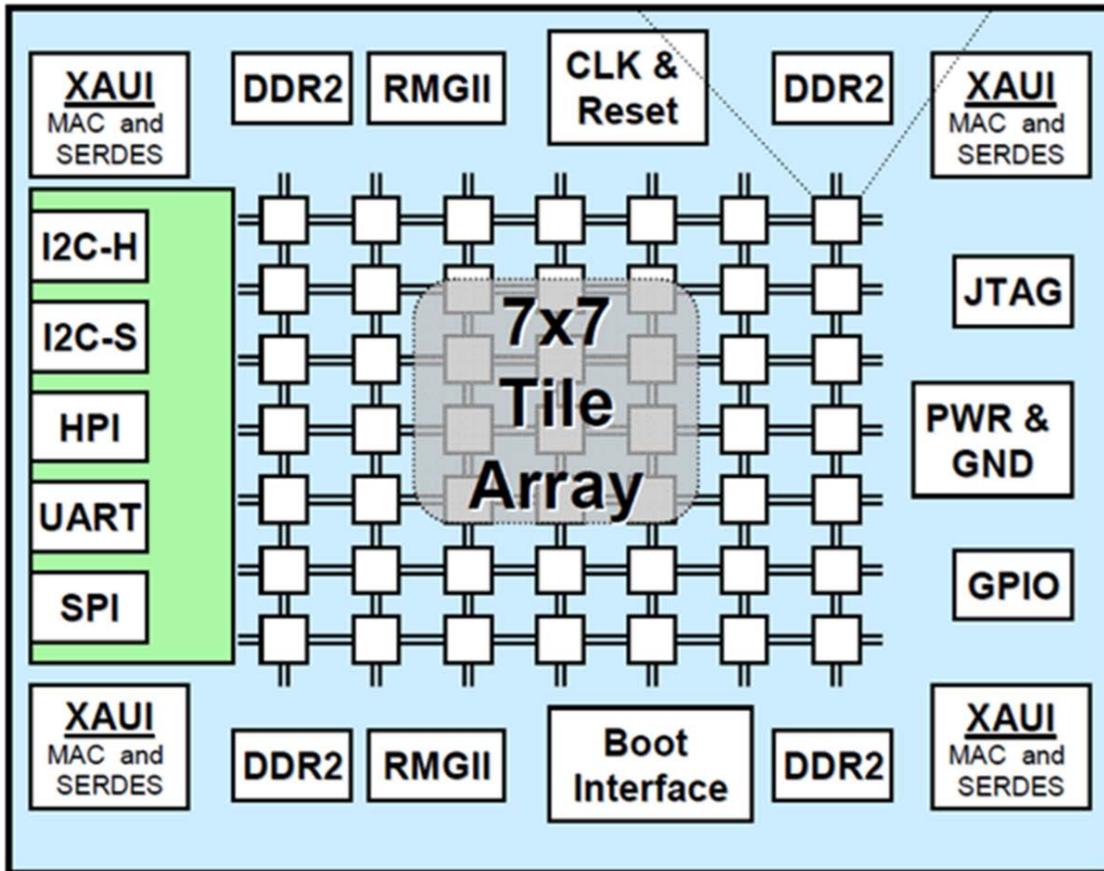


Figure 1-4. The block diagram of the Maestro 49-core many core microprocessor. Image used with permission. The center of the diagram shows the 49-core tile array, which is surrounded by various peripheral systems. The tiles are supported by five on-chip networks. DDR2 = double data rate 2, XAUI = X [for 10 Gigabit] Attachment User Interface, GPIO = general purpose input/output, RMGII = Reduced Gigabit Media Independent Interface, CLK = clock, I2C (-h/-s) = Inter-Integrated Circuit interface high-speed and standard, HPI = Hardware Platform Interface, UART = Universal Asynchronous Receiver/Transmitter, SPI = Serial Peripheral Interface (bus), PWR = power, MAC = Media Access Controller, SERDES = serializer/deserializer [7].

Tilera was provider of the Intellectual Property that the government purchased and redesigned using Boeing's 90-nm RHBD ASIC library, to become the Maestro ITC processor we tested. Note that the redesign was significant and a floating point unit (FPU) capability was added among various other changes from the original commercial device. The tiles in the Maestro ITC are constructed with 8 kB L1 data cache, 8 kB L1 instruction cache, and 64 kB L2 cache. The Tilera architecture has been used to produce commercial computers. The tile processors themselves are based on the "Raw Architecture" [45]. This device was developed to support high performance general space computing applications. As such it was designed with RHBD methods. However, due in part to size constraints, the tile caches were implemented with SEE-soft ARM (see Acronyms) (formerly Artisan) static random-access memory (SRAM) cells which were then protected with parity in both of the L1 caches and byte-level single-error correction, double-error detection (SEDED) error detection and correction (EDAC) circuitry, which protects every eight data bits with five check bits. (Note that the L1 tags are protected with EDAC, not parity.) The caches are designed to be fault tolerant (FT).

The Maestro ITC and its ancestor Tilera devices have already been used to examine space applications. A fast Fourier transform (FFT) algorithm and a parallel-processing image-analysis application (CRBLASTER [see Acronyms]) were examined for benefit of utilizing multiple cores on Maestro [46].

Test structures for the Maestro ITC have been evaluated for SEE phenomena over the last several years. Logic gates and flip-flops on this process have been examined for radiation effects [25,26]. The SRAMs used to build the Maestro ITC were characterized in separate SRAM chips [47].

1.4.4 Cobham Gaisler UT699

The UT699 uses the LEON 3FT, Sparc V8 element of the Cobham Gaisler IP Library [6,48,49]. The library elements can be combined to form SOC applications that combine many of the resources required for computer systems. The UT699 architecture is shown in block-diagram format in Figure 1-5. The LEON 3FT block of the UT699 contains an 8 kB L1 instruction cache and 8 kB L1 data cache. The UT699 includes but is not limited to an FPU (in the LEON 3FT block), a memory controller, a PCI (Peripheral Component Interconnect) controller, an Ethernet MAC, and a SpaceWire port.

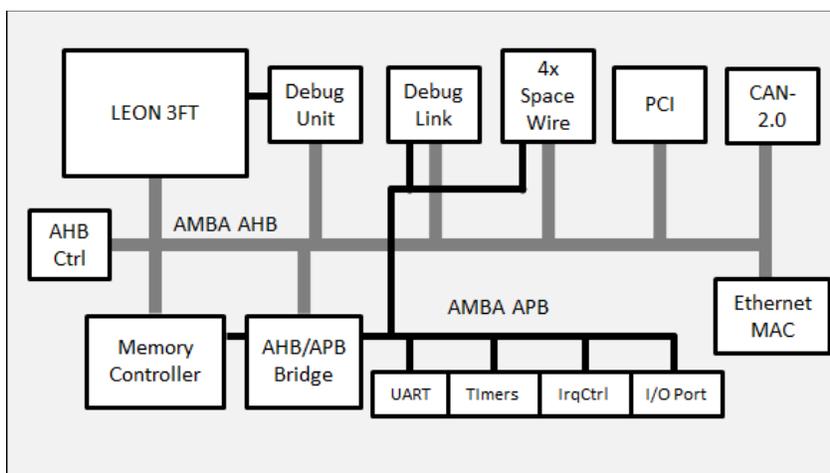


Figure 1-5. Block diagram of the UT699 (AHB = Advanced High-performance Bus, APB = Advanced Peripheral Bus, IrqCtrl = interrupt controller, CAN = Controller Area network)

RHBD is implemented in the UT699 to improve both the TID hardness and the SEE response. For this work we concentrate on the SEE response. SEE hardness is accomplished by building the UT699 out of elements of Cobham's 0.25- μm RHBD library. The UT699 is composed of 203,120 SEE hardened SRAM and 30,877 SEE hardened flip-flop (FF) cells. The SRAM single event upset (SEU) threshold is about 8 $\text{MeV}\cdot\text{cm}^2/\text{mg}$, while the FF threshold is approximately 54 $\text{MeV}\cdot\text{cm}^2/\text{mg}$ [50]. The register file SRAMs are protected by EDAC, and the cache SRAMs are protected by parity and replacement.

Previous testing by Cobham Gaisler includes predictions for upsets in the UT699 in a geosynchronous Earth orbit (GEO) [51,52]. They established rates of less than 1 event in 1×10^7 years for overwhelming the FT SRAM cells, when refreshed once per 100 ms, and 1 event in 7.4×10^4 years for events in the FF cells (which is an upper limit—actual sensitivity will depend on how many FFs are important for an application). The FF cell sensitivity establishes the threshold at which other SEEs are relevant for application sensitivity.

1.4.5 Freescale

The material in this section was developed for a report for the NEPP program [53]. The Freescale P2020 and P5020 are of interest to this guideline because they offer multicore microprocessor environments in the 45-nm feature size range, with the former being a 32-bit device while the latter is a 64-bit device. These devices are expected to exhibit SEEs consistent with cutting-edge device behavior. Further, as multicore SOCs they provide the opportunity to expand examination of multicore and on-chip peripheral test methods. The PowerPC processor architecture used in Freescale devices is currently flown on RAD750 computers from BAE [3], and they are of interest for future space processors from BAE [54].

A block diagram of the Freescale P2020 is shown in Figure 1-6. It is a dual-core e500-v2 Freescale SOC. The e500 cores each have a 32 kB I-Cache and a 32 kB D-Cache. A single 512 kB L2 cache is available for general use or can be configured as an on-chip SRAM. The additional peripheral resources on the P2020 (such as the gigabit Ethernet and serial Rapid I/O ports) are of interest in terms of establishing test methodology for these ports.

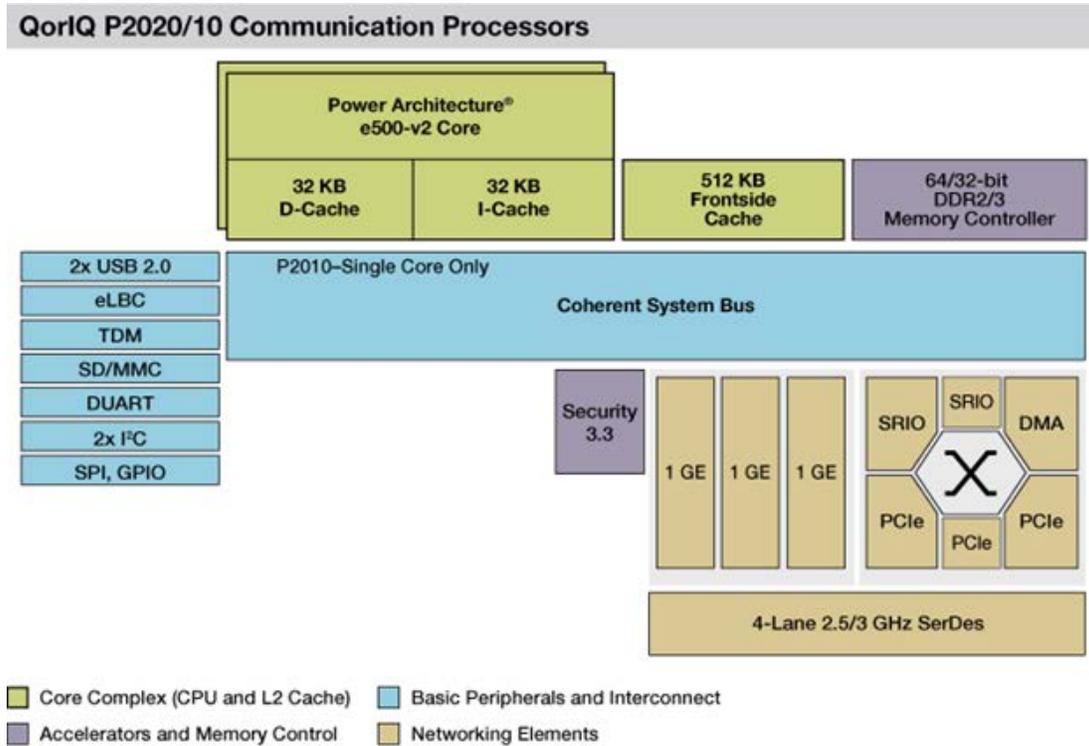


Figure 1-6. Block diagram of the P2020 dual core processor [8]. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.

A block diagram of the P5020 processor is shown in Figure 1-7. The P5020 is of interest because the e5500 cores represent the more modern core architecture where the processor is a 64-bit architecture. The P5020 is an architectural successor to the P2020, with somewhat different processing capabilities and more peripheral capability. The P2020 offers a simpler, smaller scale platform in which to develop most of the test code before porting on the more complex P5020 systems. These two systems, taken together provide an interesting case study that we use throughout this guideline for examination of several of the primary thrusts of the guideline across multiple related devices.

1.4.6 Other Devices

SEE testing of microprocessors and SOC's have focused largely on devices and architectures that have been used in space missions. There are, however, some key technologies that are potentially very important for future use. The leading candidates for further exploration are ARM-based devices and modern Intel and AMD devices. This guideline also finds benefit from examining FPGA efforts.

An example of an ARM-based SOC is the A6X processor in the 4th generation iPad [55]. Mobile devices are particularly interesting for this test guideline because of reduced problems with heating and low power usage, both of which make them more interesting for space users. This device has two ARM Cortex A9 cores running at 1.4 GHz, and the PowerVR SGX 554 quad core graphics processing unit (GPU) [56]. There is considerable interest in running this type of device in space, including comparison of processing

power between the iPhone 4S and the computing system on the Curiosity Rover [57], and the use of the iPhone 4 on the International Space Station (ISS) [58].

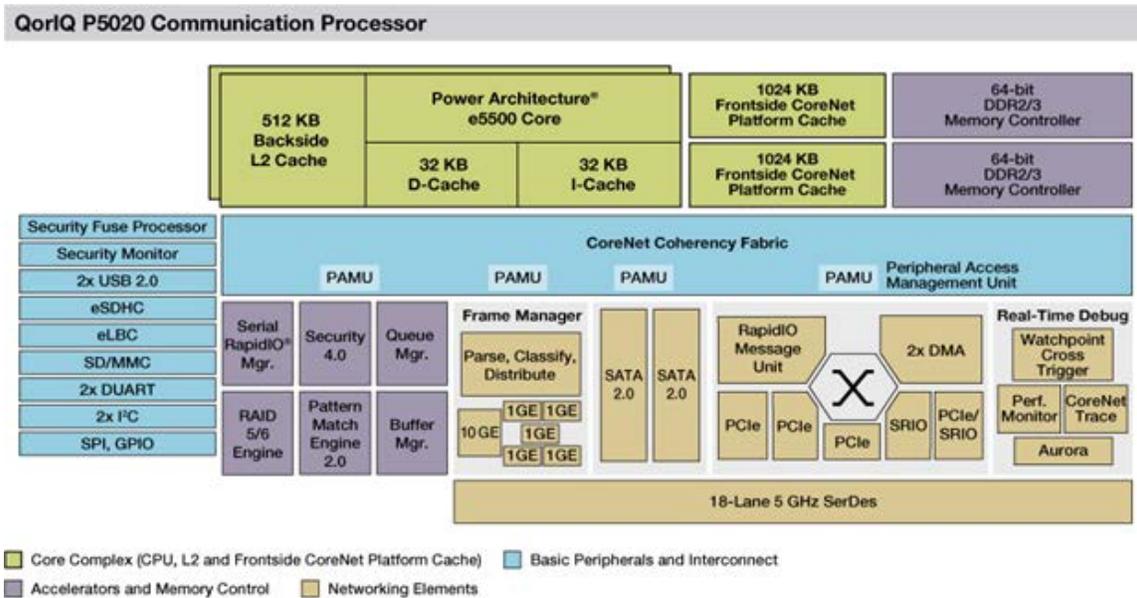


Figure 1-7. Block diagram of the P5020 dual core e5500 processor [11]. It has significantly more features than the P2020 and provides more opportunities to explore test methods. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.

Although they have been studied in the past and are generally very difficult to test (likely to be more difficult than the Freescale devices discussed in this guideline), SOCs from Intel and AMD should be considered as important potential SOCs for space use. Recent improvements in power consumption moved Intel closer to the power level of mobile ARM devices by mid-2013 [59].

The intersection of RHBD and FT can be examined in FPGA studies, especially the examination of the Xilinx Virtex 5QV which has been examined for SEE Sensitivity [27,28,29]. The referenced work discussed various test data produced on this device as well as the potential benefit of a “tri-flux test,” which can be used to establish that an observed event type is partially due to overwhelming the fault tolerance.

Another important aspect of FPGA study is soft- and hard-core processor implementations in FPGAs. There is considerable interest in the study of embedded ARM processors such as in the Xilinx Zynq™ series FPGAs. Similarly, there is increasing interest in the use of devices such as the Xilinx pico-blaze™ and similar soft-core processors and how to build SEE-robust systems with these devices using both hardware and software approaches [60].

1.5 Seven-Phase-Approach Guideline Development

The guideline is based on seven focal points, to ensure a comprehensive and cost effective test approach for SEE testing of SOCs. In this section we review each of these points.

1.5.1 Manufacturer and User Collaboration

SOC testing (and similarly microprocessor testing) is confronted with difficulty on two fronts. First, manufacturers hold essential knowledge of how to understand the data coming from a test setup and how to interpret results related to the structure of their devices. Second, application users are the best resource to understand the system impact of upsets and how to best test devices to ensure the right data are collected

to support flight applications. It is essential to try to bring both of these groups to the table when determining the SEE sensitivity of an SOC.

1.5.2 Peripheral Evaluation Approach

Microprocessor testing has been carried out and reported for more than two decades. The key to understanding the sensitivity of SOCs is to understand the SEE sensitivity of the interconnect systems, and all the components on the device. Of course this includes the processor. But it is the networks and peripherals that extend SEE examination of SOCs the most.

1.5.3 Impact of Fault Tolerance on SEE Characterization

FT is used to improve the operation of essentially all devices. Commercial devices are often built with FT, though it has been observed that many users disable the FT because it impacts the performance of the device. Furthermore, users are also interested in FT beyond the strict hardware resources—including the use of multiple cores to provide increased FT in a system. Fundamental FT hardware structures must be examined to ensure that they work and that their impact in testing is understood (as FT systems are often overwhelmed in ground test conditions while they would not be in space applications). Ultimately, the correct handling of FT questions is highly important, but at a minimum the purpose-built hardware systems provided by a manufacturer must be verified.

1.5.4 Impact of Radiation Hardening by Design on Characterization

RHBD construction, especially of special-built parts for space use, requires special test methods. RHBD devices require more consideration in the ions used for testing because they are expected to have complex SEE response curves. Typically, non-RHBD devices have a simple SEE response. Non-RHBD devices typically have a very low linear energy transfer (LET) threshold for some event types, which then saturates at such a high level that no other sensitivity will be observed during testing or contribute to space rates. This is not the case for RHBD devices where multiple mechanisms may turn on at different LETs but do not mask other mechanisms. This is exemplified in Figure 1-8. RHBD devices are also expected to have greatly improved static element SEE sensitivity, which means that SET may become more important. Notice that the non-RHBD device behavior saturates very quickly (at low LET), and at very high cross section (usually close to the die area) resulting in a huge number of events when testing at reasonable fluence levels ($\sim 1 \times 10^5$ particles/cm²).

1.5.5 Impact of Multicore Design on Characterization

Multicore devices have become the norm in all modern computing applications. Even cell phones are using dual and quad core microprocessors. Thus, although the space community is just now encountering a few multicore devices, they are definitely expected to become more common in the future. Mechanisms for multicore communication, and the ability to test the various cores in a multicore, make up the primary considerations for multicore SEE testing discussed in this guideline. Communication is primarily handled by direct communication channels, cache and memory coherence, or both.

1.5.6 Use of General Test Methods

The use of existing SEE characterization methods should be investigated to determine their applicability to support SOC characterization and tailoring, as appropriate. All the various types of test methods of importance to SEE testing are covered in this guideline. This includes everything from test board selection to analysis methods and reporting.

1.5.7 Investigation and Analysis of Prior Sample Testing

Some sample tests have been performed in direct support of this guideline. Additionally, many other test efforts conducted by other members of the radiation effects community have been examined. These tests

provide concrete examples of the effectiveness of various test methods and recommendations discussed in this guideline.

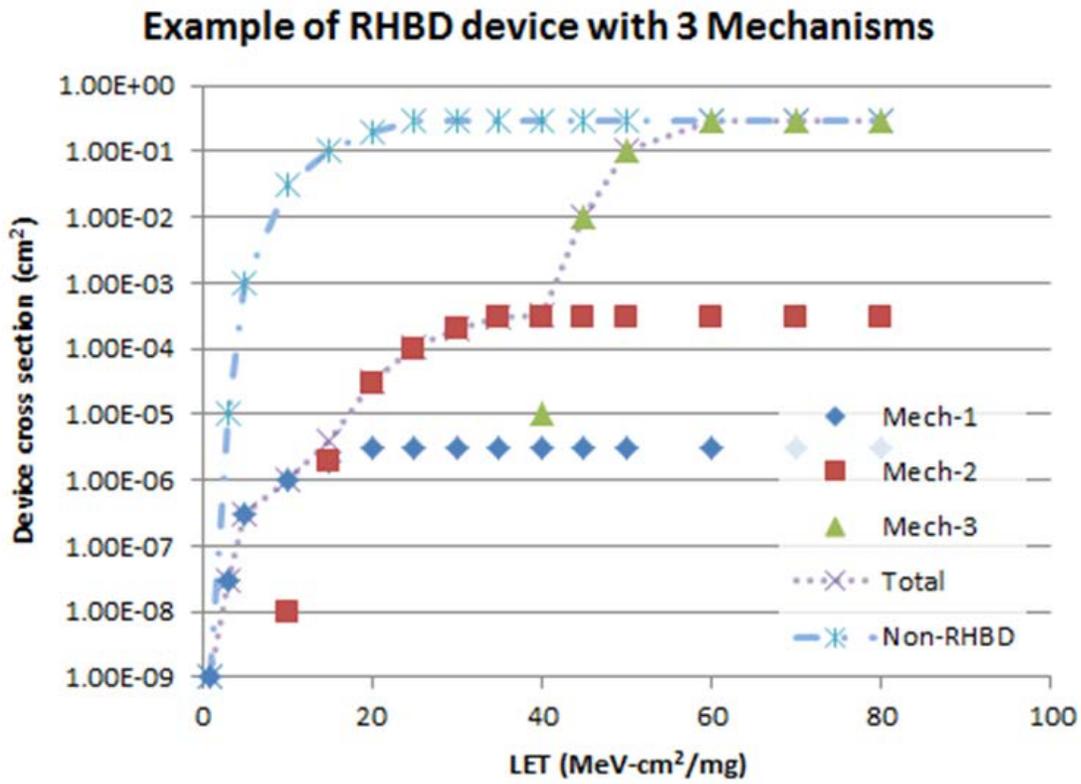


Figure 1-8. Example of RHBD compared to non-RHBD SEE sensitivity curves (example; in reality it may be impossible to extract the individual mechanisms without test structures, however they may have distinct signatures).

2.0 RADIATION CHARACTERIZATION NEEDS

This section discusses radiation characterization of SOC devices. Characterization data can be used, in part, to support a qualification program depending on end user requirements. Performing radiation characterization on a device covers the determination of the relevant upset modes of a device, and obtains sensitivity information for all modes against radiation types relevant to a program. In this section we discuss these standard elements, then examine challenges specific to SOCs of interest to this guideline. We briefly discuss the impact of test and application software on upset data. And finally we finish with a set of recommendations for establishing an appropriate characterization effort.

2.1 Standard Testing Needs

This section provides a review of the standard information necessary to achieve adequate detail on the target device to enable radiation characterization. We focus on the standard radiation data, including proton and heavy ion SEE sensitivity, and how the target device may alter the requirements on this data.

2.1.1 Data Needed

In order to establish the radiation performance of modern SOCs, it is necessary to build a data set sufficient to enable rate calculations for program use. There are four elements that must be considered to ensure appropriate characterization is achieved for a given program. These are:

1. mission environment,
2. SOC details such as RHBD construction,
3. program application of the SOC, and
4. the tool that will be used for rate calculations.

In this section we will briefly discuss what the test engineer must examine regarding each of these elements.

Mission environment is a standard element of IC radiation characterization. Several guidelines and test standards can help guide planning efforts [31,32,33,34]. The environment must be defined well enough to be merged with the rate calculation tool in order to produce rates. Generally speaking, for an Earth orbiter, this information is the standard orbit parameters of inclination and altitude. For most programs a specific environment will be specified. The reason to be aware of this information up front is because test data may be tailored to the environment.

Details of the SOC are important for establishing the approach to collecting radiation data. The device's mechanical requirements and thermal limitations forced by the planned test hardware are the biggest drivers of test setup issues. It is often cost and complexity prohibitive to create custom hardware for the operation of modern SOCs. But it may also be impossible to achieve a reliable test setup using existing demonstration equipment due to the inability to sufficiently expose the SOC for heavy ion exposure. In addition to these issues, it is also very important to be aware of built-in fault tolerance and any RHBD elements of the SOC. Further, for modern SOCs the feature size may be small enough to require consideration of proton direct ionization. Also, based on process type certain types of SEE such as Single-Event Latchup (SEL) may be important considerations. From a more functional perspective, the test group must also take stock of the SOC's processor core type, hardware interfaces, internal bus architecture, and any on-chip memories.

The actual intended use of the SOC by the relevant program is very important for establishing an appropriate characterization approach. Although this guideline is intended for general radiation characterization, and the desire would be to contain the majority of programs within the decisions made as to how to proceed with characterization, the program usage can be so important that it is at least instructional to discuss how program use can impact test planning. First, the program may not intend to use all of the elements of the

device. Second, in the event of fault-tolerant device elements, programs may not actually implement them.² Third, although test algorithms usually provide conservative estimates for device sensitivity, it is possible that actual flight algorithms will perform differently and the difference may be critical to how the program uses the device.

There are a couple of options for rate-calculation software, such as CRÈME [61,62], SPENVIS [63], and SpaceRad. JPL also uses custom software that can support the Edmonds method for rate calculations [64]. Although the environments supported by the tools may vary, the input data they require for rate calculations is the same. The next subsections provide details about heavy ion and proton testing, and touches on worst-case conditions for upsets.

2.1.2 Heavy Ion Test Data

Heavy ion test data represent the sensitivity of a device to the passage of heavy ions through sensitive regions of the device. For heavy ions, all relevant event types are identified along with their relevant normalizations (e.g., number of bits). Then each event type is monitored and occurrences counted during (and after) radiation. For each tested condition and event type, t , the quotient of the number of events and the applied beam per unit area is used to construct a cross section, as in Equation (2-1).

$$\sigma_t = \frac{N_t}{\Theta_t} \quad (2-1)$$

For heavy ion testing the primary data set required for each event type is the relationship between cross section and linear energy transfer (LET). An example of this response is shown in Figure 2-1. During rate calculations it is important to collect angular data to determine the degree to which the device follows the cosine law regarding effective LETs (LET_{Eff}). The cosine law is given in Equation (2-2) and relates LET_{Eff} to the normal incident LET and the angle of incidence, θ , relative to normal.

$$LET_{\text{Eff}} = \frac{LET}{\cos \theta} \quad (2-2)$$

Thus it is recommended that not only the basic cross section vs. LET curve be collected, but also the angular response should be examined. Here we proceed from the following position. If angular data can be collected, then it should. If not, then the cosine law should be assumed to hold to 60 degrees (meaning that ions striking off of normal incidence can be assumed to follow Equation (2-2) up to 60 degrees, and after that the LET_{Eff} is taken to remain constant). This is generally a conservative estimate, but can miss event types that are not seen at normal incidence. It is also true that RHBD structures may have significantly increased SEE sensitivity at angles other than normal incidence, and for these devices, angular testing of the device or related test structures is required and should be performed along multiple axes [65,70].

2.1.3 Proton Test Data

Proton test data needs are very similar to that for heavy ions. The relevant event types must be counted as an appropriate³ number of protons of various energies are applied. In the case of protons, however, the response is required relative to proton energy with two main parts of the energy range being relevant, and this is sometimes complicated by increased proton sensitivity due to modern interconnect materials such as

² As a direct example, consider the parity protected L1 cache found in many microprocessors. In order to use parity protection in the data cache to enable silent error correction, the cache must be operated in write-through mode (where new data is written through the L1 cache, into error correction protected memory). This mode incurs a reduction in processing power that in some applications is not acceptable. Thus, the L1 cache will be a source of software crashes even though it is protected by fault tolerance.

³ “Appropriate” refers to a number of protons sufficient to observe a statistically significant count of any relevant events while limiting TID exposure.

Tungsten plugs [66]. The first consideration for determining the proton sensitivity is that a threshold energy and saturated cross section must be acquired in order to enable fitting the Bendel curve that describes the expected response when proton SEE is due to secondary reactions [67]. An example of proton data from [17] is given in Figure 2-2. It should be noted that in modern devices proton direct ionization may occur which can cause the device sensitivity to remain high below the cutoff for spallation events, and this may be evident in Figure 2-3.

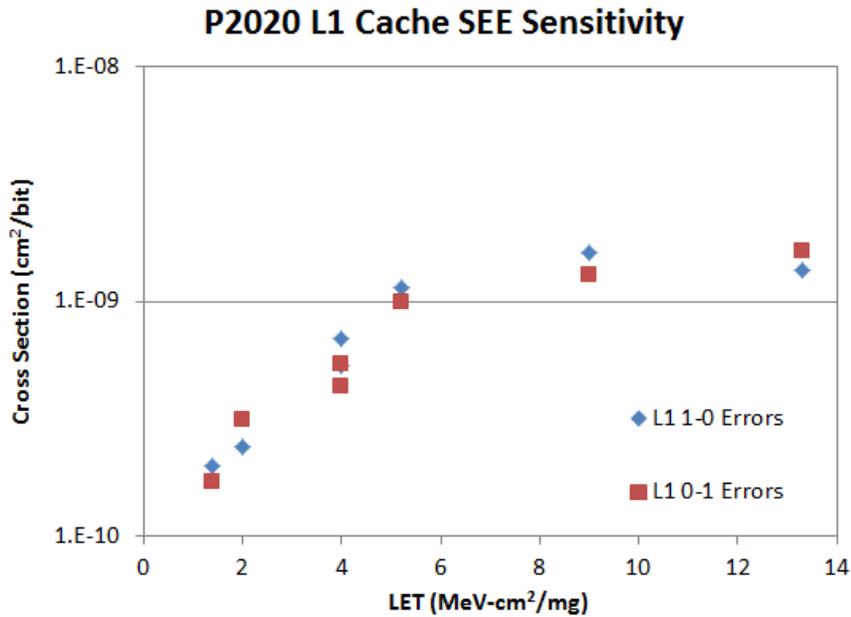


Figure 2-1. Cross section vs. effective LET example for the L1 cache of the P2020 SOC [53].

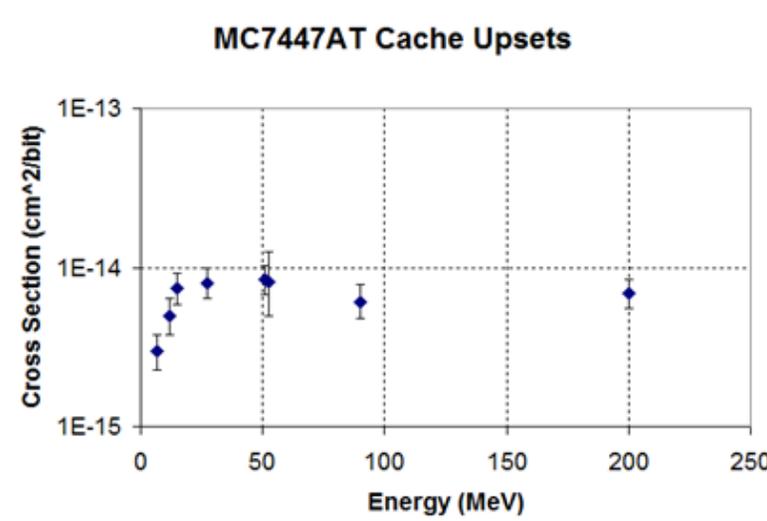


Figure 2-2. Example of proton SEE response in testing the Freescale MC7447A [17].

The second consideration in the proton energy spectrum is the device response as the energy decreases and proton direct ionization becomes a possibility [68]. In general, this becomes a concern for feature sizes at or below 90 nm (although it may contribute at larger feature sizes). An example of SRAM data showing proton direct ionization impact on the device response is shown in Figure 2-3 [69]. It can be seen that the response can be one to two orders of magnitude higher than the flat part of the curve. Unfortunately, for many of the devices we are discussing in this guideline, it may not be possible to obtain good data at low

proton energy. For these devices it is recommended that data on parts from the same fabrication line be used to establish the proton sensitivity. If this is unavailable, a conservative bound to the possible low energy behavior can be inserted into the rate calculation tool used to see how sensitive the rate might be and determine if the unknown response is a significant problem. For further discussion of testing for low energy proton SEE, see *Hardness Assurance for Low-Energy Proton-Induced Single-Event Effects*, by Dodds [71].

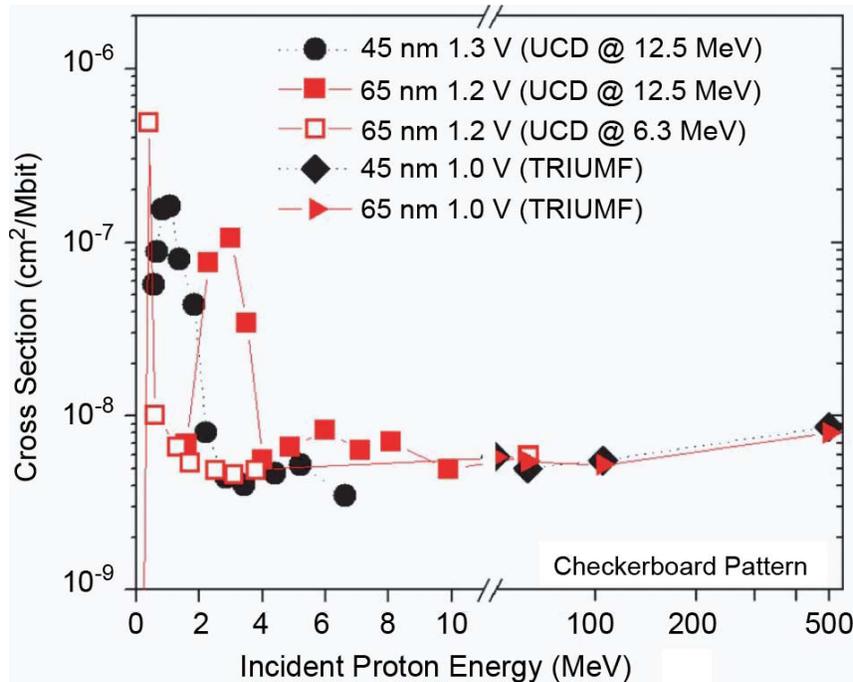


Figure 2-3. SEU cross section vs. proton energy for IBM 45-nm and 65-nm SRAMs [68]. Image used with permission. It is unclear why the peaks do not line up. However, the general observation that the proton cross section can increase with decreasing proton energy is the key finding for this guideline.

Proton exposure can further be complicated by high LET particles produced from their interactions with certain device materials, such as Tungsten plugs, which were not present in older devices. This means that early work on establishing the distribution of particles generated as secondaries during proton exposure of microelectronics [72] may not apply to modern devices, as demonstrated in [66]. This is only important if the device response changes between LETs of 10 and 50 MeV-cm²/mg.

2.1.4 Alternate Exposure Types

Aside from standard beam testing, there are some other alternatives that can help to determine the behavior of devices. In addition to determining the device behavior, however, alternate methods can provide valuable insight on software or other parts of test setup for beam testing. The two most common alternatives are laser testing and micro-beam testing [74,75].

We first discuss the very useful elements of these test methods, and then we highlight limitations and difficulties. Both laser and micro-beam are useful diagnostic and exploratory test approaches for the development of RHBD structures. In addition, they are useful for examining the behavior of test algorithms in general. They can also be useful for identifying the portion of a circuit that causes a known SEE type by matching the signature of a known event type from broad beam testing.

Both test methods have limitations to their viability as primary test vehicle for SEE effects. Isolating testing to small regions of a complex device is the primary limiter of using these methods. Even though it is their major strength, it creates a huge test matrix. This occurs because the device is large in terms of area and because its operating state has a very large set of conditions. Because of device complexity, test algorithms

are also complex. This means a very large number of individual test runs will be required during raster scanning in order to get good results. For example, when used on a transistor, a laser or microbeam test is concerned with the worst-case bias state where the most sensitive node is reverse-biased. Then the laser or microbeam is used to map out the key nodes where SETs can be observed. With a processor, it is unknown what function should be executed in order to sensitize a node to a charge-injection event.

The size of the laser beam area is a fundamental limitation for scaled devices. The smallest spot size is on the order of 1 micron, which is considerably larger than the cell area. Laser beams (due to lack of view area) and microbeams (due to range) sometimes cannot penetrate metallization layers, which is a severe limitation. Although laser irradiation from the back of a device can be used to overcome this, the spot size is still much larger than the feature size. This means there is a high likelihood that laser-induced behavior will be different than actual ion testing.

The best approach for laser testing is (A) if readily available, use it early in the algorithm development process to assist debugging—but do not let it lead to custom detection algorithms because laser upsets may not happen in broad beam testing, and (B) use it after broad beam results are collected to help iron out the cause of anomalies (principally to verify if anomalies are due to single targets or coincidence of multiple targets, or to enable manufacturers to remove vulnerabilities). The best approach for micro-beam testing is to use it after broad beam testing to help isolate targets of interest to collaborators (such as the manufacturer) with an interest in improving later designs. Micro beam testing can also help in examining anomalies. Note, however, that anomaly investigation using either method can be very difficult if the anomaly is not easily found.

2.1.5 Worst-Case Conditions

Collection of SEE data for characterization should be collected using worst-case conditions. However, many of the SOCs we are examining cannot be exposed to worst-case conditions. These primarily fall into four categories: temperature, voltage, TID exposure, and application use.

Actual worst-case conditions should be determined for the real application and translated, if possible, to the test equipment. In many cases this cannot easily be done because it is not possible to force operating voltages, or to ensure that workload is as desired. It is recommended that the radiation test engineer and application engineer attempt to ascertain the different types of problems the device may encounter, and then establish if worst-case conditions are possible, or if nominal conditions provide a sufficient means to evaluate device performance. The actual application need (and therefore the application engineer) should provide general guidelines and help establish what parameters of the operating space are actually relevant for use.

Temperature should be controlled for SEL. It is known that SEL is more likely to occur when the device is run at elevated temperature. For most of the parts we are examining here, the temperature is likely to be elevated during testing (perhaps as high as 85°C at the surface if thermal management has been altered). Unfortunately, we cannot recommend testing the subject devices at the manufacturer's specified high temperature unless the device is not physically altered and you can leave the recommended heat sinking in place. Of course if you are using a heat sink, it may not be reasonable to operate at the specified high temperature. In practice this document recommends that the test personnel ensure that the device is not artificially held at a low temperature and that the highest reasonable temperature is applied during SEL testing. Do not allow this to potentially compromise modified test parts, though.

The operating voltage when performing SEL testing should be as high as possible, within the device specification. When testing for SEU or SET, the voltage should be as low as possible, within the device specification. In practice it may be impossible to alter the operating voltage of a pre-built board, and it may introduce test artifacts due to mismatched voltages between multiple components. This is a side effect of testing components that are part of complex systems. An effort should be made to control the voltage if possible.

The interplay between TID and SEE may have an impact on devices. Methods for determining the impact of TID on SEE testing are discussed in Schwank et al. [33]. Because the SOCs discussed in this guideline may contain many different types of circuits, they may be very sensitive to this interaction. However, most of the devices discussed achieve fairly high TID levels (above 10 krads(Si)), and most missions of interest here require levels below this, so we have not addressed this problem in any depth. Some missions, such as Jupiter orbiters, may need devices that can withstand hundreds of krads. The general approach suggested is to prepare devices with and without the mission TID requirement; then test both to determine if there is a significant change in SEE response. If no major difference is seen, perform most of the testing on devices without TID exposure. The risk here is that SEE testing also effectively adds dose, and so the devices are always a moving target if TID can alter the SEE response. So using devices with low TID exposure is best for ensuring repeatable results.

It is important to note that SEE testing exposes devices to TID. This cannot be avoided. Care should be taken to keep the TID incurred to a controlled level.

The flight application's worst-case conditions should be used during testing, if known. Since the flight application is usually not known at the time of testing, this document recommends understanding the SEE sensitivity of the device relating to individual subsystems. From the subsystem sensitivity, an application worst-case assessment can be made by assuming all observed operational problems can occur on the flight application.

Single event gate rupture (SEGR) and single event burnout (SEB) are possible in the transistor structures in SOCs. These event types are most likely when there is high bias applied to transistors, near their maximum rating. In order to test for these, specification maximum voltage should be applied to power rails and IO pins.

End of life radiation performance is also a consideration when designing a test. This is typically part of a qualification effort, rather than general radiation characterization. This guideline is focused on radiation characterization, but we can provide some recommendations. It is expected that all SEE phenomena will be worst case at either beginning or end of life. Thus it is recommended to consider both fresh and aged devices for the test matrix used in SEE testing.

Finally, for some SEEs, high clock rate is worst case. While for others, low clock rate (or static condition) is worst case. It is generally not possible to establish which is worst case for a specific SEE type, so high and low frequency (or no clock) should be considered for test planning. It should be noted, however, that this effect is relatively minor in most cases and may be extraneous to the test plan if not required by the application.

2.2 SOC Characterization

There are some details of radiation characterization of SOC that are more difficult or unique to SOCs. This section discusses these briefly.

2.2.1 Time-Model of Test Beam and Targets

The implementation of a test algorithm will leave the test device vulnerable to SEU in planned and unplanned regions. The structure of the test algorithm will impact how events are observed and may be important for event counting. It is important to know what the time-model is for any desired upset sensitivity measurement and its corresponding algorithm. At the same time it is important to understand the way in which upset sensitivity of unintentional targets works.

When testing, there are three types of targets typically built into test algorithms. The first type comes from the test algorithm where an operation is performed and the result is checked immediately. In this case the portions of the device relevant to that operation are all that are tested, and the result is immediately verified. There is no buildup of errors inside the algorithm. Because of the way this type of test is performed, there

is often very little duty cycle to any particular part of the target operation. (For example, if one instruction is tested many times, there may be very little pipeline optimization and the instruction's six or more clock cycles may be required for every execution resulting in a low effective duty cycle.) The second type of target is the integration target. Here the idea is that a target is sensitive continuously, but that the best approach to the test algorithm is to periodically count errors and reset all the targets to known values.

The final type of target structure is an unexpected target that parasitically attaches to all test algorithms in one way or another. Traps, interrupts, and system configuration registers make up some of these targets. An example is a hidden configuration register. Here we end up with targets that can impact how the system operates without being directly observed or observable. By nature, any given test algorithm will be unable to fix these errors, and they can give rise to anomalies in the test data. Because the errors are not fixed, they also contribute to error buildup and over time may begin generating strange results.

All three target types are diagrammed in Figure 2-4. Here the idea is to show the test flow in blue (or for continuous targets, show that the test flow is unimportant), then show how upsets can build up during the test due to this test flow, and finally indicate how the test algorithm must work to detect events. The first group of three arrows focuses on unhandled targets that build up errors during testing with unknown potential impact that gets more problematic with time. The second group of three arrows focuses on periodic test algorithms with periodic error checking and test state reset. Note in this grouping that upset buildup occurs, and this can be used to observe error correcting code (ECC) systems that are unable to fix errors. In the third grouping are other event types that are constantly monitored or to which the system is constantly sensitive (such as progress of an algorithm or response to a machine interrupt).

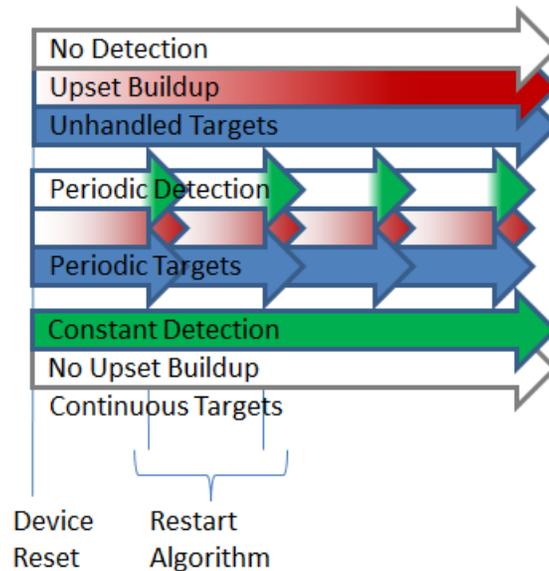


Figure 2.-4. The time structure of different types of targets in a device. The top, unhandled target, structure is expected for any part of the device not returned to a known state periodically.

Upset buildup contributes to higher event counts in fault-tolerant systems because multiple events enter the system before they can be cleared. Upsets in the unhandled targets can lead to complex anomalous behavior (i.e., the SOC configuration can become more and more corrupt). For some devices there may be hidden targets that do not clear with device reset, so power cycling is recommended between test runs. We can point directly to the special purpose registers in most high performance processors (such as memory configuration). These registers are usually configured once and assumed to remain unmodified. If the beam changes these, the test software is unlikely to be able to periodically scrub all such registers.

2.2.2 Selecting Test Software and Hardware

For each structure of interest to the general characterization approach or customer program, a software algorithm for stimulus and a related hardware system are required. (In some cases, test structure data does exist, and it may be possible to use that rather than test a DUT to evaluate. However it may be of interest to verify performance of structures for which test structure data exists.) For an SOC, there may be many ways to approach any given test structure, but in all cases either an external debugging system is used to stimulate the SOC, or an in-system algorithm is used. In most cases the hardware approach for testing involves internally routing signals so that either self-monitoring algorithms or debugging tools can be used to examine test data. In addition, the hardware approach on external I/O signals may be a loop-back interface, an external communications system, or a bit error rate tester (BERT). Because of the complexity of modern SOCs, it is not recommended to build a custom test board unless there is a long-term interest in continued testing; the technical difficulties lead to a very expensive development effort. Appropriate test hardware selection is discussed further in (Section 3.3).

Software algorithms for SOC testing are discussed in greater detail in Section 3.6. Here we do need to point out that for general testing the recommended software approach is to develop software that enables the SOC to test itself. This has been discussed generally in Guertin and Irom (2009) [18]. As this approach sometimes leads to difficult error types to interpret, it is also recommended that test engineers be familiar with debugging tools that enable direct examination of device resources.

2.2.3 Event Normalization

Event observation is relatively straightforward, but there are some confusing elements of normalization when testing SOCs. For any target event type, observed events are straightforward to count. Similarly, beam counting is also straightforward. But for any given exposure, the important measurement is the cross section normalized to some intrinsic structure. The most common structures are memory bits and the entire device. When testing SOCs, specific structures, or subsets of structures will actually be tested. This occurs for three reasons. First, in many cases the SOC cannot be entirely exposed (due to thermal or socket reasons), and the portion of the device exposed is only roughly known [30]. Second, when testing a complex device, it is often the case that not all of the relevant nodes of a design will be active at all times during exposure. Third, the structure of the test algorithm and its matching to beam delivery can be important for establishing normalization (this is only the case when the beam delivery is not approximately uniform in time, such as when using the NASA Space Radiation Laboratory, NSRL).

One important point to make here is the relationship between event normalization and test algorithms. An SOC is primarily a set of microprocessors with support peripherals built into the chip. The microprocessors and peripherals are made up of three primary types of items: memory bits, sequential logic chains, and I/O devices. The memory bits are the easiest to normalize as: “all memory bits are always sensitive under most test programs”; however it is usually the case that a significantly reduced fraction is actually relevant.

In this document, we will often discuss static testing. However, there is some evidence that use of on-chip memory during testing may result in altered response. For example, a static, unused cache may have different sensitivity than a cache that is actively loading and storing values. The largest effects are likely to be limited to devices with clock gating on the memory arrays. It is reasonable to suggest use of memory will be the more realistic case for SEE testing of devices in the future [52].

Relative to the memory devices, the sequential logic chains are much more difficult to normalize. First, the actual implementation of execution structures is more-or-less unknown to the test engineer without direct support from the device’s manufacturer. Thus, it is not obvious what is even being stressed. Second, most sequential logic chains in microprocessor elements are triggered by dispatching instructions. Thus, a test program must be written to issue instructions to run the desired chain. Third, microprocessor sequential structures are very small, but there are a lot of them. This final point means that if the desire is to test an actual sequential structure, the test engineer must be aware that the cross section is going to be significantly

smaller than any memory or register structure. For purposes of calculating a rate for a sequential structure, the smallest reasonable number of sensitive nodes or bits should be used in

I/O devices round out the majority of structures of interest for SEE testing. The difficulty with I/O device data normalization is determining an appropriate test algorithm and data structure as a basis for normalization. That is, if you determine that the output drivers have a specific cross section for transients, but no user algorithm will be sensitive to that type of transient more than 10% of the time, then you may overestimate the relevant event rate. This topic is currently being examined for a future version of this guideline. For the present, however, the approach recommended here is to establish the cross section as the usual upsets (whatever upsets or glitches can be observed) divided by the fluence, but then provide a second value where you divide this by the number of transmitted data bits (bits sent or received by the DUT during the test exposure). I/O data is often transmitted using first-in, first out (FIFOs) and other buffers, so the cross section may increase as the data transfer rate increases and the utilization of FIFOs and other buffers increases. The recommended use of this comes down to developing (A) a traditional space rate, and (B) a space rate per transmitted bit. These values can then be used as alternative bounds, taking the higher rate as the more conservative. (As a specific example, if a radiation test were performed at 1 Mbps, then the test device is used at 1 Gbps, this would suggest increasing the traditional SEE rate by 1000x to account for the difference between the test system and the application.)

2.2.4 Estimating Angular Response

Modern SOCs are usually very difficult to expose for normal incident ions, due to the predominance of flip-chip architectures and heat issues requiring careful handling of modified chips. In some cases the only viable approach is to open a small hole through a heatsink or heat spreader. These devices are even more difficult to expose for ions incident at angles 45° or more relative to normal. This means that it is often very difficult to collect test data for a large amount of the solid angle of environmental particles. In the absence of a good understanding of the directional mode, a reasonable approach is to estimate the angular response when using rate calculation tools (note that this means the engineer should use whatever information is available to create such an angular response estimate, e.g. DICE cells do not follow the cosine law [70]). A model where the cosine law holds to 78° can be approximated by intentionally setting the rectangular parallelepiped (RPP) ratio to 1/5th in tools like CRÈME96 [62,76]. It should be noted that this does not account for all angular effects, including some known to occur in space. Further, this interpretation is not physical and can result in strange situations such as RPP depths of 0.2 μm – but it makes the rate calculation software get the angular response right, which is more important. Certain special situations (especially devices with low observed SEE sensitivity, like DICE latches, or SEGR which is worst case at normal incidence) require further consideration of additional angular effects.

2.2.5 Higher Beam Energies

Many candidate devices are flip chip design and have packaging over the silicon. The packaging is often designed to support a significant heat sink, possibly including a heat spreader over the silicon. Because of this, and difficulties with hardware setup it may often be the case that the device die cannot be thinned. Only relatively high energy heavy ion beams can be used to test these devices. One facility that can produce beams capable of hitting the sensitive volume of these devices is the Texas A&M University (TAMU) cyclotron. Figure 2-5 shows the LET versus range curves for the 40 MeV/amu beams at TAMU. Note, discussions about running these beams with operators at TAMU prior to arrival are recommended; and especially note if the Kr beam will be needed.

Higher energy beams may be used to test these devices. These higher energy facilities are usually cost-prohibitive, and this type of testing is outside the scope of this guideline (for example one such facility is NSRL, where beam time costs approximately five times that at more traditional facilities such as TAMU).

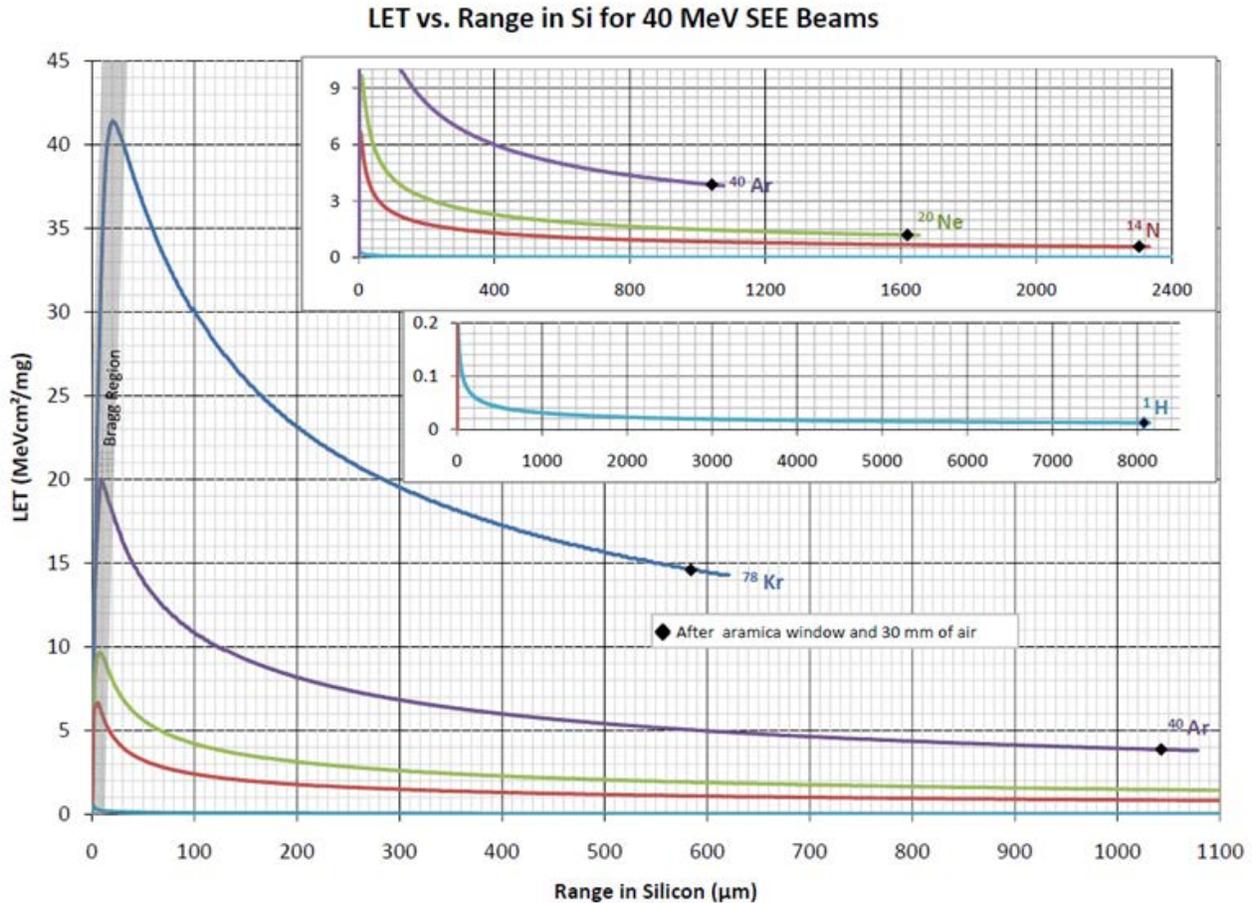


Figure 2-5. The range versus LET plot for 40 MeV/amu beams at TAMU [77] Image used with permission.

2.2.6 Beam Flux

The flux of the beam used for testing is critically important on both ends of the range, high and low. With RHBD devices, high fluxes may be required to observe upsets within reasonable test duration. With fault tolerant (FT) devices, low fluxes may be required to reduce the occurrence rate of anomalies due to overwhelming the FT. In general, the beam flux should be selected so that at least a few upsets (of the type desired by the test plan – device, bit, etc.) occur per minute. With some devices or structures within devices, however, event rates will be several thousand per second even at the lowest reasonable test fluxes [30].

When suspicious events are seen, care must be taken to determine if these events are due to multiple upsets (some of which may not be cleared during execution). The time-model of the test algorithm, as discussed in Section 2.2.1, can lead to unexpected events through a couple of mechanisms. The hardest mechanism to rule out, and therefore the one that establishes the difficulty of determining if an event is due to test artifacts or due to an underlying upset sensitivity, is the mode where two upsets in a short period of time lead to the event type (this is not necessarily due to overwhelming fault tolerance – for example you could get an SEE while another SEE report is being generated). In this case, and in the case of possibly overwhelming fault tolerance, the behavior of the system in the presence of faults is described by the same set of theoretical foundations [27]. Figure 2-6 shows the theoretical curve for system response as a triple-module redundant (TMR) target is hit with increasing upset rate. In this case there is an extrapolation of the ideal case (where the system response does not saturate (dashed)) and a plot of the actual case (where the system response does saturate (solid)).

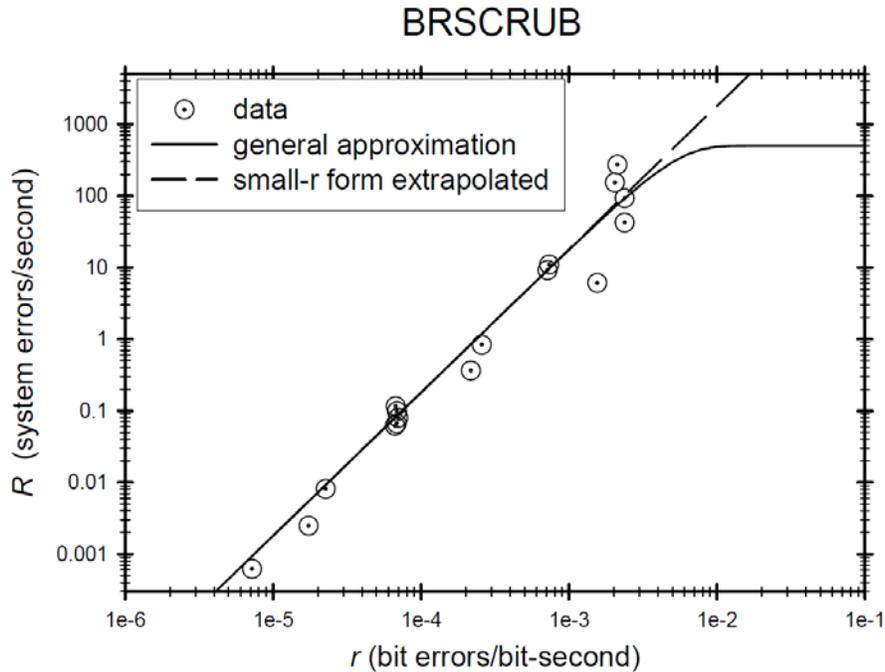


Figure 2-6. System error in ideal fault-tolerant system as a function of flux (similar for beam delivered over an integration window for an algorithm) [27]. Credit to L. Edmonds at JPL for figure.

Figure 2-6 also assumes the ideal case that the system only exhibits errors when the TMR is overwhelmed by multiple hits. In a real system there may be underlying sensitivities where the system encounters an error from individual targets that do not require multiple hits. These targets are important to understand because they may be very rare compared to fault-tolerant elements, but in space the reduced flux and very long integration times will make these underlying sensitivities more important to a project. An example of such an underlying sensitivity is discussed in Guertin and Irom (2010) [17] and is shown graphically in Figure 2-7. Here it can be seen that once the raw bit flip rate gets down to 1×10^{-5} bit-errors/bit-second, the error rate stays essentially constant.

One may be inclined to argue that if flux dependence such as that seen in Figure 2-6 is observed, then there is evidence that the observed error mode is due to flux and will not be seen in space. To show this, the test engineer could propose to increase flux when an anomaly is observed, in order to demonstrate that the anomaly rate increases. Unfortunately, this does not address the question of an underlying sensitivity. In the event of anomalies, only two test methods are viable. First, decrease the flux until the event drops below the threshold for the project. Or second, show that at different fluxes the error rate is the same, thus eliminating the possibility that the anomaly is an artifact of the test. Unfortunately, the latter shows that the error mode will occur in the project application and analysis of event impact and space rate will be required.

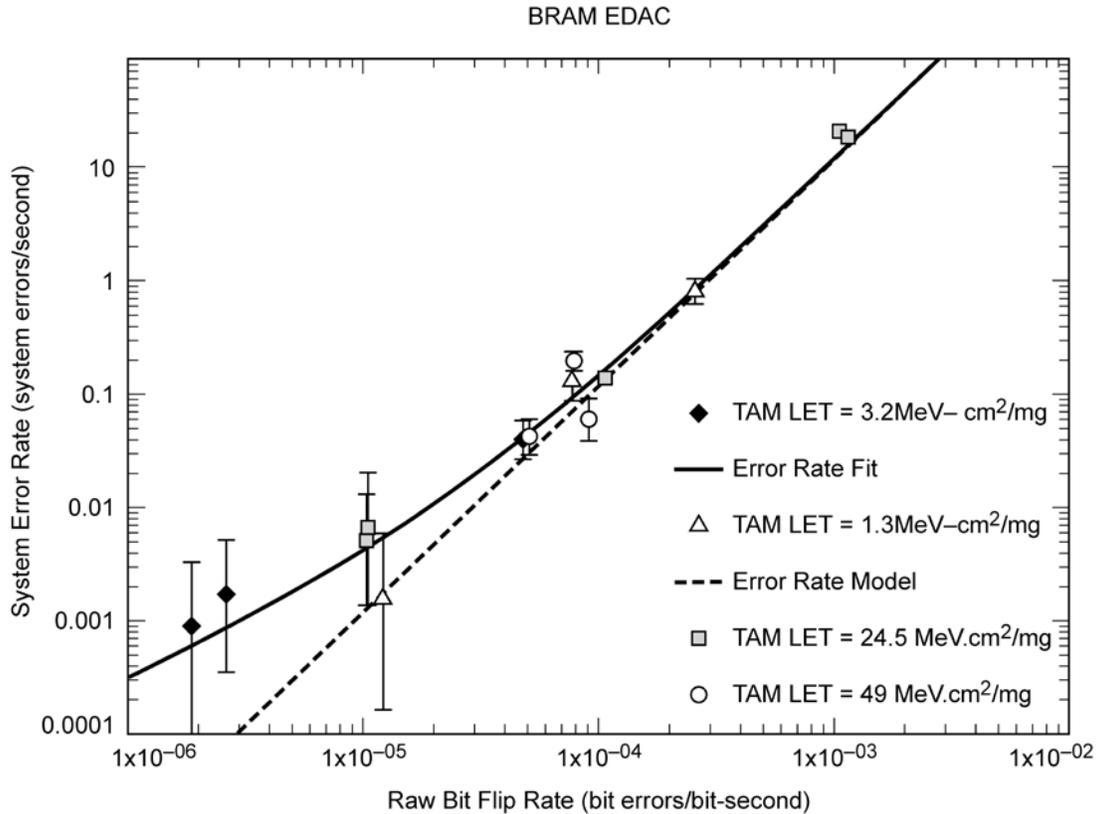


Figure 2-7. Similar to Figure 2-6, this shows how the real device response differs from the ideal situation when an underlying error upset mode exists in a fault-tolerant system [28]. Image used with permission.

2.3 Software Characterization

The difference between upset rates established during radiation characterization and the upset rates that will actually be seen in use comes from differences between test and operational software packages. In order to see the potential impact of these differences, one needs only to look at the upset rates of different applications on the same processor as examined in [78]. In this work it was seen that depending on what the processor is doing, the fraction of faults that result in an error ranged from below 0.1% to above 90%. Similar results have been obtained on the tested SOC's by noting the two most common methods used for testing (test algorithms are discussed in more detail in Section 3.6), running a stand-alone highly robust test application, compared to running a simple test algorithm on top of an operating system. The former has been demonstrated to be sensitive to less than 0.1% of faults, while the latter usually has errors on more than 20% of all faults.

Because of the widely varying fault-to-error ratio as a function of the operating software it is recommended that user software be tested. However this is usually not an option for a number of reasons. Thus, we recommend that once the basic sensitivity of the SOC is known, then fault injection should be performed on a simulation of the SOC and application. Significant information about fault injection and emulation can be found in Quinn et al. (2012) [79] which summarizes a lot of information related to faults in microprocessors.

In the absence of ability to perform fault injection, a bounding approach can be taken. However, this requires either very conservative estimates or very good information about the device and its use. For example, in many devices the L1 cache bit sensitivity can be well known and may be on the order of 1×10^{-9} cm²/bit at saturation with threshold below an LET of 1 MeV-cm²/mg (see Section 6.4 for test data on Freescale SOC's). Even though the device may have parity protection on the L1 cache, the user

application may desire to disable this (which is not uncommon because parity protection on the L1 cache causes a significant performance hit). In this case, the L1 cache almost certainly drives the SOCs error rate because it has ~300,000 bits, giving a cross section for the L1 cache of $3 \times 10^{-4} \text{ cm}^2/\text{L1}$, with multiple L1 caches on a multicore SOC.

2.4 Summary of Recommendations

This section reviews the key recommendations from this section, highlighting needs for data collection in radiation characterization.

1. Determine if heavy ion testing is needed based on project environment.
2. Determine if proton testing is needed based on project environment and results of heavy ion measurements.
3. For heavy ions or low energy protons, determine device preparation sufficient for testing. The main thing to note is the set of beams and necessary materials to traverse.
4. Determine if heavy ion testing can be performed with tilt angles of 60 degrees or higher—if not, try to get test structure data; then for non-RHBD devices, assume the cosine law works up to 60 degrees. For RHBD, be aware of a significant hole in your testing because steep angles may form the majority of the SEE sensitivity.
5. For SEL testing, allow the device to get as hot as possible within the device specification and without risking damage to the part.
6. If the device voltages can be altered on the test board, use the highest reasonable voltage when performing SEL testing and the lowest reasonable voltage when performing other SEE tests.
7. If TID is a concern, prepare some devices with near the end-of-mission TID level and use them to determine if SEE response varies. Also note that TID may alter SEE response during testing, and try to enable your test to distinguish this effect.
8. Ensure test algorithms match beam delivery method (pulsed versus continuous).
9. Possibly use laser testing to establish viable test algorithms, but do not limit detection, or dramatically increase algorithm scope based on laser as laser results can be a superset or a subset of the SEE response in the natural environment.
10. Identify the structures you intend to test by determining all of interest, or narrowing by project requirements. Pay particular attention to large targets such as caches or embedded memory. But also beware that for RHBD and FT devices, testing should not be overly focused on protected elements.
11. For the purposes of rate calculations, always assume the smallest reasonable number of sensitive bits have contributed to observed cross sections – this way, rate calculations are conservative.
12. Pick a test approach, but err towards having the DUT test itself.
13. Determine hardware and software for testing each structure of interest.
14. Try to establish approximate flux and fluence (test matrix) for each target structure before going to the beam. (For fluence, estimate how much is needed to get a sufficient number of events. For flux, estimate the maximum flux before test anomalies become problematic.)
15. Be prepared to run at very low flux to rule out low cross section events on FT devices
16. Always power cycle the device between beam runs unless there is a significant time limitation and anomalies are being ignored.

17. Use laser testing to iron out issues uncovered with beam or for debugging of test software. (There are additional uses of laser evaluation, but they are suggested for a future version of the guideline.)
18. Be aware that laser testing may result in behaviors that are not observed in beam testing or in space applications.
19. Worst case conditions should be identified for all structures likely to be in the test device, and within the limits established by the application.
20. Test planning should include worst case for parameters as identified by application engineer, for: SEL, SET/SEE, TID, ageing, SEGR, SEB. These are particularly important for qualification needs and should be established with the program or user. The goal of the characterization effort would be to establish a safe operating area (SOA) for the operational parameters.
21. For rate calculations, use information about known device/test structure response to create models in rate calculation tools.

3.0 TEST PREPARATION

In this section we review all of the details that go into preparation for testing of a modern complex SOC. Where appropriate we will discuss specific details of test preparation, but necessarily each device is unique and will require care to test correctly.

This section discusses device preparation, error detection, the approaches that can be used to design a test, and the types of software algorithms that can be used. The final part of this section provides a review of recommendations from the section.

3.1 Introduction

Test preparation covers all the aspects of the SEE test leading up to actually going to the test facility. The general goals of test preparation are the following. First, we must ensure hardware is available to enable functional testing of the DUT. Then we must come up with hardware and software elements that enable testing of the device. And, we must discuss the types of beams that can be used for testing, and the design of a test campaign. The target devices are so complex that a realistic test approach must include being prepared for unexpected test issues or planning to perform testing on more than one test trip.

The present work targets SOCs, but the majority of thinking on this subject has been carried out on high performance microprocessors [1,20,22]. Some details have changed from these earlier works, regarding test algorithms and hardware operation. We will discuss these here.

3.2 Beams and Test Facilities

This section reviews details of test facilities and how the available beams may be used to test the SOCs targeted by this guideline. First, we discuss beam penetration in devices. Then we discuss general information regarding proton and heavy ion beams that can be used for testing. Finally, we review the various facilities that can be used for testing and provide a matrix for part preparation depending on the target facility.

3.2.1 Beam Penetration

Beam penetration in silicon is well-documented for the various ion beams that can be obtained. For modern SOCs, other materials are important for beam penetration. Most devices employ a heat sink, which is often made of aluminum. If a heat spreader is used, it is often made of copper. For some devices there is also an epoxy fill between the heat spreader and the die.

Table 3-1 provides material and density information on the most common materials used in the mounting and thermal control of flip-chip devices [80,81]. Approximate beam range is reduced to only 26% of the Si range when traversing copper (actual range should be calculated using particle physics tools), which means that copper heat spreaders clearly cause problems. Heat sinks are often made of aluminum alloys such as 6063, which has density given in the table. Since heat sinks usually have a fairly thick contact plate and require contact with an intact heat spreader to provide useful heat transfer, it is clear that heat sink material in the beam must be carefully examined and modified as needed.

Table 3-1. Densities of materials used in SOC packages.

Material	Density (g/cc)	Si-Range Ratio
Si	2.3	1
Cu	8.9	0.26
Moulding	1.8-1.9	1.24
Al	2.7	0.85

3.2.2 Proton Beams

Proton beams are usually very penetrating. This makes testing with protons fairly simple because they can traverse both a heat sink and a heat spreader, in addition to the actual DUT die. There are several proton facilities in the U.S., including University of California, Davis (UCD), Massachusetts General Hospital (MGH), and Loma Linda, in addition to the Tri-University Meson Facility (TRIUMF) in Canada⁴. Most heavy ion facilities can also provide protons if needed, though the energies are limited compared to the proton facilities. Proton range is often long enough to enable testing with tilt angles.

3.2.3 Heavy Ion Beams

Heavy ion beams are produced with fairly low energy and are not very penetrating. In many cases the penetration does not exceed 100 μm , in which case flip chip devices essentially cannot be tested. When heavy ion ranges are greater than 100 μm but less than 1000 μm , then it is possible to test flip chip devices, but only in regions where both the heat spreader and heat sink can be removed while maintaining a working device. In this range, testing with angles is not possible due to the limited penetration except under special conditions where the geometry lines up favorably (i.e., shadowing does not affect regions of interest). In some high energy facilities, it may be possible to test with heavy ions without removing heat sink or heat spreader, but it is necessary to ensure that the beam penetrates into the active region of the device and that the Bragg peak of the ion beam occurs deep enough such that the LET of the ion nearly approximates a constant through the sensitive region of the device.

3.2.4 Laser Sources and Other Beams

Several facilities are available that can provide laser injection of SEE. We primarily recommend using these facilities if they are expedient, but acknowledge that test artifacts generated from laser testing may be too difficult to disentangle with these complex components. They are very useful in examining known SEE types in detail, rather than surveying a device for general SEE behavior. Since SOCs tend to be quite large, a scan for general effects can take a very long time. Also note that most of the components discussed here are flip-chip devices that require very difficult device preparation to enable lasers to stimulate them for SEE.

Two facilities that can be used to investigate SEE with lasers are the Naval Research Laboratory (NRL) in Washington, D.C. and JPL in Pasadena, CA. Other facilities are available including some in Europe.

3.2.5 Facility Review

We examine the common facilities for radiation testing of electronic components. The focus of this section is to provide a way to rapidly determine appropriate facilities that can be used for desired beam properties while assuring that DUT preparation requirements are known. Heavy ion information is similar to that found in Irom (2008) [1]. Information from the various facilities [82,83,84,85,86,87,88] provides the material presented in Table 3-2.

⁴ There are numerous proton treatment centers and alternate proton sources that are being investigated and used in light of the closure of the Integrated Science and Technology (ISAT) hall at Indiana University, but we do not list them here because this topic is still in flux, and this document is not able to capture the changing nature of the issue.

Table 3-2. Review of Proton and Heavy Ion facility beam properties and how they affect device preparation. (ISAT = the Integrated Science and Accelerator Technology Hall [formerly Indiana University Cyclotron Facility IUCF] – now closed), NSRL = NASA Space Radiation Laboratory, RADEF = Radiation Effects Facility (Finland), LBL = Lawrence Berkeley National Laboratory, UCD = University of California, Davis, UCL = Université catholique de Louvain). (Note the range column provides energy for proton facilities and LET for heavy ion facilities.)

Facility	Primary Beam	Range E(MeV) /		Required Modifications				
		LET (MeV-cm ² /mg)	Range in Si (um)	Ceramic, Face Up	Bare Flip Chip	Flip Chip w/ Heat Spreader	Heat Sink over Die	
ISAT (IUCF) [13] (offline)	p	30-200	213-5500	None	None	None	Uniform	
UCD [14]	p	1-65	1-820	None	None	None	No	
TRIUMF [15]	p	20-500	99-25000	None	None	None	Uniform	
TAMU 40-MeV/amu	HI	1-15	600-1600	None	None	No Spreader	No	
TAMU 15-25-MeV/amu	HI	2-80	125-250	Remove Cover	Thinned	No Spreader	No	
BNL-Tandem	HI	1-6	85-200	Remove Cover	Thinned	No Spreader	No	
BNL-Tandem	HI	6-85	30-85	Remove Cover	n/a	n/a	n/a	
LBL-High LET	HI	2-100	53-80	Remove Cover	n/a	n/a	n/a	
LBL-High Penetration	HI	1-50	150-500	Remove Cover	Thinned	No Spreader	No	
UCL-High LET	HI	2-60	43-80	Remove Cover	n/a	n/a	n/a	
UCL-High Penetration	HI	1-33	90-270	Remove Cover	Thinned	No Spreader	No	
RADEF	HI	2-55	90-200	Remove Cover	Thinned	No Spreader	No	
NSRL	HI	0.05-100	2-600mm	None	None	None	Uniform	

Each facility in Table 3-2 is analyzed for the difficulty involved in testing the types of SOCs discussed in this guideline. The columns are as follows. The facility is just the facility name. Each facility is primarily a proton (p) facility or a heavy ion (HI) facility. The range listed is the approximate range of either energy (for protons) or available LETs (for heavy ions) including beam tuning and degrading. The range in Si column is a rough indicator of the approximate depth the provided particles can penetrate. The final four columns indicate what package modifications are required to test at the facility. The first column refers to parts that are mounted in ceramic packages with the die facing up. The second column refers to parts where the die is visible and is flip-chip bump (or similar) bonded to a lead frame on the package. The third column is the same as the second with an additional metal cover that spreads heat positioned on top of the die. The final column refers to what has to be done to devices that in general require a heat sink in order to operate. In this column “uniform” indicates that the heat sink should be made to uniform thickness anywhere where the beam must cross the heat sink. None means no modifications are needed. No means the heat sink and/or spreader should not be present. n/a means that the device should not be tested at the indicated facility.

For the proton facilities and the high energy heavy ion facilities, ceramic packaged devices and bare flip-chip devices can be irradiated with no modifications required. For lower energy beams, ceramic package covers must be removed while bare flip-chip devices must be thinned. The lowest energy beams cannot be used on flip-chip devices due to insufficient range regardless of the preparation of the device. When heat sinks must be used, the heat sink must have a constant thickness in order to enable reliable beam property estimation. In the case of heat spreaders, in all but the most energetic beams, these must be removed over the target portion of the DUT, leading to complications described below.

3.3 Device Preparation

The SOCs of interest here largely fall into two groups, RHBD application-specific integrated circuits (ASICs), and high-performance commercial devices. (A notable exception is the Maestro ITC which fits into both categories.)

Generally speaking, RHBD ASICs do not dissipate a large amount of power. As such they can be prepared for radiation testing by simply removing the cover over the die. Thus, we do not discuss these directly. Instead we focus on the modern commercial devices, which are very challenging to prepare for the beam.

3.3.1 Board Selection

Hardware selection for an SOC test is very important. During exploration of this subject a considerable amount of effort has gone into board selection and preparation. Some of the methods developed proved to be particularly useful while others proved to have pitfalls. These are discussed in this subsection, based on some example efforts that reflect the research conducted to develop this guideline.

One of the subject devices is the P5020 SOC, which is a more complex device than the P2020. This device has 1295 pins and provides 18 serializer/deserializer (SERDES) lanes. Radiation characterization efforts usually cannot afford to create a new test board to test this type of device. There are usually a few options for off-the-shelf test boards for most devices. An exception to this is the case where the manufacturer is collaborating and willing to help develop a test board.

A few key qualities of test boards are desired. These often do not all come together, but it is very helpful to know which do and which do not when selecting a board.

1. Utilize inexpensive demonstration kits. Testing generally requires limited functionality as test hardware must be built for each test.
2. Use boards that have the SOC socketed (modern multi-GHz processors are socketed, so performance should not be an issue here).
3. If the part cannot be removed from the board, ensure the board can fit in any apparatus that will be used to machine or acid etch the DUT.
4. Ensure there are no active components under the DUT.
5. If possible get a board where components are located at least one inch (2.5 cm) from the DUT (so that the board can also serve as a proton test board, or will be ready for high energy heavy ion facilities).
6. Consider using a different test board for proton and heavy ion tests as the difficulties of each may be more amenable to different boards.
7. If a custom board is being made be very careful about constraining cost as radiation test requirements and desires may result in a design that is significantly outside of normal design methodology.
8. Try to obtain a board where thermal management will be easy. This includes easy changing and positioning of heat sinks.

3.3.2 Physical Manipulation

Because of beam range and thermal control issues, physical modification of DUTs is both necessary and very difficult. Four types of DUTs have been encountered in development of this guideline and serve as useful examples.

The Cobham Gaisler UT699 is an example of a relatively simple device to test. It is a face-up chip in a ceramic package with a metal lid that can be directly removed. Because it does not require a heat sink, after removing the metal lid the die is ready for exposure.

The OPERA Maestro ITC is an example of a bare flip-chip where the die is exposed. In this case, the die is the only material that must be traversed by the beam. A common approach to preparing this type of device is to thin the device to enable high LET testing. This type of device could be tested with higher energy

(>25 MeV/amu) low-Z ions that typically have at least 1 mm of penetration in Si. However, only a few ions can be used for testing without thinning the device. This means that unless a lot is known about the process regarding SEL sensitivity, and all SEE curves saturate at low LET, a complete test cannot be performed without thinning. Thinned devices may be more problematic when testing in vacuum. However at facilities such as TAMU the thinned devices can usually perform in air even with high LET particles.

The Freescale P2020 is an example of a device with a heat spreader where the actual die is not a flip-chip. For the P2020, if the heat spreader can be removed (or milled directly over the die) then only a thin region of fill material will hinder the beam getting to the die. The P2020 is very low power when resources are not enabled, and can often be tested without thermal control.

The Freescale P5020 is an example of a modern device package where a flip-chip device has a copper heat spreader mounted on top of it. The P5020 is very sensitive to thermal control issues. Because of this the heat spreader cannot be completely removed. This means that extreme caution is required with heat spreader milling and thermal control using modified DUTs and heat sinks during testing. It is recommended that only small regions of the heat spreader be removed over known subsystems of this type of device with assistance of manufacturer information. It is unlikely that the P5020 can be reliably operated in a vacuum test system, however the beam must also irradiate through the substrate, so higher energy heavy ions are required.

3.3.2.1 *Altering Board to Socket DUT*

Modification of a DUT can be considerably simplified if the DUT can be modified off of the test board. FPGA testing can serve as an example for SOC testing. In some FPGA testing, evaluation boards are altered by removing the DUT, putting down an array of pin sockets, and adding an array of pins to each DUT. Special tools and care are required for installation and removal as these sockets are not zero insertion force (ZIF).

3.3.2.2 *Flip-Chips and Heat Spreaders*

Flip chips with heat spreaders are the modern type of microprocessor and SOC mechanical structures. An example of this type of the device is the i.MX6Q processor shown in Figure 3-1. This type of package enables the factory to control the thermal contact with the immediate heat sink (the heat spreader). For full operation the heat spreader must be connected to a larger heat sink with a fan. A drawing of this arrangement is given in Figure 3-2. The thickness of these heat spreaders is usually more than 0.5 mm.

According to the Stopping and Range of Ions in Matter (SRIM) application [89], Ne at 40 MeV/amu (TAMU ion) has a range of 1.68 mm in Si (see the TAMU plots) and a range of 0.55 mm in Cu. Heat spreaders are usually quite thick since they are bulk material, and easily exceed 0.55 mm. Thus, for heavy ion and low energy proton testing, it is necessary to thin or remove the heat spreader in any device constructed this way.

3.3.2.3 *Thinning Devices*

Machines exist for milling the Si substrate of a target IC. An example machine is the UltraTec ASAP 1 [90]. These machines require the device to be loose so that it can be mounted in the thinning apparatus. There are several groups that can perform this thinning, including JPL.



Figure 3-1. i.MX6Q processor. Note that the cover is a heat spreader, which ensures good contact between the flip-chip die internal to the package and the external heat sink [113]. Image used with permission.

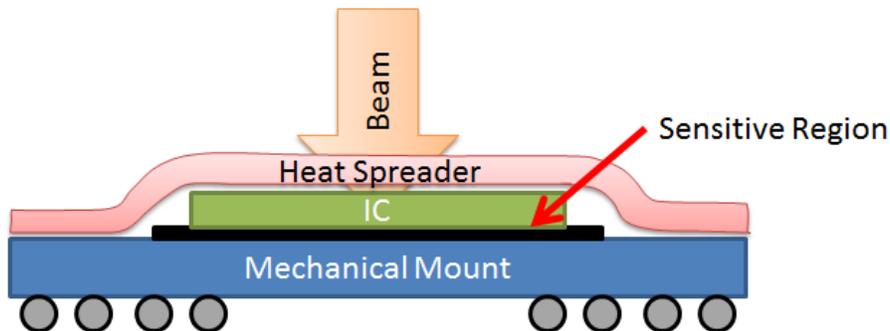


Figure 3-2. A drawing of the mechanical structure of a flip-chip IC with a heat spreader. The sensitive region is the portion of the IC that is closest to the mechanical mount.

Typical thinning can achieve 35 to 120 μm of remaining substrate (numbers below 80 μm require high precision in the machine tools). 80 μm thickness is enough to enable testing at normal incidence with LETs up to 90 $\text{MeV}\cdot\text{cm}^2/\text{mg}$ (which have ranges of at least 80 μm at some facilities). Testing at angle requires higher energy beams as the thickness of the substrate at angle will stop most of the 15-MeV ions (and lower energy), which can only cross about 50 μm of material if a device is rotated to 60° (since the path length is 100 μm), as seen in Figure 3-3. Thinning can go below 80 μm and possibly get as thin as 35 μm , but thermal issues will become very important and this guideline does not regard anything below 80 μm as reliable at the stage of planning. It is possible that during test preparation it may be shown that a device can operate reliably for testing at thicknesses of 50 μm or below. If this can be achieved, then the resulting devices would be good for SEE testing with the possibility of changes to the deep charge collection mechanisms. However, it is recommended that tools such as ultrasonic thickness measurement devices be used to verify very thin devices to ensure they meet the test plan requirements.

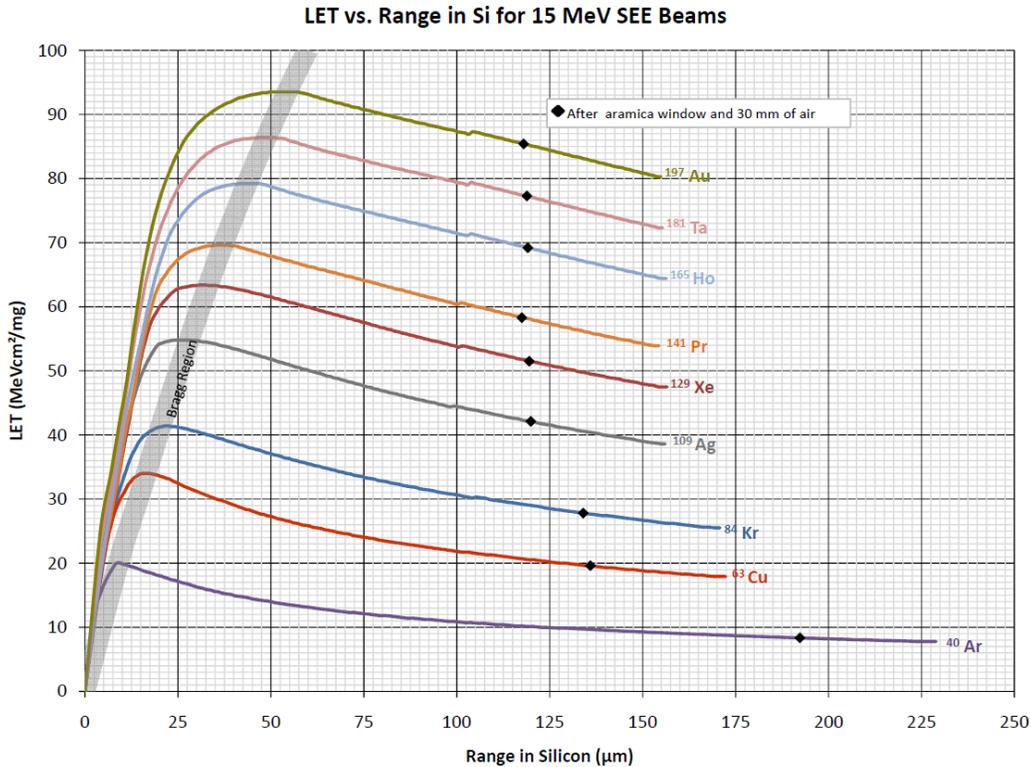


Figure 3-3. The ion beams available at TAMU with 15 MeV/amu. Note that from the poly (p-phenylene terephthalamide)-aramid (Aramica) window the higher LET ions will be near the top of the Bragg Peak after 80 µm of Si [77]. Image used with permission.

3.3.2.4 Opening Heat Spreaders on Loose Devices

Heat spreaders, such as that in Figures 3-1 and 3-2 must be removed over any region of the die to be tested. When the device is loose or un-mounted there are several options that can be used. The key about loose devices is that they can be mounted solidly in a carrier that is then locked to either a mill or a custom mechanical tool (such as the UltraTec ASAP 1 discussed earlier).

If the heat spreader is to be completely removed, a simple mill can be used to carve the heat spreader at the point where it makes the lip for mounting on the lower portion of the device (use Figure 3-2 as a reference). As discussed earlier, however, thermal issues can make this approach unreasonable because the die loses all contact with the heat spreader.

Alternatively the heat spreader can be partially removed directly over the die. This approach can only be performed reliably on a loose device because the mounting of the part in the milling machine must be much better than can be achieved on parts mounted to boards. When milling heat spreaders in this way, care must be taken to avoid milling bits coming into contact with the die. Because of the properties of copper it may be possible to leave a small amount at the bottom of a carved hole and then peel the remaining portion out with tweezers. (It may be possible to mill a heat spreader on a part that is mounted to a board. We believe the flexing of the mounted board will result in vibration and uncontrolled plunging of the milling bit through the heat spreader and into the die. This method cannot be recommended but may be possible to perform without damaging the device. Success may depend on how big the mounting board is, its stiffness, etc.)

3.3.2.5 Opening Heat Spreaders on Mounted Devices

Heat spreaders can alternately be modified while mounted on a test board. The board provides a very difficult flexible surface with spring-like behavior. Because of this, we found that very sharp milling bits

should be used for the last pass on the heat spreader. The board must be solidly mounted to the mill's table. And a mechanical support must be used to hold the device against vertical motion (this means producing mechanical supports that contact the board from above and below – or, alternately, mount to the board) as close as possible to the DUT location. If the mechanical support cannot stop the board flex, preloading board flex should be applied to reduce possible DUT motion. Figure 3-4 shows a DUT that had its heat spreader removed over the SOC while mounted on the test board. Alternately, the method indicated above in Section 3.3.2.4 may be employed, where a thin layer of copper is allowed to remain, and is later removed by cutting and peeling with tweezers. This latter method probably requires multiple devices to develop the methodology for the given device – and may benefit from using a dull milling bits when the material becomes thin.

3.3.3 Thermal Control

Thermal control is very important. In extreme cases loss of thermal control will destroy test boards. Because of modifications, even thermally stable systems may not work correctly after modification of heat spreaders or modifications to heat sinks. With facilities that have beams in excess of 15 MeV/amu, testing in air is often done. But the user may also desire to test in vacuum, or even elevate the temperature of the DUT, which are described below.

The approach selected for thermal management should depend on what is needed for the DUT. In some cases we observed that the heat generated depended on how much of the on-chip resources were used, and the frequency used to run the device. Higher frequencies and increased resource usage both increase the heat generation rate. After thinning, or partial or complete removal of heat spreaders, the response can change, so it is recommended to design the thermal management approach around the physical setup for testing.

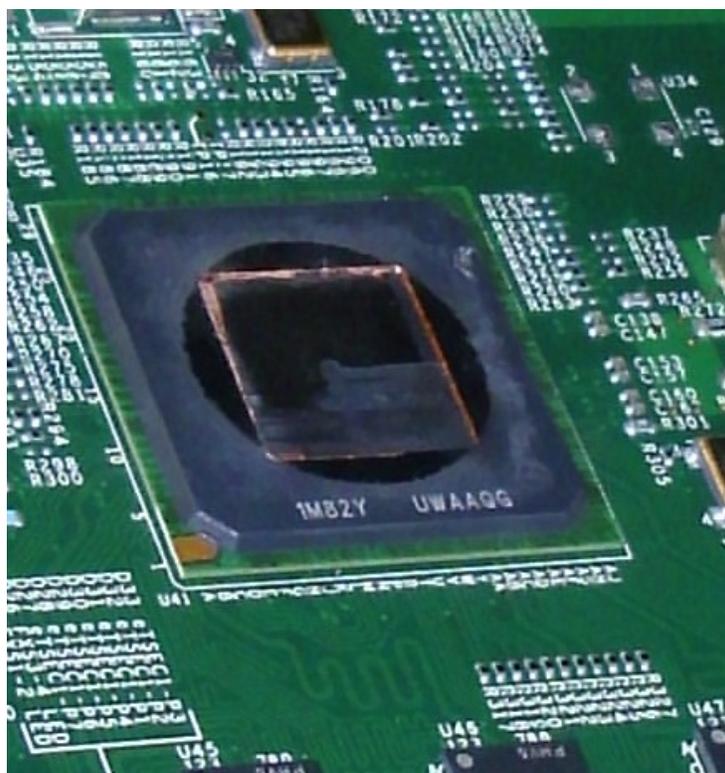


Figure 3-4. A device with heat spreader milled while on the board. The copper ring shows the heat spreader that remains. Photo used with Jet Propulsion Laboratory permission.

A similar thermal management problem was encountered for Irom et al. (2002) and Howard et al. (2001) [36 and 40, respectively]. Two approaches to heat sink modification are recommended. The first is to cut a hole in the heat sink that entirely exposes the region above the die. The other approach is to make multiple heat sinks exposing subsections of the die and use a test regime that includes test runs across the various modified heat sinks (see the P5020 example in Section 6.4.5.2). In the specific case of [40], Howard et al. found that thinning the die resulted in devices that would fail due thermal stress when running, and that ultimately it was necessary to construct a special heat sink that was thin enough that the beam had sufficient range to cross the heatsink and the unthinned device.

In most cases an arrangement can be made where hot DUTs can be thermally controlled by direct application of cool air—though in some cases the air is compressed and may pose a mechanical risk to delicate bond wires. Some facilities can provide cool air that can be directed at the DUT. Alternatively a cool air blower of some type can be constructed or bought, such as a cold air gun. For some beam facilities such a system is available for users, so the test engineers should include this when discussing use of the facility.

Because of heat sink traversal, conductive cooling, and limited ability to thin some devices, this guideline considers testing in vacuum to be somewhat out of scope (usually, if the device is thick or a heatsink is needed, ion ranges must already be high, and vacuum testing does not offer a significant advantage over testing in air). Furthermore... testing in vacuum with fully exposed die is the more common test method. If a DUT can be prepared with the die fully exposed and it does not have significant thermal problems, other test standards can be used to assist in performing testing in vacuum. The thermal management challenges will be different for different DUTs and is usually driven by the power consumption of the part being tested.

Thermal management may also include desire to heat devices, such as for SEL testing. In this case the test engineer must be very careful, especially with physically modified DUTs. For most SOCs enough heat is generated that simply reducing the effectiveness of the cooling system will be sufficient for achieving a safe but elevated temperature. For other devices, a heater may be attached on the back of the board. It is recommended to use an optical thermal detector, or an on-die temperature sensor, to observe the temperature of the die in order to establish an operating temperature.

If the test must be performed in a vacuum, the thermal management will be more complex. For most of the devices discussed here, operating in a vacuum is likely to be very difficult, and we did not explore the implications.

3.4 Detection of Errors

Error detection and identification is required for counting of errors to determine sensitivity for various upset types. The types of errors that can be detected depend on the structure of the test algorithm and how it classifies observed errors. Test algorithms are discussed in the next section. Here we focus on how the hardware and software can be used to detect errors.

3.4.1 *Expected IO Patterns*

One method of detecting errors is by observing the actual IO patterns at the pins of the DUT. Cycle-by-cycle capture can be compared to the expected pattern. Thus, any deviation can be identified. The structure of the incorrect IO signals can be analyzed for identification of errors, or all deviations from expected behavior can be used as an upper bound for device errors. As indicated in Section 1.3.3, however, this type of test is of limited benefit when the circuit becomes complex and due to either fault tolerant (FT) or other mechanisms becomes non-deterministic.

3.4.2 Hardware Interrogation

Various tools can be used to interrogate the hardware to observe upsets. The two most common methods are JTAG tools and custom manufacturer debugging hardware. JTAG tools are somewhat standard but must be tailored to the DUT. Manufacturer hardware may exploit special debugging ports.

Hardware interrogation using these sorts of tools is done by either single-stepping through executing code or by breaking into running code. The former can be observed and verified on a step-by-step operation, while the latter can be used to check the state of all the resources (registers, caches, peripheral configurations, etc.).

Because of how slow these methods are, their limited use of resources, and questions surrounding the impact on fault tolerance testing, direct hardware interrogation is somewhat limited in its usefulness for testing.

3.4.3 Software Detection

Error detection can also be accomplished by test software monitor operations to automatically detect errors. There is considerable work on methods for fault-tolerant software. Much of this is designed to catch software bugs, but many of the concepts can be used to increase robustness of test software. Regardless of robustness, when errors are detected, software can count, report, and attempt to carry on. If the software cannot fix the error, then hardware interrogation can be used to try to determine what error the software failed to fix.

3.5 General Approach to Testing

This guideline suggests that test designers have the DUT test itself by self-sensitization and internal detection of errors [18]. In addition, other methods can be used. The main lines of test approaches developed in the past also include golden chip and test vectors [16,91]. Methods being employed today also include utilizing hardware debugging systems to interrogate DUTs [30,43].

An important point to make regarding testing is that you primarily only test for upsets that you expected to get. But those are not the only upsets that will occur. The algorithm section below will provide recommendations for how to build test algorithms to enable detection of unexpected phenomena. However, it is important to keep in mind that careful analysis of anomalies will be required either at the beam or later.

Radiation testing tends to target the response of key elements or operations because they can be set up to provide unambiguous results. However these results are typically not directly applicable to user applications. If a user application is being directly examined, it is important to have clearly defined application upset modes and have a clear description of how element-level upsets translate to the application. If it is unclear how to do this translation, then the customer program should be informed and requested to provide application-specific software and hardware rather than testing key elements.

3.5.1 Test and Validation

As suggested, other methods for testing can be used. In this section we review the most common methods. These are intended to provide general information about the SOC's radiation sensitivity. Any test method should be assumed to be only approximate until actual SEE testing is performed and the real device sensitivity is observed. (Note that lasers or other fault-injection methods can help minimize the unknown responses going into an actual SEE test, but cannot eliminate them.)

3.5.1.1 Golden Chip or Test Vectors

Early microprocessor testing was based on knowledge of exactly what the microprocessor was supposed to do on a cycle-by-cycle basis [16]. This method usually requires developing a custom circuit for the device. From a hardware perspective, we have pointed out that custom development is not viable in most cases. So, this is one reason this method is not covered in detail here.

From an error detection perspective, it is also true that modern SOCs have moved more and more of the circuit behavior inside the chip. This means that identifying cycle-level differences is not necessarily useful for identifying errors. It is expected that in a complex device the arrival of I/O signals may be offset by small amounts of time and that on-chip communications may include crossing a clock domain boundary. That is, these devices are not externally synchronized on a system clock in the way that older devices were. A good example of this might be a loop-back test on an on-chip Ethernet port. The Ethernet port and the test central processing unit (CPU) run on two different clocks that are not guaranteed to be in specific phases at the start of a test. Thus, the behavior of any communication to the Ethernet port is not deterministic unless a larger number of parameters are known that can easily be controlled.

Further, modern devices have some built-in fault tolerance that is designed to fix errors as they happen. Some of this error correction may result in a delay of a clock cycle or more due to flushing the pipeline or reloading a cache line. The golden chip test approach will identify any deviation from an expected I/O pattern as an error, even if it is due to a fault that is being recovered. For SEE testing, we desire to not count errors in fault-tolerant portions of the device that would be silently corrected. Thus, this type of testing it is unlikely to be reliable on a modern device, though there may be specific cases where it would produce useful results.

3.5.1.2 Device Tests Itself

The method advocated in this guideline is to have the DUT test itself wherever possible. This approach entails writing test software that can sensitize target structures on the device and interrogate them to observe upsets. This approach is very efficient for data collection when it works, but it can be prone to bugs that only turn up at the test facility. Although these bugs can be problematic, no other method is expected to showcase the bugs.

A couple of options for having the DUT test itself exist. The option recommended here is to use very low-level code to sensitize and collect results on fundamental building blocks of the DUT structure, such as SRAM cells in caches and registers, any special latches such as may be used in registers, and any other specific structures of interest. In the event that test personnel have data on test structures, it may still be useful to develop test software that can sensitize these structures in order to verify performance – but those in this situation may also opt to skip testing low level structures and target device subsystems instead.

Other approaches can include writing special test programs in a high-level language (such as C or C++), or using the error catching features of an operating system. These approaches are usually not analyzed through the operation of the software or the system. That is, testers do not know exactly what is going on because it may be a monumental effort to determine. The hope is that the test program will produce results indicative of how an application may perform. The test programs may report errors in calculations. And the operating system may report kernel panics or “blue screen of death”. These types of events can be tallied and used to construct cross section curves. In terms of machine operation, there is a lot of stuff going on in the C/C++ or operating system error catching that cannot reasonably be analyzed. Most user programs rely on kernel-level routines to such a high extent that they are just as likely to have the operating system crash as they are to detect and report an error.

In some cases, people write test programs intended to simulate space operation, or test actual flight software. This is sometimes called “test as you fly”. In these cases, the observed errors are more likely to be the same as those that will be encountered in space. The primary difficulties with this approach are: ground testing is accelerated and may confuse analysis; it may not be possible to simulate the application if it is unknown; and the test results may be too specific to the application and not of general use if the application changes.

It should be noted that in many cases it is possible to write test programs that have outputs that can only be correct if the program function nearly 100% correctly, but the actual outputs are relatively simple to examine. Examples of this are cyclic redundancy check (CRC) calculators and certain types of sequence

evaluations (particularly those that do not converge or diverge). These allow the test program to cover perhaps millions of operations, but only get compared once at the end.

3.5.2 Customer-Based Testing

The previous section examined general testing of SOCs. In some cases, however, it may be more appropriate to perform testing specific to the customer's needs. In this case the actual user application software and hardware are recommended for performing testing. As indicated above, this approach can be recommended in the event that a translation from low-level upset modes to system-level errors is unknown.

When performing a radiation test with customer hardware and software, it is very important to be aware of a couple key pieces of information. First, the behavior of the system may be very unpredictable at the test facility. Second, the data gathered will likely have very little applicability to any low level structures, and as such will be limited in scope of use to the given project. The upside of customer-based testing is that the system-level event rates are known, rather than needing a model to go from low-level upsets to system-level errors.

3.5.3 Preparing for Unexpected Errors

Unexpected errors are likely to occur during testing of a complex device, and SOCs are a good example. Several recommendations for being prepared for unexpected errors are given in the recommendations section below. The main items of importance are to bring appropriate test personnel and bring tools that will enable debugging to remove bugs that show up during testing. The most important thing to bring to the test, regarding unexpected errors, is a healthy appreciation for the possibility that such upsets may plague the collected data. A good plan for collecting data in spite of the impact of unexpected errors is essential.

3.6 Radiation Test Approaches for SOCs

In Section 3.3 we discussed the hardware setup for SEE testing of SOCs. The hardware and operation approach recommended is to have the SOC sensitize itself for radiation. Once sensitized, the SOC is exposed to the desired radiation environment. Several data collection and maintenance approaches exist for running this type of test.

1. Utilize debugging tools to examine the interim and/or final state of the SOC after exposure (often these tools can also be used to sensitize the device as well).
2. Have the SOC integrate SEE phenomena, and after exposure, examine the device for upsets (e.g. collect upsets in an unused on-chip memory).
3. Perform periodic readout of SEE targets to observe upsets, then reset the device for the next period.
4. Have the SOC perform operations constantly, and observe any errors in execution.

Note that approaches 2–4 can also be examined with debugging tools especially when they result in hanging of processor cores. These debugging tools may be able to determine if there is a particular vulnerability in a test approaches.

Although it is expected that test algorithms and flight algorithms may have significantly different SEE response, it is not generally possible to obtain flight algorithms for testing (primarily due to export rules and intellectual property issues). Thus, test engineers must make due with algorithms that showcase the SEE sensitivity of the device, and then make (conservative) assumptions as to how the results apply to the flight software.

3.6.1 General Algorithms

When designing an algorithm for sensitizing an SOC, the following aspects should be taken into account. The algorithm can be a small portion of a larger piece of code that prepares the SOC as desired. The algorithm should be relatively short, preferably smaller than the size of the L1 cache. And, the algorithm should employ design practices that will enable reliable identification of target events while enabling debugging of errors and examination of anomalies.

As a general rule, the following elements should be employed in any SOC radiation test algorithm [18]:

1. Use a test cycle approach that sets up targets, allows upsets, and collects and stores upset information for immediate or later reporting.
2. Eliminate dependence on operating system—test code should disable operating system control upon execution.
3. Eliminate use of libraries and compilers—the former add code you have no control over, and the latter add structure and register dependence.
4. Limit use of caches for test-critical needs—keep important settings or data off the processor.
5. Store data off of the test board or in nonvolatile memory (NVM) to ensure data recovery.
6. Prepare a safe test code area—a region of memory where the test code can reside away from data and other important code elements.
7. Move the test code to the safe test code area before starting a test algorithm.
8. Service all traps, exceptions, and interrupts—observe and report occurrence. But it is not necessary to properly recover from all events.
9. Prepare the systems that interact with the test device to automatically record all useful data under normal and crash situations—there may be significant complexity which is difficult to carry out correctly by exhausted test engineers.
10. Scrub configuration registers and key information periodically.
11. Recover I/O devices on error. They may have had their configurations altered.
12. Restart the test cycle after an error occurs.
13. Verify test code integrity after errors—that is, scrub or recopy to remove errors.
14. Use a watchdog monitor to provide an alternate method to recover from a crash.
15. Limit use of memory management units (MMUs).

It is not always possible to implement all of the elements above. But a subset of them can provide a basis for a robust test algorithm. The items are listed in order of priority.

We expand this list by considering the addition of streaming test data. When possible, it is recommended that the test log be stored in nonvolatile RAM or be streamed off of the test system as it is generated. Early work with microprocessors did not stream this information off of the test computer for two reasons [1]. First, most test algorithms try to leverage off of micro-kernels to provide I/O operations; but this operation is often significantly more vulnerable to SEE than test code, so reporting is minimized. Second, serial I/O (i.e., RS232 or RS422) is inherently very slow and can cause problems with normalization of beam, again favoring minimizing of reporting.

Periodic transfer of SEE and error data is recommended to provide data up to a crash. But the two problems above should be kept in mind. In particular the amount of transferred data during test algorithm execution should be minimized, using error encoding and packing (as opposed to verbose messages).

3.6.2 Software to Operate SOCs

Several standard software approaches have been used to examine microprocessors under the “processor tests itself approach.” These algorithms largely carry over to SOCs except that they should be expanded to enable examination of the hardware interfaces and additional on-chip resources presented by the SOC.

Software test algorithms come down to two basic types. The first is static testing, where targets are prepared and monitored (continuously or periodically) for unexpected changes. An early use of this approach is described in Bezerra and Kuitunen (2003) [38] for registers and caches. It is also discussed and used in Irom (2008) and Hafer et al. (2009) [1,52. respectively]. This is also sometimes referred to as “semi-static” since the processor is still executing code. (True static testing is possible, and an example is described in Guertin et al. (2012) [30]. This can revolve around using JTAG or other hardware means to directly load and read values from the DUT.) The other basic type is dynamic testing.

For both types of algorithms a couple details should be known or estimated. All targets for which SEE data is being collected should have the number of sensitive bits recorded along with the data pattern stored in any static or configuration elements (users can read some configuration registers before and after irradiation). The time profile of the execution should be known (i.e., the number of instructions executed per loop or unit time and the number of accesses of the target structure per unit time). For dynamic testing it is very important to know how many of the relevant instructions are being executed per loop or unit time. For loops used during testing, their execution periods should be known.

Several common test algorithms have been used in recent testing of complex digital parts. These are discussed in more detail in the following subsections.

3.6.2.1 Static Element Soak

The static element soak test algorithm is useful for characterizing the cross section of storage elements. The structure of this algorithm is shown in Figure 3-5. This algorithm is also discussed in Irom (2008) [1] and Bezerra and Kuitunen (2003) [38], but the version suggested here differs from others in that periodic analysis of the static targets and reporting of periodic data is recommended. This recommendation comes from experience with algorithm crashes with no partial data collected—which is unnecessarily wasteful of beam time. However, unless the footprint for data collection and reporting can be kept small (specifically no libraries or operating system calls made) this algorithm can be prone to crashes. Typically the reporting is carried out by direct control of a Universal Asynchronous Receiver/Transmitter (UART) port if periodic reporting is used. If UART control is through kernel or library codes, reporting should only be performed at the end of exposure, or only a few times during exposure.

The mechanism for determining if the test algorithm is done is kept intentionally vague, and is left up to the test engineer to implement. If external input can be done with low-level I/O operations (i.e., direct control over the port) this is the preferred approach to determining if the loop is complete.

Beam should be applied while the test algorithm is executing the inner loop. During execution of the inner loop, any strange occurrences should be logged (to internal memory) for later reporting after the loop is terminated.

Once the main loop is complete, tools built into the test algorithm can be used to downlink any records of anomalies that occurred during exposure (such as exception records). It is also helpful if the test algorithm can downlink data from the variable storage for the test program (i.e. the state information for the test program can be read out), but this may be too cumbersome to put into a given test program.

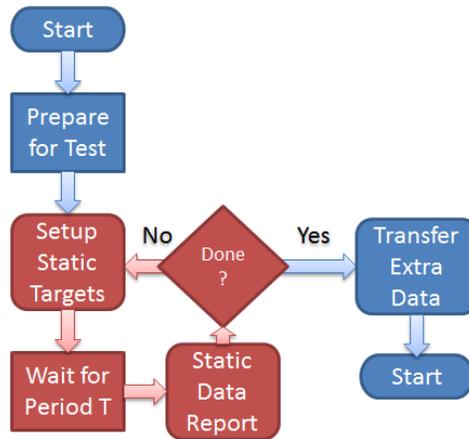


Figure 3-5. Flow chart for the "static soak" test type. A periodic cycle of setting up static elements and later checking them for upsets makes up the main part of the test loop. Irradiation should be carried out during the inner loop.

3.6.2.2 Testing EDAC-Protected Targets

Upsets in EDAC-protected systems can only be observed if one of three things is occurring. First, and easiest to deal with, is if the EDAC system has registers to track EDAC corrections. If this is the case, then the number of events can be reported by having the test algorithm periodically check the EDAC report registers. Second is if the EDAC can be disabled, so that a standard test algorithm can be performed. Third, is where the EDAC system is completely transparent on single-bit errors (i.e. it fixes them with no report).

In the last case, only double-bit errors (and higher) can be observed. In most DUTs this will require uncorrelated ion hits to the same EDAC word within a detection period (assuming EDAC protected elements are being scrubbed—if they are not being scrubbed they are essentially integrating beam during the entire exposure).

Most EDAC systems can be configured to cause an interrupt on a double-bit error. Unfortunately, this is not terribly helpful for SEE testing because double-bit EDAC error handlers usually crash the processor in a full operating system (OS) (this would be the desired behavior because the error would be unrecoverable in most cases). It is also not recommended to put SEE detection into interrupt handlers because this dramatically increases the chances of creating a crash behavior that does not need to be in the test code (if a latent machine problem [for example a double bit error not previously checked] is found during this interrupt handler, the affected core will crash).

Instead, it is recommended to disable the EDAC double-bit error interrupt when testing EDAC systems and intentionally design the test algorithm to not scrub the tested portion of the EDAC system (or do it very slowly) while rapidly scrubbing any portion of the EDAC system that is relevant to the test system. In this way the system can still be protected, but the double-bit errors will give detectable SEEs that can be used to determine the sensitivity of the protected bits. This is done by backing out the double-bit error data to determine the single-bit error sensitivity that would give the observed double-bit error results.

3.6.2.3 Targeted Subsystem Stress Test (Dynamic Test)

Static testing is very useful because in most modern digital devices the majority of the die is used for storage of bits of various importance for the application, with a large portion being for caches. The actual functional portions of the device are also important to understand from an SEE point of view. Thus the natural counterpart to the static test is the dynamic test, which focuses on stressing a subsystem of a device and looking for SEEs that cause the subsystem to perform incorrectly.

Common subsystem stress tests focus on achieving a high duty cycle for the use of that subsystem. But it is important to note that at any given time a modern microprocessor is working on no more than ten individual instructions. In order to enable these instructions to execute out of order and reach this level of parallelism, a lot of predictive logic is used. But fundamentally, the processor is only performing ten out of hundreds of operations it is capable of doing. Furthermore, each operation it is doing is split into several steps, executed over multiple clock cycles with precise control over the ways that different parts of the execution can impact the system. This severely limits the impact of SEEs in the functional portions of the processor.

A key difficulty of this type of test is that it is important to know the mix of all instructions the processor is executing in order to focus on what operational portion of the chip is causing an SEE response (this is important because, for example, FFT tests are often actually spending more time on memory accesses than actually performing calculations – which is not the actual intention of an FFT test). But this is nearly impossible in a modern processor without a hardware-level simulation because speculative execution and out-of-order execution can both result in the microprocessor executing code that differs from what is written (though the result will match the intention of the code). The reason you need to know what the code is doing is to be able to map results of one type of application to another. This is the only reliable way to try to put limitations on flight software from results for test software – but it requires going through simulation of flight code execution and comparison to instruction mixes in the test software.

For microprocessors, targeted subsystem stress testing should be performed during static testing in order to examine the possibility that it is important. As an example, in Hafer et al. [52] it was observed that actively using the L1 data cache increased the SEE sensitivity of the cache. If active stress testing is worse than static testing, then stress testing should be used.

It is not presently known if functional blocks within an SOC exhibit this same type of limited impact of dynamic elements. For example, can you determine the SEE sensitivity without actively operating the functional blocks, or do you need to run some sort of stress test? This is a region of continued investigation.

3.6.2.4 Fast Fourier Transform

One of the most common algorithms used for radiation testing of microprocessors is the fast Fourier transform (FFT) [38]. FFTs are used to quickly calculate a discrete Fourier transform, which is used to change data from position space to frequency space (and thereby enable analysis of patterns). The FFT is an algorithm heavily used in the space community for data analysis and thus represents an activity that is very close to flight code. Unfortunately, though, it is not terribly useful for determining the SEE sensitivity of structures like the floating point unit (FPU). We will discuss this briefly and highlight the issue.

The FFT is actually a fairly straight-forward way of mixing data through a product and add with unit vectors evenly spaced around the complex unit circle. The basic function is given in Equation (3-1).

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad (3-1)$$

where (X_i) is the FFT output vector, (x_i) is the original vector, and N is the order of the FFT. It is clear that calculation of Equation (3-1) is mathematically intensive, and it would appear the algorithm would have its instruction distribution heavily weighted towards FPU operations. But this is only true if the algorithm is very carefully structured for the processor that is being tested, and then it would have to almost be hand-assembled to be primarily exercising the FPU. The reason is the following. The main operation that the processor needs to do is multiply and add (this is a common single FPU instruction that combines three FPU operands and outputs a fourth, such as the PowerPC “fmadd” instruction). But it needs to do this on a set of numbers. If the set of numbers is small, then the algorithm can be focused to have little operation other than multiply-add, but the algorithm will not run very long before switching to the next data elements, because FFTs perform a number of calculations that scales with N^2 .

If N is large or if the algorithm is not highly optimized (optimization of this type is not expected even in a fairly good compiler), then the x vector and the unit complex vectors are just kept in the local cache and fetched for each iteration of the loop. Thus, if N is say 16, then the FFT focused on one data vector x will execute 256 multiply-adds of complex numbers, more than 256 counter increments, more than 256 counter comparisons, and 512 fetches of complex numbers. Once corrected for the numbers of instructions required for complex number operations, FPU calculations account for about 1/3rd of the instructions that will be executed by the FFT (under reasonable optimization). A diagram of this algorithm is given in Figure 3-6. The reality of how the code compiles may be very different than this distribution of instructions. If complex multiplication and addition is carried out through function calls (as would be the case with a c++ complex class) the instruction mix would include many instructions to setup and return from the subroutine call.

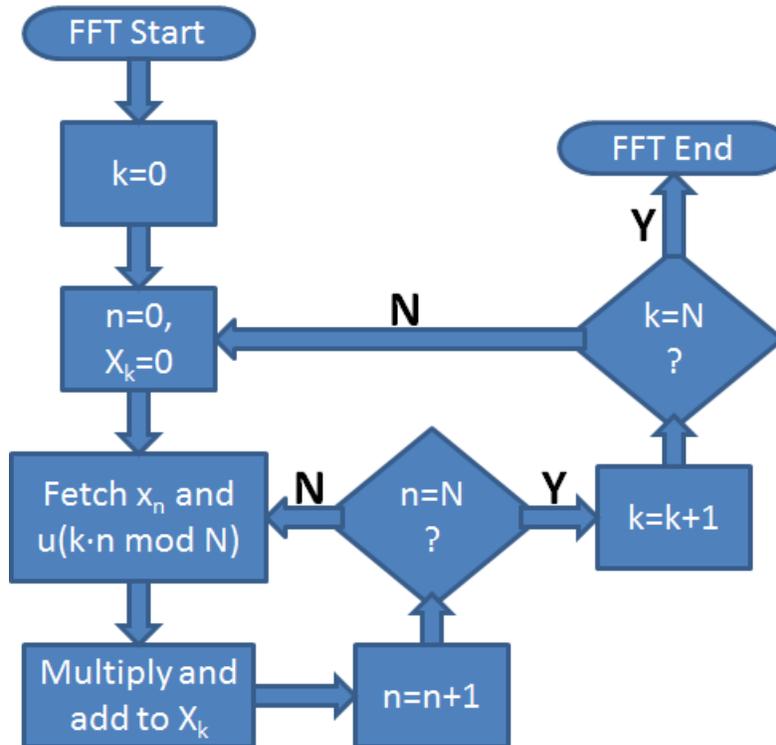


Figure 3-6. The actual algorithm executed for an FFT operation. Note that the inner loop (over n) includes four operations: complex loads (2) complex multiply-add (1), counter increment (1), and a compare/branch (1).

3.6.2.5 Dhrystone/Benchmark Tests

Static, targeted, and FFT-like algorithms are the most common algorithms used for testing SEE in microprocessors. In this section we discuss using benchmark tests to enable radiation evaluation. The reason we cover this is two-fold. First, this type of test algorithm requires critical analysis to determine usefulness. Second, Dhrystone has made its way into some specific fault emulation efforts [92].

Dhrystone is a benchmarking algorithm used to evaluate how quickly a microprocessor can perform standard arithmetic logic unit (ALU) operations that are common in running software. The implementation of this for radiation testing requires some understanding of how it operates. Fundamentally the program is based on performing a somewhat standard mixture of software operations and seeing how long it takes to execute them. One version of the source code for Dhrystone can be found at [93]. The code is freely available and should be easy to compile on most systems.

Analysis of the source code shows that there are a lot of assignment, copy, and mathematical operations. In addition there are a lot of function calls. These are, of course, good for exercising the processor. However, there is no error checking along the way (in fact, the source code indicates that many of the operations are meaningless).

The use of this algorithm in Limbrick (2009) [92] was good because the goal there was to use embedded monitoring of the timing behavior of every instruction. This was accomplished using an FPGA design that enabled direct examination of the running code. This examination included checking the results of each instruction, which is generally not feasible for processor under test to do by itself, without considerable code modifications. The generic Dhrystone algorithm is not homogeneous in time and thus is not particularly good for observing radiation sensitivity. However, it can be modified to have a homogenous structure and the relative portion of the time spent doing each operation it performs can be used to create a basis for collecting and normalizing data. For the most part, all processor benchmarks need similar modifications in order to be useful for radiation testing.

3.6.2.6 Other Algorithms

The algorithms discussed above are not the only algorithms that can be used, and indeed new algorithms can be pulled in at any time to enable testing for key items. In fact, it is recommended that in many cases test engineers develop code appropriate to the type of test they want to run, as long as they can ensure that this code is more likely to characterize the target than to characterize the operating system or other well-tested parts of the device such as caches or support RAMs.

Test algorithms should detect errors while running. In a modern running microprocessor instruction-level visibility outside of the processor is not possible. To a limited extent running code can be examined by single stepping and examining cache, memory, and register contents. Thus it is important for the test code to verify that calculations are performed correctly in order to detect upsets.

Alternate algorithms beyond what have been presented above have been used previously for testing with varied results [40,41,94]. These usually target subroutine calling, memory transfer, or operating system operation, such as Microsoft Windows [41]. For SOCs, clearly additional codes should target the memory controllers, Ethernet ports, SERDES lanes, and other on-chip resources.

In all cases, the most important thing is to be able to establish the normalization factor between the test software and a real system. Depending on the operating system and its match to the microprocessor (how efficiently and completely it uses the microprocessor), the relationship between test results and real applications can differ by 10 to 100 times (with test results being worst case). And real application responses can differ from one another by 10 to 1000 times.

3.6.2.7 Basic Functional Testing/Crash Testing

Sometimes it is simply not reasonable to create custom software for testing of an SOC, or even when custom software is run the results are dominated by exceptions or crashes. In these cases the most useful approach for determining SEE sensitivity is to run standard software and observe how often the execution deviates from what is expected. In this type of testing the best that can be done is to make detailed records of events to enable careful examination, as in Howard et al. (2002) [94].

3.6.3 Beam Delivery Structure

Most beam sources provide beam relatively uniformly in time and space. Although the methods for beam delivery may be more ordered in time than true random particles, the systematic effects introduced in these facilities are minor. (In fact irregularities in the beam delivery—such as cyclotron buckets—often largely

cancel each other out due to relative insensitivity to any particular particle.)⁵ A good example is a SECDED EDAC system. The best application of this system is when upsets to a protected word occur infrequently enough (probability p) such that the probability of having two (probability p^2) is low. In the case of (spatially) uniformly distributed beam events there may be multiple upsets delivered in a particular cluster, but the associated upsets are not likely to be in the same EDAC word. And over several clusters, the upset delivery is essentially the same as if a random source were used.

Some facilities introduce considerable complications for analysis due to beam structure. Unless test algorithms and beam structure are both well known, the risk to the quality of reported data may argue against the use of these facilities. One facility like this is NSRL where beam is delivered for 0.3 s out of every 4.1 s. Two test algorithm time structures were used to test the Maestro ITC processor using NSRL, and they are shown in Figure 3-7 [30].

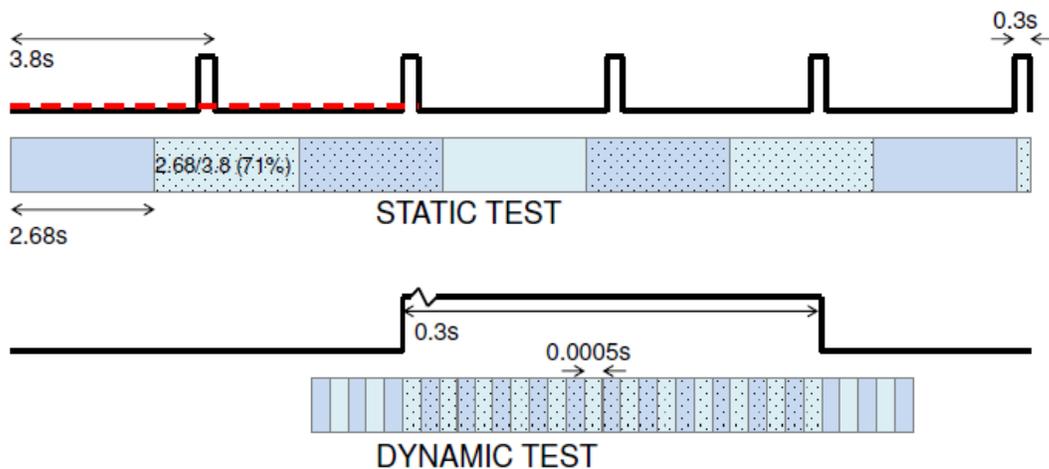


Figure 3-7. Time structures for two types of tests performed on the Maestro ITC device. The black lines refer to the beam structure (which is the same structure in each case, zoomed out bottom) while the lower dual-colored strip in each region refers to the test algorithm duration [30]. Image used with author's permission.

When performing testing at a facility with a beam that has gross structure such as NSRL the test algorithm and its time structure must be well known. The simplest test structure, the static test, has no time structure unless the elements are protected by FT. We refer to this case as “no time structure.” In the case of FT, usually the protected elements are not upsettable at the system level unless multiple upsets occur within a single element. In this case the FT protection requires periodic operation to remove upsets (called scrubbing). If this operation is very long compared to the beam time structure, then the impact is negligible. If the operation is very short compared to the beam time structure, then the impact must be calculated. For example, the flux to overwhelm FT is the instantaneous flux, not the average flux. The latter applies directly to the structure of the dynamic test indicated in Figure 3-7. The static test in this figure is an example of a test that lies in between the two cases and requires either pulse by pulse analysis or the acceptance of systematic uncertainty in the analysis.

The time structure of a dynamic test can be much more important for causing test anomalies due to the beam delivery being different than in nature. Dynamic tests are usually attempting to observe errors during operation of the chip. As indicated earlier, the advocated approach is to have the chip test itself. This naturally causes problems with beam structure artifacts. One example of how this can happen is when an upset is observed it causes a reporting algorithm to be executed, which is then sensitive to an SEE, but only

⁵ In nature, particle hits are Poisson-distributed with the time between events distributed exponentially in time. In cyclotrons they are distributed in time-buckets that arrive regularly.

because another SEE has just occurred. If the instantaneous flux is high enough, it can lead to an increased occurrence of injecting upsets while executing the reporting algorithms. This in turn can increase crash rates, depending on the structure of the reporting algorithm (usually they are not as robust as the test code because they are expected to run less often). Dynamic testing is also very similar to running an actual operating system and the types of problems expected due to beam structure are similar between the two.

3.7 Summary of Recommendations

This section covers a lot of details of testing. So for a summary of recommendations it makes sense to organize the list in a topic-based approach.

3.7.1 Board Selection

1. Use inexpensive development or evaluation boards to reduce cost
2. Select boards with socketed DUTs
3. Select boards with space between the DUT and other active components (ensures board is good for heavy ion and proton exposures)
4. Select boards with limited default peripheral implementation—they are cheaper and will probably run cooler
5. Select boards with the DUT within 6 inches (15 cm) of an edge of the test board—so there will be some orientation that can be positioned easily in front of the beam, and support angular studies.
6. Be aware that some of these requirements will reduce the performance and/or speed of the DUT, and a trade study may be desired to determine the relative importance of each variable
7. Sockets are desired but may significantly reduce the performance of the DUT

3.7.2 Device Preparation

1. Remove the heat spreader over target regions of a DUT
2. If possible, thin flip-chip devices to less than 100 μm
3. Prepare thermal management approach against actual DUT arrangement (after modification of heat spreader and heat sink)
4. Try to achieve thermal control with only blown air (not testing in vacuum). If testing in vacuum, this document may be of little help.
5. Be very cautious when powering up a modified device. Loss of thermal control can take less than 5 seconds.
6. Be aware that modified packages and/or device die may result in unexpected changes in SEE sensitivity, for example ambient light can affect SEE sensitivity of some circuits
7. Verify operation of devices with thermal management approach
8. Shutdown or do not activate unused peripherals or resources

3.7.3 Test Software and Algorithms

1. Do not use interrupt handlers as a primary means of controlling the test software (SEEs during interrupt execution can crash the processor)
2. Develop experience with hardware and software debugging tools

3. Do not spend too much time trying to learn how to debug the kernel or OS
4. Support as much of the general algorithm recommendations (in Section 3.6.1.) as possible.
5. Ensure data is transferred as frequently as possible, but keep kernel sensitivity minimal and reduce dead time due to I/O.
6. When FT exists, it should be verified
7. When FT exists, test algorithms must explore upsets beyond FT

4.0 TESTING

This section provides details relevant to test operations. The goal is to help the user perform the relevant measurements to produce the test data desired. This is done by presenting general test information describing how to determine the sensitivity achieved by the measurements, and by providing relevant information regarding problems that can occur. The goal of problem resolution is to ensure that valid data is taken, and that significant insight is not lost, while minimizing the impact of test problems on overall test scope.

4.1 General Testing

This section discusses difficulties involved in testing and test data collection. We focus on data collection that shows details covered in this test guideline. Here we focus on the microprocessor cores and general testing of the remainder of the SOC. Comprehensive discussion of targeted peripheral testing is future work for this task.

Ground-based testing cannot provide an environment that is exactly the same as the space environment. Instead, ground-based testing seeks to determine the precise device response to particular portions of the space environment. From this is developed a response curve that captures the SOC's response to the spectrum of particle types the SOC will encounter in space. The product of this response curve and the actual environment is then integrated over the various particles and energies that occur naturally. The final result is an expected event rate.

For ground-based testing, the SOC response curve is determined by testing with protons (energies up to 200 MeV for Earth orbiters, though higher energies can be used to simulate some portion of the heavy ion spectrum, or as designated by project requirements) and heavy ions (LETs up to 75 MeV-cm²/mg, or as designated by project requirements).

This section covers general issues. Then in the second part of this section we cover tester details and associated problems. Then we discuss debugging tools. And finally, we present a summary of the developed recommendations.

4.2 Beam Diagnostics and Data Quality

It is important to establish the quality of the data being taken they are taken. There are two primary problems that should be handled at the test facility. The first is that the beam normalization should be verified using methods with known behavior. One such method would be to bring a device with a known response to the beam and test that device in the beam before testing the primary test part. If this method is not available, at the very least the internal consistency of the dataset being taken should be known (i.e., the cross section vs. LET or energy curve), and the absolute response should be approximately predicted using information about the construction of the device, such as its feature size and the response of similar devices. The second thing is to verify the reproducibility of the data by testing with multiple devices or comparing to expected response as a function of LET or proton energy.

4.3 Effective Sensitivity

When testing a complex device for SEE many different error modes are expected. The total error rate is a sum of different contributions, some of which may be dependent on the test flux or fluence for a given test run. For a particular event type of interest the other events contribute in two ways. First, the other events may generate a background of test difficulties reducing the overall data collection. Second, the other events may masquerade as the event type of interest. If \mathcal{A} is the set of all event types and \mathcal{B} is the set of all observed event types, then the total rate for a given run i and event b in \mathcal{B} is given by Equation (4-1).

$$R_{b,i}(LET, \varphi, \Theta) = \sum_{a \in \mathcal{A}} R_{b,i}^a(LET, \varphi, \Theta) \quad (4-1)$$

Where R_x^y is the rate for event observation x from underlying event type y , and the total event rate is the inclusive R given in Equation (4-2).

$$R_{Total}(LET, \varphi, \Theta) = \sum_{b \in B} \sum_{a \in A} R_{b,i}^a(LET, \varphi, \Theta) \quad (4-2)$$

These contributions must be examined to establish the effective sensitivity achievable during a beam exposure. When taken in conjunction with several test runs the limitations imposed on a given run may be improved.

By specifically detailing the level of sensitivity that can be reached during a test, the test engineer can prioritize the data collection goals and not spend unnecessary amounts of time trying to improve a dataset that is fundamentally limited by testing sensitivity that cannot be improved.

4.3.1 Definition of Effective Sensitivity

We do not try to explicitly define the testing sensitivity for every case. However, it is important to define the general term and concept of an effective sensitivity. During a beam test, where a particular event type, b , is being examined under particular beam parameters \mathcal{B} , the effective sensitivity of the given test can be defined for that event type and those beam parameters as a limiting cross section σ_{limit} where Equation (4-3) holds.

$$\sigma_{b,meas} = \begin{cases} \sigma_b, \sigma_b > \sigma_{limit}, \\ \text{unknown}, \sigma_b \leq \sigma_{limit}, \end{cases} \quad (4-3)$$

There are two ways that the limit can come in. The first is by having other upsets overwhelm the test system and reduce the total amount of detectable events, possibly resulting in no events of the desired type being observed. The second is by incorrectly attributing events to the wrong underlying event type. For example, if we desire to measure upsets in the general purpose registers, we may not know if a detected bit upset is actually due to an upset in the target register, or due to a code execution error.

4.3.2 Determination of Effective Sensitivity

The most direct approach to use in establishing effective sensitivity comes down to analyzing two sources.

1. Determine the background rate for counts in the event signature of interest. (This is the rate where the test setup gives false counts of the event.)
2. Determine the amount of fluence at the lower-end of the reasonable flux window where the DUT will provide a difficult to deal with response (i.e., the fluence where anomalies start to show up even though the flux is low). (This is the rate where the test setup gives events that are so disruptive that only a few such events can be tolerated. These events are not the target event. An extreme example is an event type that permanently disables a board. A more subtle example is an event type that results in needing to spend a significant amount of time reprogramming a board before another exposure can be performed.)

In most cases only one of these two contributes to a given measurement (at a given set of parameters \mathcal{P}). In some cases, however, these two contributions may be of similar size and thus both must be examined.

The Maestro ITC testing provides a very good example of the difficulty involved in determining effective sensitivity. One event type examined during Maestro testing was the upset cross section for the tile processor registers (there are 64 32-bit registers, though many of them are hardware defined so that they may not all be used as general purpose registers). During testing two things were observed. First, many tile crashes were observed (due to overwhelming of fault tolerance—there is no evidence that a single upset could cause a tile crash except for the documented L1 data cache sensitivity issue). Second, several register upsets were observed. We determined there was an event type that was similar to a tile crash and could

contribute counts to the register upset detection. So although we could distinguish between register upsets and tile crashes, we still had to contend with crash-like events that give register upset counts.

If the crash events were the only events to consider, then it is reasonable to think that the limiting cross section would be defined where it would be difficult to separate real events out of a sea of tile crashes. This would likely be when one event is detected for ten tile crashes. This gives rise to the first part of the sensitivity determination.

$$\sigma_{\text{limit}} \cong \frac{1}{10} \sigma_{\text{disruptive}} \quad (4-4)$$

Equation (4-4) is essentially just a signal-to-noise problem. If you want to push the limiting cross section down, you just need to take more data. However you can't just increase the flux because disruptive events generally scale with the flux. So you will need to run slower to push down the limiting cross section. Note that here we are talking about disruptive events that make it difficult take more data. These are clearly distinguishable from the target event type, but they severely limit how much data can be taken. This is why the target event cross section can be found, within reason, at about 1/10th the cross section of the disruptive event (i.e., because it is obvious when a target event occurs, and it is reasonable to collect data across ~10 disruptive events).

Other events can also provide an SEE signature similar to the desired event, but they are actually due to the accelerated laboratory environment and should not be counted. In this case one must develop an estimate of the rate for the worst-case offender that cannot be uniquely distinguished from the target SEE. This is different from the previous paragraph in that here we cannot determine if we are observing true events of the type we are targeting. As an example, it was determined that double-bit errors in L2 cache tags lead to tile crashes (see Section 7.3.5). We also observed that error counts in test algorithms were sometimes correlated to how many times the algorithm had run without being reloaded from the L2 cache. We concluded that these events could be due to double-bit upsets in the L1 instruction cache. Since the structure types were similar for double-bit errors (DBEs) in L1 and L2 caches, we decided the L2 crash sensitivity established a lower-bound on the device cross section for L1 DBEs that can be observed. This sets a limit on the effective sensitivity for observing DBEs in the L1 cache.

For Maestro ITC testing we established the effective sensitivity as the cross section where observing one event is equally likely to as having 20 or more of the 48 test tiles crash. The formula for determining the appropriate tile crash probability (and hence cross section) for this is given in Equation (4-5) [30,31].

$$\sum_{i=0}^{19} \binom{48}{i} p^i (1-p)^{48-i} \approx 0.5 \quad (4-5)$$

We find that p satisfies Equation (4-6) when the test fluence is 0.40/ $\sigma_{\text{Tile Crash}}$. The resulting effective sensitivity is given by the sensitive cross section defined in Equation (4-6).

$$\sigma_{\text{limit}} = \frac{\sigma_{\text{Tile Crash}}}{0.4} \quad (4-6)$$

This result is subjective during the Maestro ITC testing because it may have been possible to examine the data more carefully to eliminate false counts and improve the signal-to-noise ratio (SNR). Also, the SNR could have been improved by reducing the flux. And finally, SNR does not limit detection; it only determines how hard it is to go after data. In our case we did not have time to take many more data, which would have increased the signal. The resulting effective sensitivity plot is given in Figure 4-1 [30]. Note that this is an upper bound on the detectable cross section. It is determined by a large set of tile crashes, and thus the actual position of the limit line is based on error bars that are more accurate than 10%. However, the whole process includes flux dependency and accuracy of the detection algorithm, so although the determination of the data point location is good, the definition of the SNR based on Equations (4-5) and

(4-6) is a systematic uncertainty that could result in a lower limiting cross section or improved effective sensitivity compared to Figure 4-1.

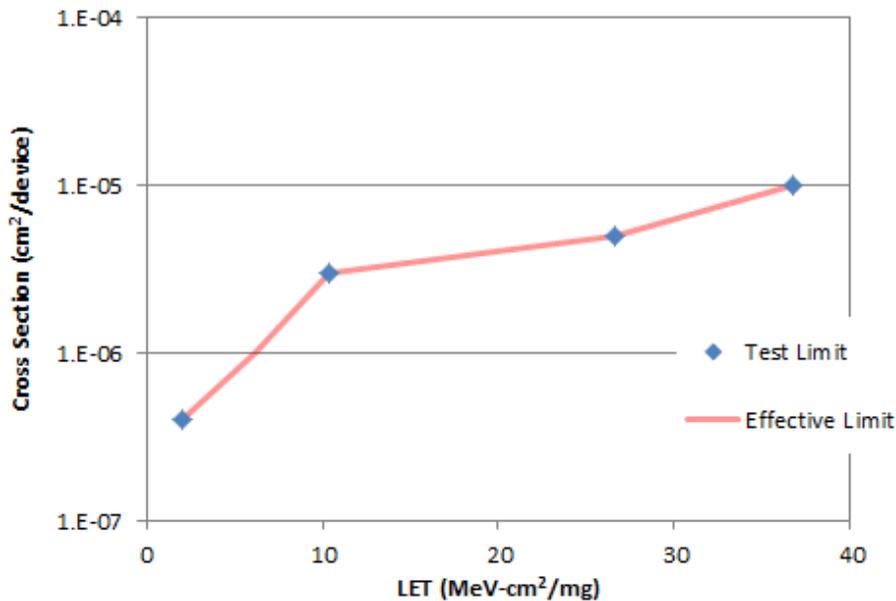


Figure 4-1. The effective sensitivity achieved during Maestro ITC testing. Event types with a cross section below the effective limit were not likely to be reliably observed [30]. Image used with author's permission.

4.3.3 Improving Effective Sensitivity

Effective sensitivity can be improved through three primary means. The first is increasing the amount of data taken to increase the signal to be detectable over the noise. The second is improving the event detection. The final method is eliminating anomalies.

Increasing data taken can defeat SNR because the signal increases linearly with data while the noise, which should be Poisson in nature, increases with the square root of the quantity of data taken. Under certain circumstances this can be a useful approach. However, it is recommended to carefully examine the noise source before arbitrarily increasing beam (which may be very expensive). In our Maestro testing about five register upsets were observed over all test runs, and all of them occurred on the second pass of the test loop. More data would allow us to determine if the register upsets were equally likely in both passes. If we tested to 10 times the fluence (at the same flux, otherwise flux-dependent problems would come in), we would expect 50 register upsets, and unlike the five observed in the Maestro case, if it is a true SEE, then we would expect (approximately) 25 ± 5 to be in each of the two loop passes. However, with only five events, we expect 2.5 ± 2.2 in each loop pass, so seeing all in one loop pass is not that unlikely. We cannot conclude whether all the upsets occurred in the 2nd pass due to the random timing or upsets or due to a systematic sensitivity. Again the reason this works is because the noise, which comes from apparent correlation to loop-pass count, increases with the square root of the fluence while the number of counts increases linearly with the fluence.

Improving event detection is probably the most cost-effective way to improve sensitivity. In this case the goal is to clearly detect an SEE and develop parameters relevant to the SEE that would be very difficult to obtain from an anomaly. In the example of the Maestro registers from above, one way to improve event detection would have been to change the program structure so that the second loop pass included a reload of the comparison code so that double-bit-errors did not contaminate the data. This would eliminate coincidental double-bit errors as a source of test anomalies in the test data.

Eliminating anomalies is always good practice in SEE testing of complex devices. However, unless the anomaly is truly surprising at the test facility, the likelihood is that it has been thought of and ruled too difficult to fix. An example of a surprise that could be fixed is the UT699 trap-in-trap behavior. During initial testing we did not expect the trap-in-trap behavior to occur at all. When we found it was occurring, the most logical thing to do was look at the trap handlers to see which operations could lead to trap-in-trap. Ultimately we determined the behavior was coming from the window overflow trap handler and eliminated it. Then we made sure that the conditions causing trap-in-trap were duplicated in the test code so that we could see the offending trap event (that is, the trap-in-trap sensitivity will affect real applications, but by removing it from our test code [to improve SEFI response] we had to intentionally add the ability to the test code in order to observe the behavior and obtain cross section data).

4.3.4 Recommendations for Effective Sensitivity

Test engineers should try to determine the limiting factors of their test setup before going to the beam. This way they can set useful targets for testing without wasting beam on measurements that cannot really be made.

In most cases the effective sensitivity is going to be the expected beam exposure at each LET. For example, at high LETs guidelines suggest that one test to $1 \times 10^7 \text{ cm}^{-2}$ or 100 events [32]. If events are expected that will result in the device crashing, then the effective sensitivity is likely reduced to about a tenth of the cross section for the device crash (testing with more than 10 crashes for one LET setting is very time consuming). If flux dependence can play a role in device crashes, then a lower flux setting may be desired, with the resulting exposure time being the limiting factor.

If an observed event can be readily isolated, consider doing that on-location to improve your data set. One way to do this is to increase the number of targets or the duty cycle for use of the target(s). Alternately you can come up with multiple algorithms that are sensitive to the desired SEE but have different additional sensitivities.

Be wary of spending too much effort creating multiple methods to detect SEEs. Multi-level triggering (where an observed event is flagged and minimal data recorded but full data is only collected if additional event behaviors are observed) can be used, as can be found in discussions of high energy physics experiments, but it should be considered outside the realm of normal SEE testing [95].

4.4 Testing Problems

The hardware and software used to sensitize the DUT are very important to understanding the results of SEE testing. This is true both for the types of SEEs the system is designed to capture, and the other events that occur and result in some signature in the dataset that may not be clearly explained.

When developing a test approach, it is expected that engineers will develop systems that can observe and count SEEs that are expected. As long as this set of event types includes the most important event types for normal operation, then general operational parameters can be established. It is common, however, for devices to be sensitive to unexpected event types. It is also common to misinterpret the most important event types. And similarly, it is common to over-characterize largely unimportant event types.

Several common types of events can occur in complex SOCs that may be overlooked or over-emphasized in developing test systems. We discuss these events in this subsection and in the next subsection discuss tools that can help to examine tester-related data corruption. If unexpected events dominate a dataset, it is likely that the results will be of minimal benefit to users. This occurs because unexpected events are very likely to be due to unexpected test program behavior.

4.4.1 Multiple Event Sensitivity

Test systems are vulnerable to multiple bit upset events where the effects of one error are dormant until a second error occurs. In real applications the time between multiple upset events is very long, and systems have time to observe and correct or recover from the system impact of most errors before a subsequent event occurs. This is true except for event types that are unknown and cannot be fixed without resetting or power cycling the device. For this reason, it is recommended that these devices be periodically reset and/or be power-cycled. The rate for performing these operations should be on the order of once per year in flight (if hidden events are likely to be an issue on a shorter time-scale, the device will be exhibiting many other effects each year and likely need to be reset or power-cycled as a result of ground operations to deal with SEEs). During ground testing this effect is reduced by ensuring the DUT is power-cycled before each beam exposure. In many cases, however, power-cycling the DUT may be cost-prohibitive in terms of test (and therefore, beam) time. The use of power cycle, or quick reset, may require discussion.

During testing it is expected that systems will occasionally exhibit upsets in upset-handling routines, and they are also likely to exhibit the dormant upset type. This is especially true when attempting to collect statistically significant numbers of events (more than 10 in a run) in a brief period (less than 10 minutes per run).

Another major source of multiple event sensitivity is due to the test algorithm. Many test algorithms attempt to collect “semi-static” data. In this case although the processor is running, the target structures are not performing any action, and the goal is to setup conditions before applying beam, and then check the status of the prepared material after the beam exposure to identify upsets. Clearly the structure of this type of test anticipates multiple upsets in the tested structures. These types of algorithms, and collected test data from these algorithms, must be tested against multiple fluxes and total fluences delivered between setup and examination of test structures to ensure there is no evidence that multiple events are altering the recorded data.

Under normal circumstances test engineers will not be able to completely handle all potentially semi-static structures within the device during testing. For this reason, it is recommended that an examination of flux and fluence sensitivity of all algorithms be made before collection of production data. That is to say, it is recommended that algorithms be tested with two fluxes and two fluences, at least 10 times different in size, to ensure no flux or fluence dependence is occurring. If fluence dependence is observed, then lower fluxes and fluences should be preferred unless the dependence can be alleviated.

4.4.2 Flawed Analysis Due to Incorrect Assumptions

One very common misunderstanding about SEE testing is testing a device for upset sensitivity of basic elements, and then applying arguments that assume error detection and correction (EDAC) schemes in order to determine upset rates in space environments. In some instances we found that these EDAC systems are not actually activated by users.

An illustrative case is the P2020 SOC that is a source for this guideline. This device has two Freescale e500 cores that have parity-protected L1 caches. These caches have been observed during testing to show no single-bit upset in the data read from them, but instead they have cache misses when SEEs occur, which causes fresh copies of corrupted cache lines to be fetched from either the L2 cache or off chip. This behavior is desired when the SOC is run in a high-reliability mode, where the L1 data cache is used in write-through mode, and all information in both the L1 data and L1 instruction caches are duplicates of information stored in a presumably more SEE-robust location (the L2 cache has error correcting code (ECC) protection). In many cases, however, the L1 cache is not operated in the write through mode, even in aerospace applications.

The P2020 illustration does not mean that a significant change in test methods is required. It only serves to highlight that if you tested assuming the device would be configured to handle upsets in the L1 cache, then the test data would be unable to reflect the L1 upset rate. Without the L1 upset rate, if assumptions about

use turn out to be incorrect for a particular application, then the test results would have to be carefully analyzed to determine benefit for the flight user.

This discussion serves to highlight that it is important to understand low-level structure sensitivity (possibly under different usage cases, such as semi-static or at full throughput), understand EDAC resources, and report SEE knowledge relevant to a project's EDAC usage. Note that without a well-documented EDAC usage, it is recommended to assume none will be used and indicate in reporting that rates will be improved by EDAC usage.

4.4.3 Flux Dependence

This information is also discussed earlier in the guideline from a different perspective (section 2.2.6). We briefly touch on the key information here and expand it somewhat.

The main thing to understand about flux dependence is that it is very difficult to show that a flux dependent effect is the sole cause of observed upsets. The reason is that upsets that can be seen as flux dependent are hypothesized to be caused by two events (or more) that result in an observable event. An example is an uncorrectable EDAC event in a SECEDED EDAC system. Here when an uncorrectable error is observed, it is supposed that this is due to two uncorrelated upsets. To prove this it is not sufficient to show that an increase or decrease in flux corresponds to a similar change in the observation rate for errors. Instead, the flux must be reduced to show that a sufficiently low upset rate can be achieved to meet program requirements with the assumption that the error being due to a true SEE.

Many algorithms have flux dependence because of an integration interval. That is, the operation is dependent on a "setup", "hold", and then "use" window. The hold period provides an interval for multiple events to occur. These windows tend to have controllable duration, unless they are hidden as discussed in Section 4.4.1. By shrinking or eliminating the window, improved understanding of flux dependence may be gained (though this may be misleading because often the hold window is what provides the duty cycle of the test).

Finally, it is important to point out that it is possible testing may not be able to strictly prove the event rate is reduced to the desired level by direct testing. It is instead possible to model the event observations as a combination of a basic SEE sensitivity and a flux-dependent portion. The analysis effort can include a Chi-Squared minimization of a linear function of flux, as in Equation 4-7.

$$\sigma(\varphi) = c_0 + c_1\varphi \quad (4-7)$$

where c_0 is the constant part of the cross section, and c_1 is the linear portion. The c_1 term is enhanced in high flux laboratory testing, while c_0 is the key component for flight systems since it is not reduced in the space environment and thus dominates the cross section σ . Here the goal would be to show that c_0 is below some target value (such as $1 \times 10^{-7}\text{cm}^2$). However, when fitting Equation (4-7) to the data, the uncertainty in c_0 will be heavily dependent on the number of observed upsets at low φ . But if many events are seen, then a good measurement of c_0 is immediately available without fitting, and if very few are seen, then c_0 cannot be constrained well. For SEE testing, space rates are often not known within a factor of 2 or more before exposure to the space environment. The difficulties involved in establishing a good set of parameters are such that only a small fraction of required events can be reduced by doing a model fit as discussed. And since the result without modeling would only be slightly altered by this limited reduction, it is not recommended to use this modeling approach.

Conversely, if at multiple flux levels the cross section is seen to be approximately constant, then the model approach argues for a very small value of c_1 and a non-zero value of c_0 . This value of c_0 is then a basic SEE sensitivity, and will have been established to a reasonable level to rule out flux dependence.

4.4.4 Anomalies

A natural extension of examining flux dependence for identification of a combination of basic and flux dependent SEEs is the observation of anomalies. The underlying SEE sensitivity examined in the previous subsection is something that is worried about if it is not expected (i.e., an anomaly). This subsection deals with how to examine anomalies during testing. We cover the most common sources of non-DUT anomalies, which are stray beam and effects in support circuits. And we provide a specific example in the “register partial reset” event observed in the Cobham Gaisler UT699.

Anomalies should always be attributed to the DUT unless there is a way to show the anomaly is due to some target other than the DUT. It is often very easy to isolate events to the DUT by altering test arrangements and showing the DUT continues to exhibit the anomaly. However, sometimes the number of events is very low. When this happens it is difficult to correlate changes in observed anomaly rates to changes in setup. In practice it is often valuable to isolate events to a particular structure and then focus testing on that structure until the observed SEE is better characterized. This bypasses other elements of anomaly investigation because presumably the observation algorithm can be made arbitrarily good (as needed) to provide event isolation.

One common source of anomalies is stray particles striking devices believed to be protected by shielding or beam collimation. The most common version of this is stray neutrons in proton facilities. One way to examine this case is to shield the DUT from the beam and expose the test chamber or room to the same environment as during a normal beam run. If events are observed at the same rate as when the DUT was not shielded, it is probably a sign of background particle anomalies. Unfortunately, because shielding and collimators can actually produce significant quantities of background neutrons, this is not a definitive test. If shielding the DUT does not significantly impact the event rate, it may be necessary to go further and try to ensure that the DUT is out of the path of neutrons or other particles coming off of the collimator or shielding material during exposure. The DUT can be put out of the beam path, while attempting to put the rest of the circuit in the same position as before the DUT was moved out.

Another source of anomalies is errors in detection systems or support circuits. These can occur because of stray beam, as in the previous paragraph. But they can also be due to unexpected conditions that occur during exposure of the DUT. In most cases, the same approach of shielding the DUT and seeing if the events still occur is sufficient. However, sometimes the DUT exposure leads to instabilities that arise in other circuits, so exposure of the DUT is required. In this case the test engineer will have to determine the most appropriate course to follow. Selecting the right course may require significant detailed knowledge about the structure of the device and the software running on it.

One example of an anomaly during testing occurred when testing the UT699. During testing it was observed that the C-based code would encounter “trap in trap” more often than expected, and it would also exhibit EDAC uncorrectable traps [34]. The EDAC uncorrectable trap was indicative of an event caused by multiple upsets in a register, which was not expected to be due to a single ion, and therefore initially thought to be caused by uncorrelated SEEs. The cross section for these events was observed to be around $3 \times 10^{-6} \text{cm}^2$, which means that in any test run we would expect three or less events (when testing to a fluence of about $1 \times 10^7 / \text{cm}^2$). This was further confounded by the fact that the cells in the registers had a cross section that could be as high as $1 \times 10^{-8} \text{cm}^2$, and since there were thousands of these cells in the device (including the caches), we could not immediately determine if the events were due to integrated fluence (equivalently flux), or if the event type was due to an underlying device sensitivity. A lot of effort was put in to trying to test with lower flux; however, test runs were already on the order of thirty minutes during early testing. Increasing the flux would definitely lead to more anomalies, but in this case may have determined the answer quickly. The approach taken, however, was to identify the event. Analysis of the signatures collected during isolation of events clearly indicated an upset mode where the DUT has 16-bit portions of its registers set to 0, which we defined as register partial reset. Because of the nature of the error, a significant fraction of the time that this event occurred the only observable effect in the test system

was to observe that 16 bits of a test register were changed to 0. Once this was done, it was easy to increase the beam flux and exposure durations, knowing that any increase in anomalies would be anomalies that are fundamentally different from the identification algorithm for events. In this way we were able to show that the event did not scale with increased flux, and therefore was likely a true SEE.

4.5 Debugging Tools

Debugging tools are very important for testing of advanced devices. Debugging tools run from special-built manufacturer tools all the way through third-party tools and user-written algorithms to detect errors in a running program. Specialty tools are designed to enable debugging of hardware problems on boards and software problems in user applications. Custom and third-party tools can provide an examination of software such as profiling code performance and detecting errors in a running program. Additionally all observed information coming from a running program can be used for debugging, especially with a detailed understanding of how user software and manufacturer hardware are supposed to work, and how their operation may be impacted by SEE. The main thing to keep in mind regarding debugging tools is that they are not designed to collect or explore SEEs, and as such the test engineer will have to understand the tool well enough to interpret what it is saying relative to SEE behavior.

In this subsection we discuss how to use various existing tools, and how to develop test code in such a way that the test code and output files become additional debugging tools. The goal is to be comprehensive regarding the types of tools, but not regarding specific tools or specific hardware. We also aim to highlight the areas where test engineers may feel compelled to collect data of limited value so those areas can be avoided.

4.5.1 How Tools Can Be Used

There are three basic ways debugging tools can be used. First, they can be used to fix problems in test software. Second they can be used to examine the state of a device that has encountered an SEE. And finally, they can be used as the primary test sensitizing and data collecting system. Debugging tools operate under conditions different than normal hardware operation. They are designed primarily to troubleshoot software, with limited hardware support to troubleshoot circuit problems. As a result, we do not recommend using debugging tools as the primary radiation sensitization.

The first recommended use of debugging tools is to examine anomalies. In this regard an anomaly may or may not be an SEE. The problem is that the event has exceeded the test software's ability to handle the event. By using the debugging tools, details about the execution state of the software and the details of the current hardware state of the processor are available. At the very least, noting the hardware and software conditions of an anomaly can help identify major flaws. These tools then can help identify an SEE type the system was not originally designed to isolate, as well as help identify problems in software that could not be tested for precise beam-induced upsets without real beam-induced SEEs.

The issues identified in the last paragraph lead to the use of debugging tools to identify modifications and possible debugging needs. The information can be used to eliminate problems in test systems on-site. Embracing this ideology at the beam facility can lead to problems with completing test plans. But not addressing these problems can lead to flawed data sets.

An example of this occurred once when a test group observed crashes related to bit errors in an L1 cache. They were not able to identify why the software was vulnerable to this problem, so they collected data on this event. This was very intrusive as the L1 cache gets bit errors readily, and it overshadowed any other error type that could have been observed. With on-site debugging tools they could have easily had the test system break when the L1 cache parity handler triggered in order to figure out why the test system was incorrectly crashing on this event.

4.5.2 Manufacturer Tools

Manufacturer tools are available for virtually every platform. In order to connect to the study devices, we discuss manufacturer tools relative to the study devices. The tools include the Gaisler Research Monitor (GRMON/GRMON2) for the UT699 [96], CodeWarrior for the Freescale devices [97], and the Tiler Board Test Kit (BTK) for the Maestro ITC. These tools typically include the following capabilities (these are somewhat limited with the BTK):

1. upload test code
2. execute the code in a single-step or breakpoint configuration
3. interrogate registers, caches, and memory
4. break into a running system

Between points 1 and 4 above, these tools can generally be used to force the SOC to execute custom software from a desired platform once the platform is booted. However, it is recommended to have the test code take complete control of the SOC by shutting down any interrupt or multitasking operations that might take control from the running test code.

An example of reading registers in a LEON device using GRMON2 is shown in Figure 4-2. This shows a snapshot of the currently relevant window registers and processor core status and configuration. This printout can be used to quickly identify if the currently executing line of code is in the right context with the right information in its registers. Similar tools exist to allow easy viewing of the processor status for all devices.

```
grmon2> reg
      INS          LOCALS        OUTS          GLOBALS
0:  00000008      0000000C      00000000      00000000
1:  80000070      00000020      00000000      00000001
2:  00000000      00000000      00000000      00000002
3:  00000000      00000000      00000000      00300003
4:  00000000      00000000      00000000      00040004
5:  00000000      00000000      00000000      00005005
6:  407FFFF0      00000000      407FFFF0      00000606
7:  00000000      00000000      00000000      00000077

psr: F34010E0   wim: 00000002   tbr: 40000060   y: 00000000
```

Figure 4-2. Example console display of the contents of register window 2 on a LEON processor [96]. The upper set of registers are the in, out, and local register sets (8 registers each) for register window 2 (specified by window invalid mask (wim) in the lower set), and the 8 global registers. The processor state register (PSR), trap base register (TBR), and condition register (Y) are also reported. This provides an immediate snapshot of the status of the running processor core.

In order to be ready to use these tools at an SEE test it is recommended to explore methods where the debugging tools can break into running test code.

4.5.3 In-Code Debugging

In-code debugging refers to a type of debugging where the behaviors of the hardware and software are examined through utilizing code explicitly designed to interrogate hardware and software operation. This type of debugging is very important and has a basis in understanding the hardware-software interface.

During testing the data stream returned is determined by what is written in the test code. It is important to design the test code to enable as much in-place debugging as possible while also providing adequate test

data to support additional analysis after the test. The following section discusses code constructs and actions that can enable on-site examination of anomalies (and crashes).

4.5.4 Adequate Test Output Files

Test output files can be a treasure-trove of information. But often when a lot of data is collected, the output files become a wilderness of information looking for questions to be asked. Due to constraints it will be difficult to analyze large amounts of data for all but the most important information. Thus, it is important to make sure that test output files provide good information, but don't just generate mountains of unusable data.

Recommendations for the information to be recorded and transferred during testing are given below.

1. Periodic output of test state (the state often includes some indication of recent SEEs)
2. Full test log with basic operations and key events: test configuration changes, interrupt events, etc.
3. Error log or immediate error output
4. SEE records stored during test and reported after the test completes

The algorithm recommendations in Section 3 provide information for how to implement the output of data during the actual test operation.

The immediate benefit of having a stream of data is to be able to quickly identify common error reports leading up to crashes. So obviously, the last errors observed provide a very good bread-crumbs trail for the error. If data are stored in non-volatile memory, that data should be dumped following any crash. The generated data from the recommended list can provide data in the event of code anomalies or crashes (if the data can be recovered after crash).

It is possible that error messages may result in runaway conditions. If this occurs, it is recommended to build in a number limitation so that only the first few messages are reported (unless the runaway condition does not overwrite the early messages). The error report right before the runaway condition likely leads to the condition that causes the runaway. Finally, if possible, attempt to use hardware debugging tools to break into the system at the point of the runaway condition and see what causes it. Bear in mind the possibilities are pretty large, so the current configuration of the MMU including virtual memory mapping and memory coherence issues related to cache contents (among others) could all give rise to runaway conditions.

4.6 Recommendations

The list of recommendations that come out of this section follow.

1. Have a method for establishing the beam and data quality on-site in order to avoid problems during analysis or reporting.
2. Know how to establish the limiting sensitivity threshold for the device.
3. Have a tiered plan for test goals, which can be flexible to debugging time
4. Determine if flux or event identification improvements can improve testing sensitivity.
5. Cycle the power off and on for every run, unless it is cost prohibitive to do so. Indicate what is done before the start of each run in the test log (power cycle, reset, etc.).
6. Turn off power as often as possible. This differs from 5 in that it is recommended to turn off DUT power as soon as possible after an exposure in order to allow the DUT to cool
7. Determine if multiple events are altering your data by varying flux and fluence on semi-static tests, and be aware that the test code always has some semi-static portions.
8. Collect data on basic structures (possibly while operating at different duty cycles).
9. Do not use debugging tools as a primary means of sensitizing a DUT.
10. Do use debugging tools to examine anomalies and identify software bugs.

11. Understand EDAC systems.
12. Report worst-case numbers with caveats that EDAC improves things – force the project to document their EDAC usage before assuming it.
13. Bring a software person to the test.
14. Record anomalies and details about the anomalies.
15. Analyze anomalies with both hardware and software debugging tools.
16. If anomalies do not impact key data or cast doubt on understanding of device operation, do not be led astray by detailed examination.

5.0 ANALYSIS AND REPORTING

This section provides a detailed review of reporting of testing, test data, and data analysis work. It is intended to be used primarily as a checklist that includes items that should always be included and items that the test engineer should evaluate if they are relevant for a given test.

The focus here is on the following. First, what information should be taken before testing, as applies to DUT and test equipment preparation? Second, we focus on information to record during beam testing. Data analysis details are covered third. We then discuss test report specifics. And this section concludes with a review of a few key recommendations.

5.1 Pre Irradiation Measurements and Observation

While preparing devices for an SEE test, some key observations should be made that will be noted for the test records. They include the following.

1. The full device markings (it is useful to take photos that clearly show the markings)
2. All modifications to the DUT should be recorded
3. Any measurements deemed important for the test should be recorded for each DUT while running each of the test algorithms
4. Nominal operating behavior

For determination of nominal operating behavior, be sure to record the test board used and how the DUT is installed (e.g., is a heat sink used, or are you using a nitrogen line or fan to supply cool air).

The behavior of the DUT in the test algorithm to be used during exposure should also be recorded. Each test algorithm should be examined for general operation noting the following information.

1. DUT resources used (caches, memory controllers, etc.)
2. Operations performed by the algorithm (be as specific as reasonable, but note that instruction-level sensitivity is generally not measurable)
3. Operational speed (operations per unit time)
4. Data transfer rate/throughput (for I/O operations)
5. Temperature of the DUT (a qualitative assessment as many algorithms will not reach thermal equilibrium during testing or pre-testing)
6. Current draw (if possible, this measurement can be used to observe DUT drift during testing and provide insight into differences in test algorithms)

The number of instructions executed by a test algorithm, or the number of bytes transferred through an I/O interface can be surprisingly different than expected for various algorithms. For example, an I/O interface running at 3 GBps is running at a throughput that is significantly lower than the processor bandwidth (which is more than 10× higher), so any testing of the processor during I/O operations may be more than 90% wait operations. Similarly, I/O operations may be performed through direct memory access (DMA) in such a way that the processor does very little and memory buffers are highly utilized. While it is not practical to analyze the SOC in extreme detail, it is very helpful to record the construction of the software loop, the number of instructions actually executed for a loop iteration, the duration of a loop iteration, and how much I/O (bytes in and out) is performed. With this information, the algorithm and data can be analyzed to estimate worst-case behavior.

In addition to the test system and DUT, the test engineer should also note any details of the test that can impact collected data. If the beam structure at the test facility is not uniform in time, this should be noted. General beam uniformity and the impact of flux on beam uniformity and counting should be considered before arrival at the beam facility (if possible).

5.2 Test Log Requirements

During testing the test log should include several different types of information. Some of the material here is fairly common, but some is particular to SOCs. Generally the DUT is prepared for exposure by starting from a fixed state (such as DUT off), powering up and running all equipment for the test algorithm, exposing the DUT to the beam source, then turning off the beam source and recording all relevant information before returning the test system to the fixed state. This entire procedure is a run.

The test log is recorded for each run. It is recommended that any terminal or other DUT-related capture files be referenced by a global run number used to identify each set of operations described in the last paragraph. Each run should have the following information recorded.

1. DUT Information
 - a. DUT identification
 - b. Approximate temperature
 - c. Nominal current draw at a fixed point in the operation
2. Beam Parameters
 - a. Ion
 - b. Ion Energy
 - c. LET (heavy ions), or Energy (protons) at sensitive region (i.e., surface of DUT to 20–30 μm into the die)
 - d. Flux (note overall beam structure beforehand, and record any changes to beam structure)
 - e. Fluence
 - f. Live time of the beam (can be calculated from Fluence divided by Flux)
 - g. What degraders are used
 - h. Angle of incidence of the beam on the DUT (including axis of rotation if DUT is rotated relative to beam)
3. Test Algorithm Details
 - a. Unique name for the test algorithm (new versions should have different names)
 - b. Any settings used to tailor the algorithm (such as “all 0s”)
 - c. Approximate live time of the test algorithm (can usually be recovered from the number of loop iterations on a repetitive algorithm, or the fixed operating time of a single execution algorithm)
4. Static Data Collected
 - a. Number of sensitized bits (this may be recorded before testing as a quality of the test algorithm)
 - b. Number of observed upsets
 - c. Specific error patterns or information to enable multiple bit upset (MBU) analysis should be collected, but may not fit into the test log.
5. Test Code Results (especially for dynamic tests)
 - a. Number of main loop iterations (this is the widest loop structure in the test algorithm)
 - b. Count of errors for each detectable error type
6. End of run handling
 - a. The final state of the test algorithm after exposure should be recorded.
 - b. Include notes to provide details on any unexpected phenomena.
 - c. Include notes to provide details on recovery actions taken.
 - d. Often DUT reset is used to determine if the DUT still functions after a crash or hang. If this is part of the test protocol, perform reset before power cycling of the DUT.

- e. It is recommended to turn off the DUT after all information is collected and before the next run. This ensures a reliable operating state and provides a cool-down period in the test operations.
- f. Finalize and close all log files (power supplies, terminal logs, IO captures, etc.).
- g. Verify the DUT temperature is nominal before beginning the next run.

In general, at the end of a run the state of a complex SOC is not well known. It would require an enormous amount of effort to verify the entire state of the DUT, and it is not clear that manufacturer tools would be reliable because they are not generally built to handle conditions that cannot occur during normal operation (e.g., a bus with a deadlock-robust protocol deadlocking may be outside of what the manufacturer tools can test). At the end of a run, if all seems good, then the state of the test system should be recorded as “good”—this should be clearly written in the log. If anything appears to have crashed it should be noted in the test log (e.g., “core 1 crashed”, “cores x, y, z crashed”).

Changes to test algorithms during testing may be required. A modified algorithm should be given a new name so that data is not incorrectly associated to the wrong test algorithm.

5.3 Data Analysis

Data analysis for complex SOCs generally follows that performed for SEE testing of common components and standard guidelines are good resources [31,32]. There are a few topics that require elaboration regarding SOCs. These come about because of FT systems and multicore. The former necessarily requires special data analysis to verify FT systems and/or to extract data that is only observed because FT systems could not correct errors. The latter is required because multicore devices add a level of complexity to the dataset.

5.3.1 Effective Sensitivity

The complexity in an SOC suggests that testing intended to highlight upsets in a target subsystem may be overshadowed by system level upsets, or by upsets in some other portion of the SOC that is inherently weaker than a target subsystem. The weakest part of the test system of the SOC, the test hardware, and the test software, becomes the limiting factor in the test effort and is responsible for the effective sensitivity of the test.

For each algorithm the test engineer must determine the SEE type that is causing the majority of the observed events. For example if double-bit errors in the L1 instruction cache, leading to failure of parity to detect incorrect instructions, cause execution of modified instructions, this can be the cause of observed SEEs.

If test anomalies are occurring, they can usually be taken to be the floor of the effective sensitivity. Test anomalies can include things such as: unexpected characters in the report sent to the test log, test code crashes, changes in test code operation, processor core hangs or crashes, etc. If anomalies are being observed, it is best to use their cross section as the effective sensitivity.

5.3.2 Static Counting

For many SEE types to be analyzed, the number of sensitive bits needs to be determined. For cross section calculation, the number of detected events must be provided by the testing. For statistical reasons it is best to test to the point where 100 or more target SEEs are observed (this provides data points with 10% or better uncertainty).

For static testing the number of sensitive bits is the number of bits loaded and then tested for upsets, e.g., registers and caches.

If a multicore SOC is being tested, the number of bits tested in each core should be used. Core-to-core counts (cross section) should be compared.

5.3.3 Functional Codes

Each test algorithm should be analyzed for the information recommended in the pre-irradiation measurements and observation Section, 5.1. For each algorithm, the general function the test is designed to accomplish should be clear, and the methods by which it detects upsets should be documented. For dynamic tests the number of sensitive bits may be unknown and cross sections should be provided per device (or per core).

5.3.4 Analysis of I/O Test Data

I/O test algorithms follow the same recommendations as for functional codes (Section 5.3.3) except that the hardware involved may have structural elements that obscure results. For I/O tests there is no clear way to normalize observed upsets against the number of tested bits; however, it is recommended to provide cross sections per bit transmitted (thus, you must record the number of bits transmitted during beam exposure in the test log). For I/O tests it may also be useful to report the device cross section. For example if 3 GBps is transmitted for 10s during beam exposure at $1 \times 10^3/\text{cm}^2\text{s}$ and 2 bit errors are observed, the device cross section is $2 \times 10^{-4}\text{cm}^2$, while the per bit cross section is $7 \times 10^{-15}\text{cm}^2/\text{bit}$. It could be that each bit has the per bit cross section for an upset, or it could be that no matter what the transmission speed, bits will be upset with the device cross section (such as if the upsets come from bit flips in IO buffers that are essentially full all the time).

5.3.5 Analysis of Upsets in EDAC Protected Systems

Upsets in EDAC systems, following the algorithm recommendations in Section 3.6.2.2, should be due to double-bit errors in an EDAC word. In general it can be somewhat involved to calculate the sensitivity of the bits in the protected system. In practice, the following information must be known for a SECDDED system:

1. The detected number of SBUs in an EDAC word – S (note to get an S, you have to have a DBU with one of the associated bit upsets in a [hidden] check bit – it is still a DBU)
2. The detected number of DBUs in an EDAC word – D (observed as two SBUs in the EDAC word)
3. The number of EDAC words (across all cores) – W (keep in mind EDAC schemes can span from 13-bit EDAC words with 8 data bits up to 72 bit EDAC words with 64 data bits and beyond)
4. The amount of beam between the preparation of the EDAC protected system and the detection of events (note that this should be constant for all detections) – B
5. The number of bits in an EDAC word – N
6. The number of detection loops during the test - L

The total number of “EDAC Events”, E is simply S+D. And if the probability for a bit to be upset in a given detection loop is $\sigma \times (B/L)$, then E is given by Equation (5-1) [this is the low upset limit in Figure 2-6].

$$E = WL \binom{N}{2} \left(\sigma \frac{B}{L} \right)^2 \quad (5-1)$$

which can be used to calculate σ as in (10), for SECDDED systems only.

$$\sigma = \frac{1}{B} \sqrt{\frac{EL}{W \binom{N}{2}}} \quad (5-2)$$

5.4 Test Report Requirements

When reporting the results of SOC testing the following list of items should be included in any basic test report.

5.4.1 Description of Test Approach

The hardware and/or software approach used to sensitize the DUT should be presented. The physical hardware setup should always be included. If hardware debugging or test circuits are used they should be described (for example, if JTAG is used to test static elements). Any software used to perform testing should be described. The metrics relevant to establish the behavior of the test algorithm are given in Section 5.1.

Any changes/alterations to DUTs or test boards should be reported.

5.4.2 Reporting of Results

All SEE types analyzed should have the following information provided. The SEE type should be described. The test algorithm(s) used to extract the SEE should be noted. The relevant data analysis steps taken, per Section 5.3 should be documented. The resulting cross section versus energy or LET should be provided. And any anomalous events that appear to be directly related to the given SEE type should be provided.

5.5 Recommendations

1. Analyze all algorithms and test operations before going to the test facility.
2. Record all modifications to DUTs or test boards.
3. Determine the effective sensitivity achieved by each test algorithm at each energy or LET.
4. Document how many static elements are tested, including bit counts.
5. Provide details regarding how dynamic algorithms sensitize components via instruction types, number of instructions per main test loop, number of main test loops per run, number of performed I/O operations and transmitted bits.
6. Normalize I/O SEE cross sections to bits transferred and to device (or core).

6.0 SAMPLE RESULTS

The recommendations and background information presented were used to perform testing of modern commercial and aerospace SOCs. The goal of this testing was to explore the environment created based on the recommendations throughout the guideline and the seven-phase approach (described in Section 1.5) used to define development of SOC radiation characterization.

6.1 Examples Layout

We will examine three cases in this section. First we will examine testing of an RHBD and FT single-core device in the Cobham Gaisler UT699. Then we will examine an RHBD multi-core device (with some FT), the Opera program's Maestro ITC, built by Boeing. Finally, we will examine recent testing of Freescale P2020 and P5020 devices as commercial dual-core devices with some FT.

For each example, we will highlight which parts of the seven-phase SOC characterization approach is examined. We then pull out key findings and examine how these were identified and how they contributed to this guideline.

There are some areas of the seven-phase plan that are not well-covered by examples. They primarily involve testing of peripheral components built into the SOCs. Because of the structure of the SOCs of interest here, our test approach (described in Section 3) results in indirect testing of a couple of the IO interfaces (in particular, the main memory and low-speed serial interfaces) as well as the on-chip buses that provide access to these resources.

6.2 Example from UT699 Testing

The UT699 from Cobham Gaisler is an SOC built on the LEON 3FT Sparc V8 processor. The processor is an element of the Cobham Gaisler IP Library [48,49]. Test efforts to characterize the UT699 for SEE will apply to similar devices built on the same IP library on the same ASIC technology. Collaborative work with Cobham Gaisler was instrumental in both performing successful testing, and in exploring the register partial reset event that was observed.

The UT699 contains the LEON 3FT along with Advanced Microcontroller Bus Architecture (AMBA) High-performance Bus (AHB) and AMBA Advanced Peripheral Bus (APB), which connects the processor to its memory controller, and a host of peripheral devices. The block layout of the UT699 is shown in Figure 6-1. As can be seen, this is a good SOC for providing many common resources for space borne computing.

Because of the nature of the test approach recommended in this guideline, the first stage to testing is to develop microprocessor test code. In order for this code to work, it must be transferred to the processor, and we prefer to have normal running code, so this transfer is done through the memory controller. The memory controller is active during the test because the test code is designed to at least periodically reload the instruction cache. I/O from the processor is accomplished by communication across the UART port. Thus the APB bridge, and UART peripherals are tested during a basic test.

Once basic testing of the processor was achieved, the next step was to identify additional I/O ports for further examination. The SpaceWire ports provided a useful target for testing of I/O ports.

6.2.1 Seven Work Points for UT699

We will now examine how the UT699 example applies to the seven guideline thrusts. We only discuss the first six of the work points, because the seventh is sample testing and is what this section targets.

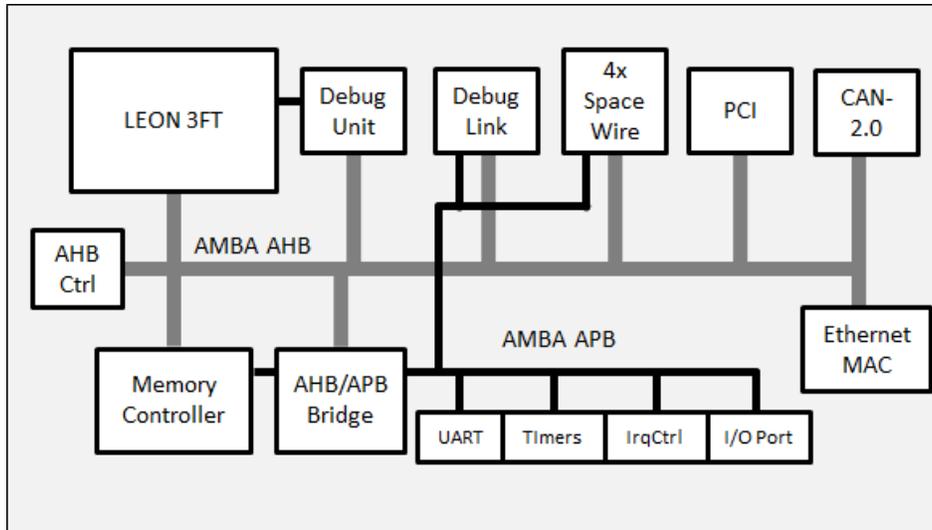


Figure 6-1. The block diagram of the UT699 (IrqCtrl = interrupt controller). (This is not the physical layout.) Image used with permission.

6.2.1.1 Collaboration

UT699 work was instrumental in examining the benefits that can be gained from manufacturer collaboration. Rather than being stuck on unexplained upsets we were able to gain insight into the design of the elements showing upsets. That information was instrumental in being able to determine the source of the upsets.

The SEE type observed was unexpected, but by collaborative testing it was possible to show that in almost any space application the event rate from these events would be lower than the predicted values due to known SEE sensitivity of other structures in the UT699 [52].

The key findings here relate to sensitivity of results and of having a good working relationship with the whole company. Cobham Gaisler was very accommodating, but it was also clear that non-positive news introduced an edge to the communications. It is very important to avoid claiming results are “unexpected” or “anomalous” without also making sure the relative impact of the result is properly communicated (this will also come up with Maestro). SOCs are very complex devices, and their use in a real system depends on how they perform the role of CPU and peripheral I/O provider. This can only be examined through running these devices. And as discussed throughout this guideline, results are highly dependent on the quality of the test code running on the CPU. So it is prudent to present any unexpected results as neutrally as possible until the underlying phenomena are understood and relevant application rates can be discussed.

6.2.1.2 Peripheral Approach

The UT699 SpaceWire ports were tested for SEE. Given the UT699 structure, these ports are expected to be utilized by many aerospace users. The UT699 has four SpaceWire ports. This proved to be useful because the ports could be programmed to transfer data to each other over a “null” SpaceWire cable.

The primary finding regarding peripheral testing from this work was that the definition of an SEE in an I/O system is not trivial. This comes down to duty cycles of the I/O system and their buffer memories. The SpaceWire results are discussed below.

6.2.1.3 Fault Tolerance

The UT699 employs the LEON 3FT core, which has parity protection on its caches and SECDED EDAC on its register file. Thus, this device was a good case for examining test methods on FT devices.

Key findings are the following. First the Sparc window architecture and supporting trap system for register window swapping is inherently vulnerable to uncorrectable SEU in the register file. Second, unexpected upsets in FT devices can result in situations where reduced flux is required to disentangle FT events from underlying true SEE sensitivity, leading to unreasonably long test exposures. However, going to higher flux and observing the SEE sensitivity to be independent of flux can establish that overwhelmed FT is not the cause of anomalies. In order to reliably increase the flux, however, the event must be identifiable without causing system hangs or crashes, which disrupt data flow.

6.2.1.4 RHBD

As discussed in Section 2, RHBD devices are more complex to test for SEE than commercial devices because various structures will have different SEE thresholds. The UT699 is a good example of this situation and it has two known key regions in its LET response [50]. First, it has SRAM cells with a relatively low LET threshold of about 8 MeV-cm²/mg. Second, it has flip flops with an LET threshold of 54 MeV-cm²/mg.

Key findings for this topic are as expected. Below the threshold for the FF SEUs, the SRAM cells lead the SEE response (mixed with FT effects). But above the threshold for FF SEUs, the behavior of the device quickly becomes too complex to accurately test. This is partially because the anomaly event type discussed later saturates at relatively low LET, so that when the FF SEUs turn on they quickly dominate the response.

It is also worth noting that RHBD devices are often based on spatial redundancy or feedback mechanisms that dramatically reduce SEE sensitivity, such as in the DICE cell. However, the geometrical structure of the cells contributes to a complex angular sensitivity that is very difficult to test for [70].

6.2.1.5 Multicore

The UT699 is not a multicore device.

6.2.1.6 General Test Methods

The UT699 helped highlight some general test method items that made it into the larger guideline. This device has a debug support unit (DSU) which was instrumental in identifying crash behavior. During testing we also encountered problems with understanding the behavior of the processor when executing library code (code not explicitly written by the test code programmer).

The key findings here are the following. Testing with standard library and build tools should be avoided when the goal is to understand all device SEEs. This includes hardware-specific standard algorithms, such as the window overflow and underflow handlers. Manufacturer debugging tools were also found to be very helpful for identifying attributes of crashes. In the case of the UT699 debugging tools, one useful attribute was that GRMON automatically engages when the processor enters error mode (which can be entered through trap-in-trap events).

The UT699 is also a good example of the test planning issue regarding low cross section events with relatively high LET threshold. In the event that devices are known to have an LET threshold above 8 MeV-cm²/mg and the device technology is known to not contain tungsten plugs, testing with protons is of limited benefit. In this case there are tungsten plugs, but the device was expected to be robust to SEE, so it made more sense to start with heavy ion exposure.

6.2.2 Test Setup

The test setup for the UT699 is instructive in showing how a manufacturer evaluation board can be used for testing. The UT699 was mounted to the board for our testing, and the lid was removed. Although the board is a compact peripheral component interconnect (cPCI), it can be powered and used in a stand-alone

configuration. The hardware setup is shown in Figure 6-2. Because the device does not transfer heat directly to its cover, the delidding process does not require careful attention to thermal control.

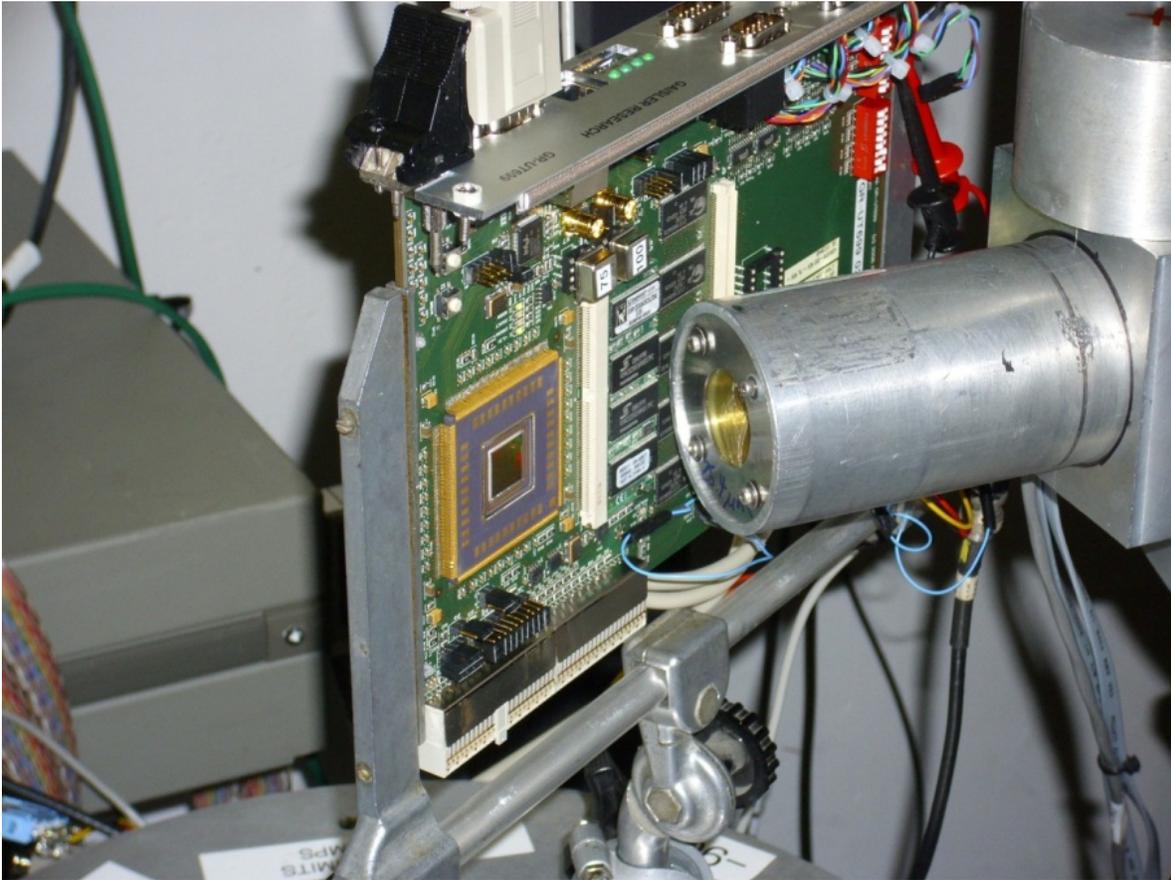


Figure 6-2. The UT699 ready for testing. It is mounted on the GR-CPCI-UT699 evaluation board and the DUT is exposed by removing the lid. Photo used with permission.

The position in Figure 6-2 reflects the test plan for the UT699 in that the device was tested at multiple angles including 0 and 60 degrees. Because of the relatively high LET threshold for the most sensitive elements (threshold of $\sim 9 \text{ MeV}\cdot\text{cm}^2/\text{mg}$), we decided not to test with protons. Testing was performed with ions from TAMU providing LETs between 9 and 60 $\text{MeV}\cdot\text{cm}^2/\text{mg}$. Although the heavy ion results indicated some proton sensitivity might be possible, we chose not to test with protons because the heavy ion data suggested there would be very few events of interest during proton testing. From [72], we know that the number of particles with LET above 8 $\text{MeV}\cdot\text{cm}^2/\text{mg}$ produced in a $1 \times 10^{10} \text{ cm}^2$ exposure to 200 MeV protons is about 3×10^3 . Since we originally thought the device cross section for these particles would be below 1×10^{-4} for the entire device, this means observing one event for every $1 \times 10^{11} \text{ cm}^2$ of fluence. It is possible that the UT699's tungsten plugs [30] could generate much higher LETs, but the number of such particles would be small compared to the total above LET 8 $\text{MeV}\cdot\text{cm}^2/\text{mg}$. Data were collected with protons showing that no observable upsets were seen in the UT699 with protons [73].

6.2.3 Response of Caches and Registers

The data and instruction caches share the same SRAM cell used in the register file of the LEON 3FT in the UT699. Combining these cells provides a larger target for measuring the per bit SEU sensitivity. The result of testing these bits is presented in Figure 6-3.

The SRAM cells in the UT699 are known to be the structures most sensitive to SEU. Because of this, these cells are protected by either parity or EDAC. The sensitivity found in the SRAM cells was analyzed to determine the risk of multiple-bit upsets generating uncorrectable errors. It was determined that the rate for this is below that of the FF elements and is less than 1 in 1×10^5 years (the exact value depends on the orbit, and calculations are found in [52]).

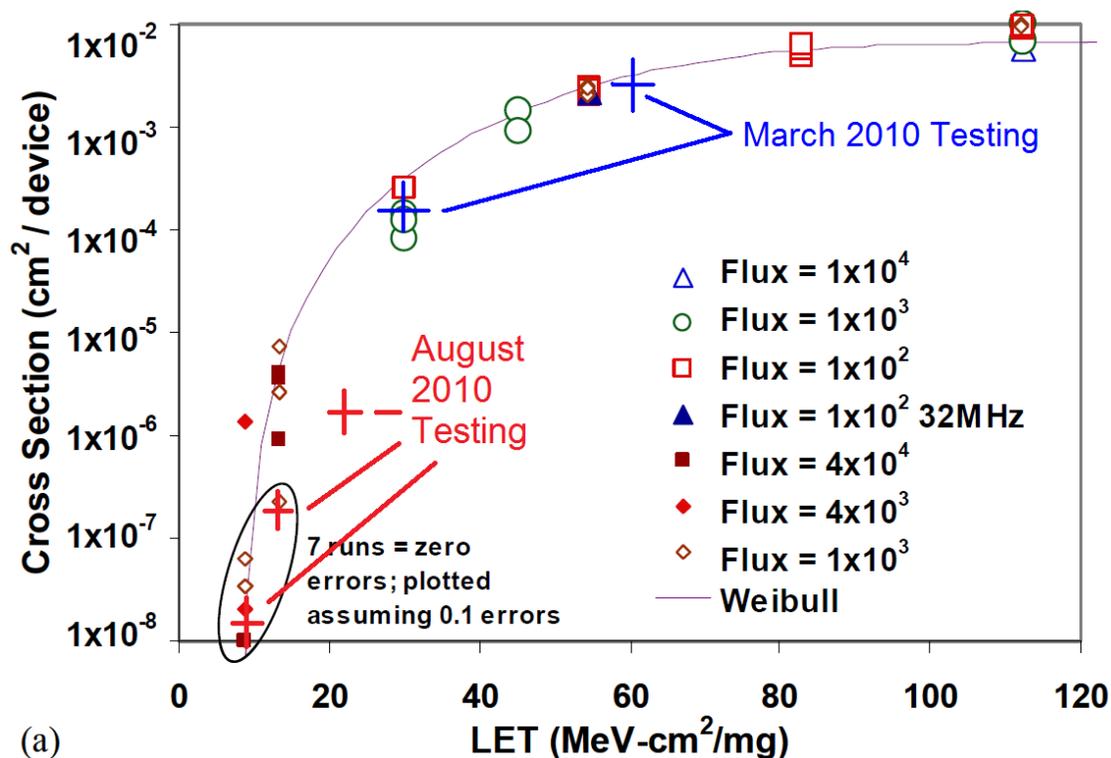


Figure 6-3. The cross section for SRAM cells was measured and compared to earlier data presented in [51]. Upsets in these bits will be corrected before they can impact the operation of the processor. Figure used with author's permission.

During several testing campaigns we used several different test algorithms to determine the sensitivity of the static random-access memory (SRAM) cells. All the algorithms were based on monitoring the built-in error correction counting registers. The L1 caches have counters for parity events, while the register file has counters for EDAC corrections. When an EDAC uncorrectable error occurs, the UT699 encounters trap 20_{hex} (hexadecimal, or 32 in decimal). Some of our test code counted these directly, others noted the trap but did not count it, and still other test code crashed. However, as discussed in the subsections that follow, there is no evidence that EDAC uncorrectable errors came from actual SEEs on multiple register bits during testing. The only significant finding was that the UT699 appears to have different cache bit sensitivity depending on whether or not the cache is active⁶. For this reason, we recommend testing static elements both in static and dynamic modes (though you must attempt to make the duty cycle for observing bit upsets be as high as possible).

This testing is a good example of a common pitfall in testing. This information can be well-tested and cross sections locked down. The number of events collected can be very large (more than 1000 at a single LET, with total exposure less than 30 minutes). However, in an FT device, the traditional weakest elements will be the most tolerant to SEE because they are the ones that are targeted for FT. Thus, although it is good to

⁶ Activity-dependent SEE sensitivity is expected in newer devices, especially if lower activity results in active voltage reduction. This is a topic for an updated version of this guideline and is out of scope at this time.

verify the FT works correctly, actually characterizing the SEE response of the bits is of lower importance than other upset modes that may not be protected by FT.

6.2.4 Initial Anomaly Results

Testing of the UT699 was accomplished over several test trips. Early on we tried to use some functional test code that was designed to show how frequently errors occurred. Of course the UT699 was not expected to have errors at all throughout much of the LET range. So when crashes were observed at relatively low LETs (9 MeV-cm²/mg), the test software became a severe hindrance in the effort to figure out what was happening.

The first assumption was that the anomalies were due to multiple-bit errors (MBEs) in either the cache or the register file. Some of the crashes included a signature “Trap 20_{hex}” which is indicative of an EDAC uncorrectable upset in the register file. Because of these assumptions, test flux was a primary suspect for causing the events. To examine test flux as being the culprit the only conclusive approach is to reduce flux until events disappear. But this had to be weighed against realities of our beam testing. We had to take into account that we were already having 30 minute test runs and would need to slow the test runs down another factor of 5–10.

Ultimately this slow beam rate would have been appropriate except that we also improved the detection system. The original test code was factored out. Instead we used detailed assembly-level code with limited use of the Sparc register windowing, which could lead to crashes during window overflow traps. The new code and the reduced flux testing all indicated the SEE response was due to a SEE rather than an MBU, and the new code did not experience the crashes of the original code. Instead we observed that the underlying SEE was a partial reset of registers in the register file.

This example strongly argues for low-level manipulation of the DUT using assembly language when the goal is to provide manufacturers with detailed information about what structures are causing the highest SEE response.

6.2.5 Register Partial Reset

The anomalies discussed in the previous section were isolated with modifications to test code. The events were all found to be consistent with 16-bit sections of registers being set to 0. Because of this characteristic, we defined the event as “register partial reset”.

6.2.5.1 Basic Finding

The LEON 3FT is a Sparc V8 architecture with a register file consisting of 136 EDAC protected 32-bit registers. These registers were tested for upsets in two ways. First, algorithms can monitor the EDAC correction count, which indicates how many times bit errors have been corrected.

Monitoring EDAC counters is not a rugged SEE test approach for two reasons. The testing of this particular reset event type is an example of the problem. The first reason to make sure to detect errors apart from EDAC counters is that test code should verify that EDAC systems work. The second reason is that EDAC systems have an intrinsic limit where they don’t function very well. The EDAC structure is given in Table 6-1.

We will not examine the check bit architecture any more than to say that if all of the data bits in D0-15 are ‘1’s, or they are all ‘0’s, the check bits will all be the same. So the EDAC system cannot detect a change from one state to the other.

Table 6-1. The EDAC check bit architecture used by the UT699. Note that bits 0 to 15 are in bold, 16-31 are not.

$CB0 = \mathbf{D0} \wedge \mathbf{D4} \wedge \mathbf{D6} \wedge \mathbf{D7} \wedge \mathbf{D8} \wedge \mathbf{D9} \wedge \mathbf{D11} \wedge \mathbf{D14} \wedge D17 \wedge D18 \wedge D19 \wedge D21 \wedge D26 \wedge D28 \wedge D29 \wedge D31$
$CB1 = \mathbf{D0} \wedge \mathbf{D1} \wedge \mathbf{D2} \wedge \mathbf{D4} \wedge \mathbf{D6} \wedge \mathbf{D8} \wedge \mathbf{D10} \wedge \mathbf{D12} \wedge \mathbf{D16} \wedge \mathbf{D17} \wedge \mathbf{D18} \wedge \mathbf{D20} \wedge \mathbf{D22} \wedge \mathbf{D24} \wedge \mathbf{D26} \wedge \mathbf{D28}$
$CB2\# = \mathbf{D0} \wedge \mathbf{D3} \wedge \mathbf{D4} \wedge \mathbf{D7} \wedge \mathbf{D9} \wedge \mathbf{D10} \wedge \mathbf{D13} \wedge \mathbf{D15} \wedge \mathbf{D16} \wedge \mathbf{D19} \wedge \mathbf{D20} \wedge \mathbf{D23} \wedge \mathbf{D25} \wedge \mathbf{D26} \wedge \mathbf{D29} \wedge \mathbf{D31}$
$CB3\# = \mathbf{D0} \wedge \mathbf{D1} \wedge \mathbf{D5} \wedge \mathbf{D6} \wedge \mathbf{D7} \wedge \mathbf{D11} \wedge \mathbf{D12} \wedge \mathbf{D13} \wedge \mathbf{D16} \wedge \mathbf{D17} \wedge \mathbf{D21} \wedge \mathbf{D22} \wedge \mathbf{D23} \wedge \mathbf{D27} \wedge \mathbf{D28} \wedge \mathbf{D29}$
$CB4 = \mathbf{D2} \wedge \mathbf{D3} \wedge \mathbf{D4} \wedge \mathbf{D5} \wedge \mathbf{D6} \wedge \mathbf{D7} \wedge \mathbf{D14} \wedge \mathbf{D15} \wedge \mathbf{D18} \wedge \mathbf{D19} \wedge \mathbf{D20} \wedge \mathbf{D21} \wedge \mathbf{D22} \wedge \mathbf{D23} \wedge \mathbf{D30} \wedge \mathbf{D31}$
$CB5 = \mathbf{D8} \wedge \mathbf{D9} \wedge \mathbf{D10} \wedge \mathbf{D11} \wedge \mathbf{D12} \wedge \mathbf{D13} \wedge \mathbf{D14} \wedge \mathbf{D15} \wedge \mathbf{D24} \wedge \mathbf{D25} \wedge \mathbf{D26} \wedge \mathbf{D27} \wedge \mathbf{D28} \wedge \mathbf{D29} \wedge \mathbf{D30} \wedge \mathbf{D31}$
$CB6 = \mathbf{D0} \wedge \mathbf{D1} \wedge \mathbf{D2} \wedge \mathbf{D3} \wedge \mathbf{D4} \wedge \mathbf{D5} \wedge \mathbf{D6} \wedge \mathbf{D7} \wedge \mathbf{D24} \wedge \mathbf{D25} \wedge \mathbf{D26} \wedge \mathbf{D27} \wedge \mathbf{D28} \wedge \mathbf{D29} \wedge \mathbf{D30} \wedge \mathbf{D31}$

Once the test algorithm was able to identify this register partial reset event, we were able to focus detection. Cobham Gaisler further provided detail that the 32-bit register, with 7 check bits, was implemented as three 16-bit library elements. The check bits for both data patterns (all ‘0’s and all ‘1’s) both have two check bits on and the remainder off. Thus, we expected that 1/3rd of the register partial reset events would manifest as EDAC uncorrectable errors, and require the test algorithm to be prepared for these events. That is, of the three 16-bit structures that make up an EDAC word, only those that corrupt the check bits would manifest as uncorrectable errors. Partial reset of the two 16-bit regions storing data would result in EDAC words that looked correct to the EDAC system. The results of the focused detection of the partial resets and EDAC errors are shown in Figure 6-4. This figure also highlights two types of flux dependence discussed earlier. The direct flux dependence was tested by reducing the flux and noting the cross section did not reduce. The fluence was also reduced, which would highlight sensitivity of hidden elements. Since both would tend to reduce the sensitivity and no reduction is seen, the data are inconsistent with a flux or SEE buildup problem due to accelerated ground testing.

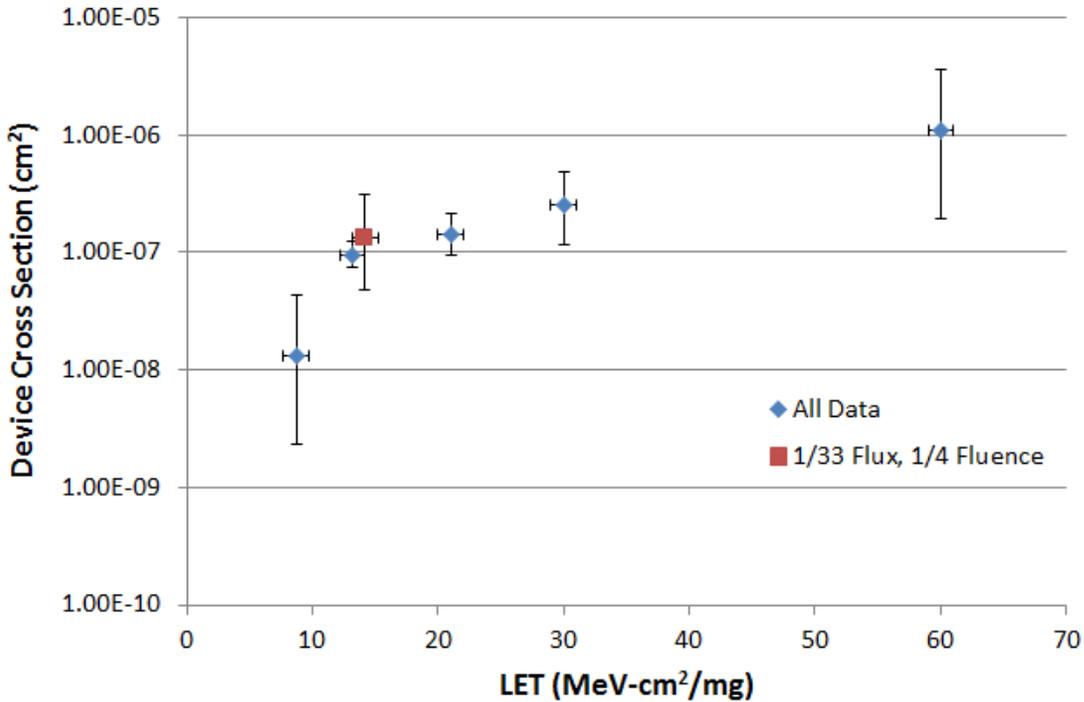


Figure 6-4. Device cross section for register partial reset events on the UT699. Also shows that lower flux and lower fluence did not change the cross section [34]. Image used with permission

6.2.5.2 Space Rate

The register partial reset of the UT699 is the only event topic (aside from manufacturer published sensitivity) where the resulting space rate from the observed events is relevant. We note that the UT699 has a predicted space rate for errors given in [30] and [52]. The former indicates that the worst case multi-bit error rate is $6.7 \times 10^{-11} \text{ s}^{-1}$ during a flare. The latter indicates that in geosynchronous orbit (galactic cosmic ray [GCR] environment) the UT699 will have a FF-dominated event rate of about one in 74,000 years. The event rate for the register partial reset is $1.6 \times 10^{-13} \text{ s}^{-1}$, or less than one in 200,000 years. Thus, this event type is not going to be the predominant event type in these orbits. (Under specific proton-dominated orbits, the low-LET sensitivity of the register partial reset may make it more important than FF upsets.)

6.2.6 SpaceWire Testing

The UT699 has four SpaceWire ports. We tested these ports for upsets during data transfer. The test software was designed to operate these ports in full duplex mode with data transfer occurring in an infinite loop until the test engineer ends operation. Using assembly language it was possible to configure two ports to transmit data simultaneously. The data are summarized in Table 6-2.

Table 6-2. Summary of the results of the SpaceWire testing.

LET _{eff} (MeV-cm ² /mg)	Fluence (cm ⁻²)	Transferred Bits	Bit Errors
13.2	4.2e7	1.4e10	0
21	2.4e7	9.4e9	0
30	6.0e6	4.2e9	1
60	4.5e5	1.5e10	0

Because of the bit error observed in the LET_{eff} = 30 MeV-cm²/mg there are three ways to interpret the cross section. The device cross section may be taken as $1/(6 \times 10^6) = 1.7 \times 10^{-7} \text{ cm}^2/\text{device}$. Or it can be taken as the transferred bit cross section of $1.7 \times 10^{-7} \text{ cm}^2/(4.2 \times 10^9) = 4.0 \times 10^{-17} \text{ cm}^2/\text{transferred-bit}$. And finally, it can be interpreted as the cross section for upsetting a buffer bit (since the transfer bit error likely occurred in a temporary storage location). However we do not have a good estimation of the number of bits involved.

Since our testing was performed at full duplex, and the data transfer was occurring more than 25% of the time, the relevant sensitivity is that given above within a factor of four.

6.2.7 Watchdog Testing

The UT699 has a built-in watchdog circuit. This circuit can be used to monitor the execution of the UT699. Although this puts the device and its monitor in the same component, which is therefore not independent, this approach was used for some testing.

The qualitative result is that we did observe events where the watchdog circuit was not triggered, but communications with the UT699 were lost. This occurred with high flux and fluence tests performed to get test data rapidly. We also observed one event, under regular test conditions (as used for the rest of the data collection in this section), where the configuration of the UART port was corrupted and had to be fixed by the test code. It is believed the failure of the watchdog to detect problems is related to the quality of the verification code (part of the test software), where the UT699 test code must determine all is well before resetting the watchdog.

6.2.8 UT699 Conclusion

This section has provided details of testing the Cobham Gaisler UT699 as it pertains to the SOC test methods development. We examined test development, identification of upsets, flux and fluence dependence, and testing of an I/O system. Future testing of similar devices (such as future Sparc devices from Atmel or Cobham) will benefit from the findings of this study.

6.3 Example from Maestro ITC Testing

The Onboard Processing Expandable Reconfigurable Architecture (OPERA) program has produced a 49-core multicore processor based on the Tiler architecture, named Maestro [8,20,30]. The Maestro ITC (Interim Test Chip) microprocessor, built by Boeing Solid State Electronics Development (SSED), is designed for space applications utilizing Boeing's participation in the DTRA 90 nm radiation hardened by design (RHBD) program [52]. Intellectual property (including the Maestro processor) from the OPERA program is available to all United States Government agencies for space use.

The Maestro ITC provides a very good example for developing SOC SEE testing. As a many-core device, it presents a solid example of a device architecture that is clearly a system on a chip, a complex device, and likely to have its general structure reflected in future devices. Thus, test methods for this device will be useful as general test methods for a large sample of future SOCs.

6.3.1 Seven Work Points

This device provides a good example of every one of the seven work points, as indicated below.

6.3.1.1 Collaboration

Since Maestro is one of the possible future directions for microprocessor architecture, and it is a very complex microprocessor, there is considerable interest in collaboration. The Maestro ITC, as an RHBD offering from an aerospace manufacturer, has considerable manufacturer collaborative interest.

What is relevant here is that we found significant boundaries to sharing software and significant difficulty in moving test efforts from one group to the next. The software sharing is best handled by developing a formal collaboration before work starts.

6.3.1.2 Peripheral Approach

Although none of the discussion here is directly about peripherals, some test data indirectly provides results for peripherals. Further, the Maestro ITC provides an excellent platform for testing several types of IO devices. The on-chip networks also present interesting cases for examining peripheral test approach. The five networks fall into two general categories—the automatic networks that handle memory synchronization (the TDN and MDN – see Section 1.4.3 for more detail), and the networks that are activated when messages are explicitly routed for transfer (the STN, IDN, and UDN).

6.3.1.3 Fault Tolerance

The Maestro ITC, with 49 cores, was too large to accommodate RHBD SRAMs. It employs FT on many different structures on the device. The L1 data and instruction caches are protected with parity bits that allow errors to be identified and correct data to be re-fetched from the L2 cache. The L1 and L2 tags and L2 data are protected by EDAC. The sheer size of the Maestro ITC posed significant problems for flux dependence and identifying true SEEs in the sea of upsets to FT structures. The key findings are that each point at which data is collected requires an established sensitivity level, and testing must include a handling approach to deal with multi-bit upsets. A notable exception is that the on-chip networks are not FT.

It is important to point out that Maestro is not specifically designed with system-level FT support. This is important particularly in multi-core devices. Lack of FT on tile-to-tile communications means that even if multiple cores are configured to execute TMR, the communications networks provide an additional common-mode error path in addition to other potential mechanisms. The good news for testing is that the lack of FT on these networks makes it easier to test them for SEE, though significant SEE sensitivity in the MDN and TDN is expected to lead to problematic error modes.

6.3.1.4 RHBD

The Maestro ITC is a mixed RHBD/FT device. As indicated in the section above, the FT portion makes testing the RHBD portion difficult because typical test fluxes are likely to cause tile crashes. The remainder of the IC, however, is built from RHBD structures. This includes the clock distribution, tile registers, IO registers, and the FFs used to build the tile pipeline. A notable exception is that the on-chip networks are not RHBD.

6.3.1.5 Multicore

As a 49-core device, the Maestro ITC is considered an extremely good example of multicore for this guideline's objectives. The key findings regarding multicore come from debugging tools and message passing. It is very difficult to establish useful test algorithms for message passing because you must know the utilization factor for your target, and with multicore hypervisors it is not clear what the utilization factor is without considerable knowledge of how a typical hypervisor works. This knowledge realistically is obtained by analysis of an existing OS implementation. But once significant knowledge like this exists, the key algorithms can probably be written such that key operations can be directly studied and the OS is not required. A worst-case approach of directly testing an OS is a possibility; but for the existing Maestro ITC, the data cache parity error will likely overshadow the desired result.

6.3.1.6 General Test Methods

The Maestro ITC is challenging for testing for several reasons. Under normal operating conditions it runs very hot. It is not possible to obtain inexpensive demonstration boards. And it contains so many on-chip circuits that it is difficult to select a good arrangement of test algorithms to cover the general operation. Key findings for general test methods include the following. Device preparation for test of flip-chip devices without heat spreaders can proceed with thinning of the substrate. But testing can also be performed with limited LET by using higher energy beams that can traverse the substrate. Another alternative is to test at very high energy facilities (such as NSRL), but this requires very expensive beam charges. Additionally, test boards where the DUT is positioned very far from the edge of the board should be avoided. Maestro testing also highlighted the need to use manufacturer provided debugging tools, especially to analyze unresponsive processors. Maestro also highlighted a minimal test hardware configuration where there is no additional memory available to the Maestro ITC other than that which is on the chip.

6.3.2 Background

As indicated above, the Maestro ITC is a 49-core multicore device built by Boeing under the OPERA program. The Maestro ITC is built as an example of the 90 nm RHBD library. The device is also a custom implementation of the Tiler architecture. The block structure of the Maestro ITC is shown in Figure 6-5.

Testing was performed using thinned devices tested at TAMU, and using thinned and unthinned devices at NSRL. Test algorithms targeting the tiles and some of the peripherals were developed and used to collect data.

Maestro is of considerable interest for flight projects requiring significant on-board processing. Because of the multicore structure there is considerable interest to bring up redundant tiles performing mission critical operations. Some examples of flight-like algorithms that could target Maestro are the CRBLASTER [98] and ROCKSTER [4] applications. Maestro has the potential of enabling autonomous spacecraft tasking, and for handling on-station image analysis and prioritizing for data transfer to Earth.

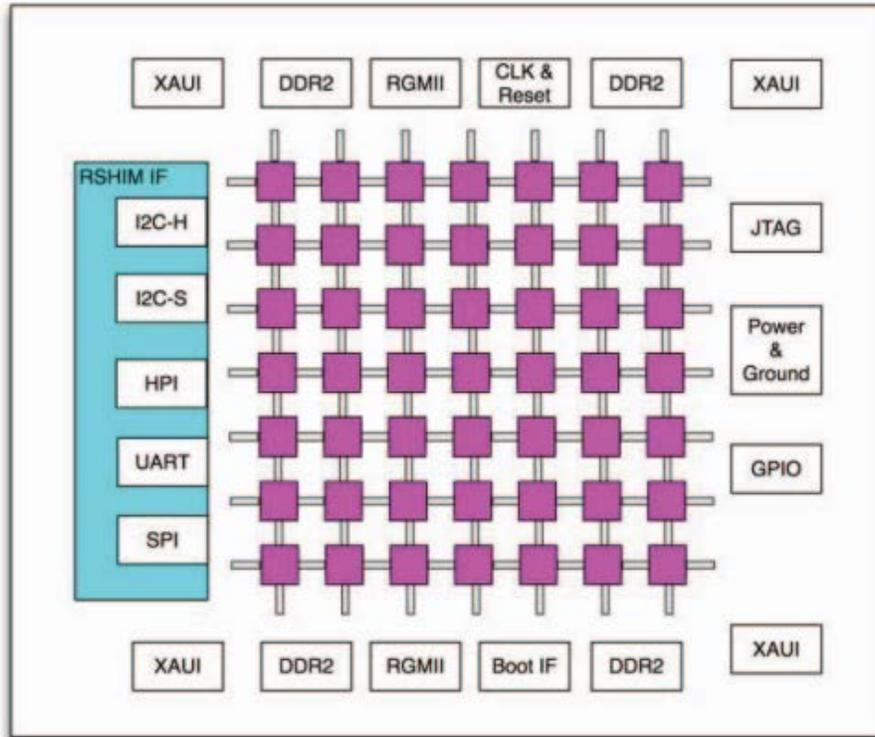


Figure 6-5. The block layout of the Maestro device. The main structure is the 49-tile array. It is surrounded by four DDR2 controllers, four XAUI ports, and a handful of additional peripherals. The tiles and peripherals are connected with five on-chip networks. Image used with permission [C. Rogers et al., "The Maestro Flight Experiment...", 2016]

6.3.3 Test Board Info

As indicated above, Maestro does not have an inexpensive demonstration board that can be used for testing. Since the test efforts included significant collaboration, testing was performed utilizing the functional test board (FTB) developed by Boeing for functional verification of devices after packaging. The FTB is a hardware verification platform, and as such is ideal for testing the DUT. The FTB can be seen in Figure 6-6 mounted on the target table at TAMU. The position of the DUT is central on the board and the board is very large (about 30 cm on each side). This led to problems with the position of the DUT. It was almost impossible to mount the FTB so that the targeting table could hold the FTB and the DUT.

Although the FTB is not ideal in its relative position of the DUT, it does provide an excellent example of almost every other aspect of an SEE test board. In particular it has the following attributes:

1. All support devices are significantly far from the DUT
2. The DUT is socketed
3. All power supplies are sourced remotely via board terminals

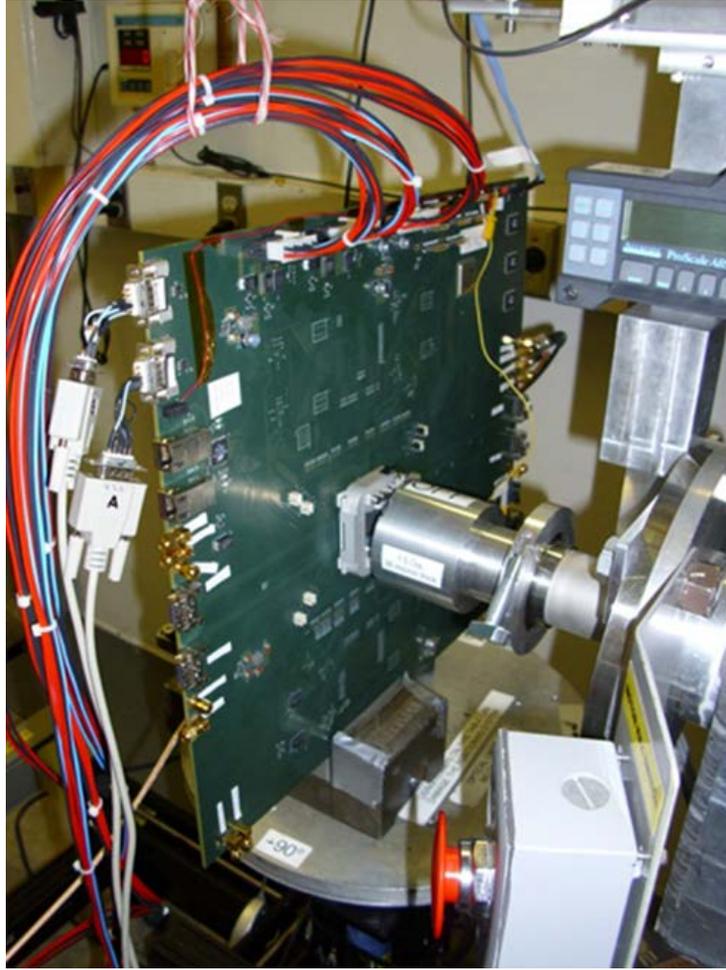


Figure 6-6. The FTB setup at TAMU. The beam pipe is directed at the DUT. The position of the DUT on the FTB is close to the center, and it almost does not fit in the beam line at TAMU. Image used with permission of Jet Propulsion Laboratories

6.3.4 Device Preparation

The Maestro ITC is a flip-chip device that requires a heat sink in normal operation. For our testing the processor was run at a low core frequency (100 MHz), and only a couple slow IO interfaces were used. We also limited the amount of time that the DUT was powered. Because of these use parameters, the DUTs did not get very hot during testing (not hot to the touch), and we did not need to use a heat sink. Unthinned and thinned devices are shown in Figure 6-7.

Thinning to approximately 80 μm enables most of the low energy beams (~ 15 MeV/amu) to penetrate at normal incidence. However, since Maestro is an RHBD device the sensitivity at angle is more important than in other types of devices. This occurs because RHBD structures are often based on reinforcing feedback, and ions at grazing angles can cause transient currents in multiple circuit nodes. Unfortunately, because of the socket size, angular testing was not performed on the Maestro ITC.

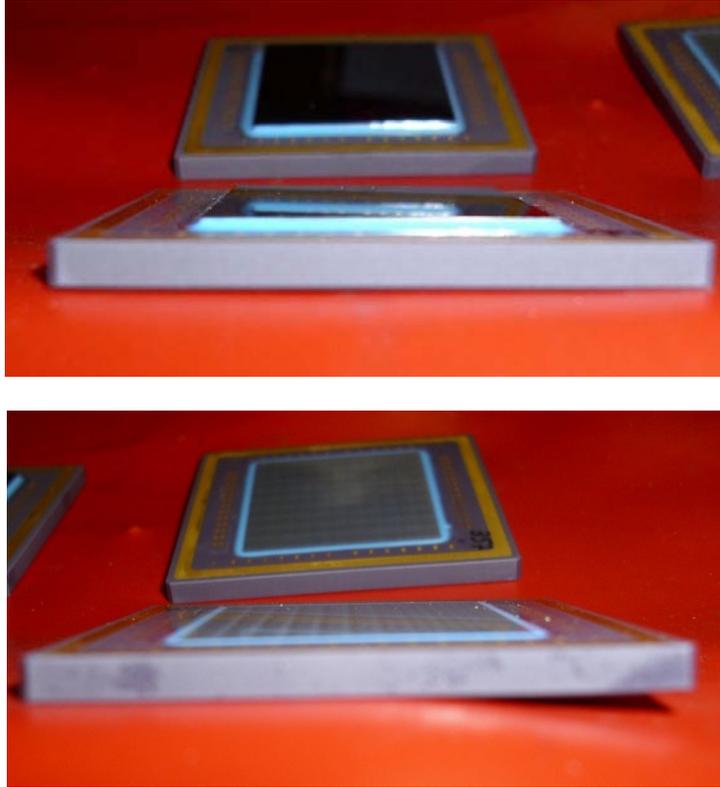


Figure 6-7. Unthinned (top) and thinned (bottom) Maestro ITC test devices.

The final thickness of devices is approximately 80 μm . Photos of backthinned devices used with permission of JPL.

6.3.5 Algorithm Architecture

Maestro is a many-core device, and algorithms for its SEE testing are unique. Maestro provides a very good example of possible future test algorithm structures. For this reason we present some details of the algorithms used during testing of the Maestro ITC.

6.3.5.1 General Algorithm Structure

Testing of Maestro was primarily performed by utilizing the Maestro-specific version of the Tileria BTK. The algorithms were then modified to support self-interrogation for data collection, and to make the test algorithms operate for indefinite periods of time. The initial test algorithms were designed for hardware verification, not for soft errors. Thus, they were modified to provide continuous sensitizing for soft errors. The master tile code was modified to perform periodic reporting also for indefinite periods of time.

Master tile to test tile communication was performed using the Tileria distributed L3 cache concept where the MMU on each tile is programmed so that it interprets a section of its local memory to actually be physically located on the master tile. Using this method we established a region on the master tile where all tiles were allocated a unique section based on tile coordinates. Although the tile data could also be transferred by the UDN, STN, and by shared off-chip memory, the direct memory mapping also provides exposure of the MDN and TDN during irradiation and off-chip memory was not available under the test hardware used.

The standard test algorithm is shown in Figure 6-8. This figure shows the elements discussed above. The concept is to have the test code setup operation then transition to a test loop that includes an operation to be performed during “dwell” as the main test. And finally the test loop performs periodic scrubbing of its own operation.

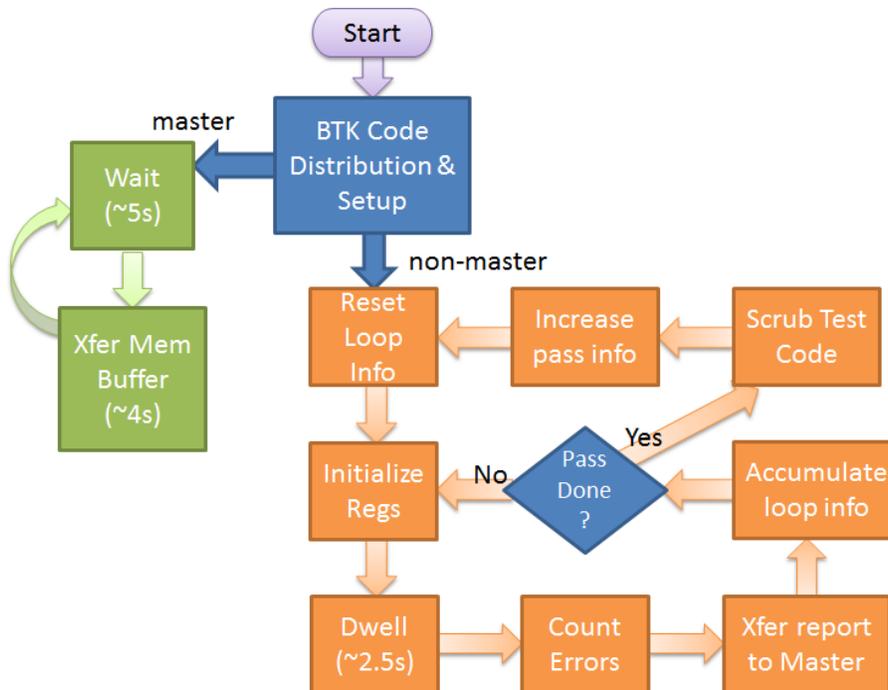


Figure 6-8. The general algorithm for test code used on Maestro. A single tile is chosen as the test master that monitors the test reporting from the other tiles. Photo used with permission of JPL.

The following subsections describe the primary test algorithms used during dwell.

6.3.5.2 Register Static Testing

One common algorithm for testing is to store values in registers, then dwell during irradiation, and finally read the registers and record differences from expected values. This approach is limited because for Maestro we did not wait for the beam exposure to end before ending dwell and reporting data. Thus, beam-related errors could occur during comparison.

6.3.5.3 Register Dynamic Testing

There are two primary ways that registers can be upset during radiation exposure under dynamic operation. The first is incorrect assignment, where an error occurs during hardware access of a register. For example an attempt to assign a computation result to register 2 could result in writing to register 3 instead or as well. The second is that the actual dynamic operation could result in an incorrect value being stored in a register. Both of these effects would be due to SETs on control or calculation logic lines.

Dynamic register testing consists of using registers to perform operations during the dwell period. Often times the actual algorithms used are things like fast Fourier transform (FFT), or some other calculation. Unfortunately there is not a particularly good algorithm for determining radiation sensitivity. An OS-like algorithm can be used to determine how a real OS will perform, but that doesn't provide information about how many structures were being used during testing. Thus, for most dynamic testing the major problem is normalization to number of structures and duty cycle.

6.3.5.4 L1 Cache Testing

Whenever possible L1 cache testing should be performed primarily with the cache disabled. (Testing with the cache enabled should also be performed to ensure the error rate, while the cache is enabled, is not significantly different. But this can be difficult because test algorithms also speed up with the cache enabled.) For Maestro the L1 cache could not be disabled. However, the L1 instruction cache has no useful

means to measure errors, while the data cache can have its error protection disabled and errors can be read directly. Because of the errata in Maestro where the L1 data cache does not correctly “miss” on a parity event, we were able to measure L1 bit upsets by directly observing the L1 data cache after writing a known pattern and observing read values that were incorrect.

6.3.5.5 L2 Cache Testing

The L2 cache on each of the Maestro tile processors is 64 kB and is protected with 5 EDAC bits for every 8 bits of data. Other than the EDAC hiding errors, the L2 cache testing is essentially the same as the L1 cache testing. The only problem is that in order to observe L2 cache upsets we had to disable the L2 uncorrectable error interrupt and observe the data directly (or to be more precise, the L2 uncorrectable interrupt was not enabled, and it powers up in a disabled state). The actual means of testing is the same as for the L1 test. The L2 cache is loaded with known values, the beam strikes the array, and after a predetermined amount of time the stored data is read to check for errors. The only difference relative to the L1 test is in the detection of errors. Detection requires two or more upsets in an 8-bit byte. See more on this in 6.3.6.

6.3.5.6 Incidental Observation

The items above were specifically designed into some or all of the test software. On-site operations were able to focus on these algorithms and verify they performed as expected. Other targets, however, were tested incidentally, and the test algorithms employed could not be examined for effectiveness. Some of the items tested include the following:

1. Performance of the MDN and TDN due to memory sharing in the test algorithms
2. Performance of the IDN in transferring communications packages from the master tile to the UART
3. Performance of the special purpose tile registers that configure each tile for operation.

Incidental observation is important because it provides a major part of test results. SOCs are so complex that the observation that things work well (even without well-defined stimulus) should be reported along with details about how the test algorithm operates.

6.3.6 Test Sensitivity

The multiple levels of fault tolerance and high count of cores in Maestro makes test sensitivity very important. When testing is performed with the test device playing an important role in sensitizing itself, the ability to extract reliable data is important. For the Maestro ITC it was determined that tile crashes were the event type that limited SEE detection. The tile crash rate is flux sensitive (in space environments the flux is much lower and tile crashes have a much lower rate than in ground testing). So, the flux could be reduced to increase test sensitivity. Additionally, test sensitivity can be improved by determining the signal-to-noise ratio and then increasing the observation time to ensure statistical reliability of the signal. Unfortunately, both of these require increasing exposure time (either due to lower flux, or increasing observed events at a given flux) which was already too limited, so neither was examined for Maestro testing. The resulting test sensitivity, determined from the fluence where about 50% of the test tiles were expected to crash, is show in Figure 6-9.

The observed register errors during static testing also highlight a key concept in test sensitivity in that our algorithm was fortuitously set up to observe that errors occurred in correlation with specific parts of the algorithm. Since the test sensitivity should have been constant through those different parts of the algorithm, the observed errors were discounted as artifacts, improving noise suppression. Tailoring of algorithms to reduce erroneous counting is a very important method for increasing test sensitivity.

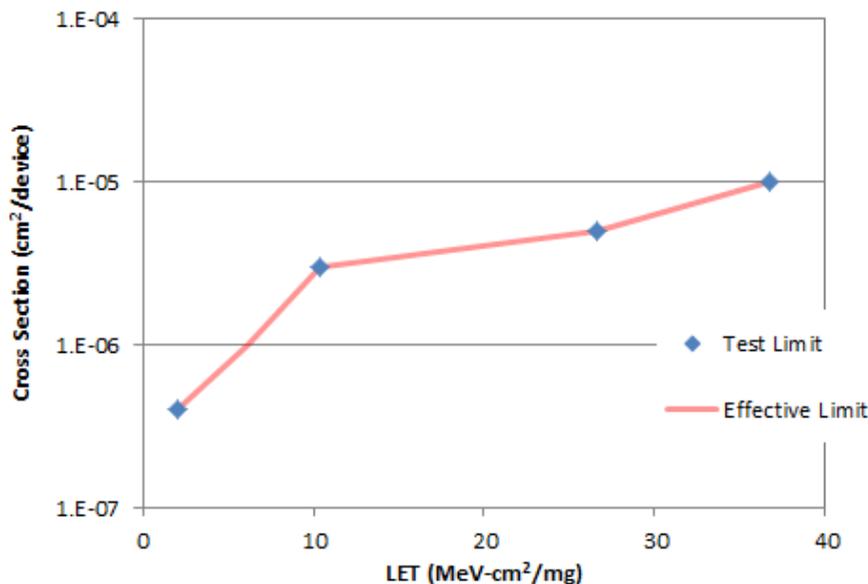


Figure 6-9. Effective sensitivity for detection of SEEs during Maestro ITC testing. Sensitivity is limited by the fluence that when delivered to the DUT results in approximately 50% of test tiles crashing [30]. (This is also Figure 4-1, duplicated to assist reading. Figure used with authors permission.)

6.3.7 Cache Testing—RHBD vs. Non-RHBD

As with most microprocessor testing, work on Maestro started with testing the caches. They provide a very large target with a lot of bits to observe for bit flips. The Maestro ITC caches are also a good example of an FT structure that must not be overly examined when their error rate is very high, but FT is supposed to suppress the errors in a real application. Instead, for the Maestro ITC, cache testing must be done in such a way as to enable testing of other RHBD structures on the device. Unfortunately, for our testing, the lack of on-board main memory made it very difficult to examine the chip beyond upsets in the caches because invalidated cache lines could not be fetched and the executing code would crash.

6.3.8 Debugging with BTK

Maestro is built on the Tileria Tile64 processor. Part of the software support for Maestro is that it is largely supported by Tileria’s Board Test Kit (BTK), which allows debugging of the Maestro ITC. One major observation of the testing effort discussed here is that tiles would crash. This behavior is not desired, especially on a device like Maestro that is RHBD and FT. Thus, it was very important to be able to analyze the crashes on the Maestro tiles in order to verify that the crashes were the result of the accelerated ground testing environment. In order to do this we used the BTK. The debugging tools in the BTK include the ability to examine the program counter and the cache contents (including the cache tags).

We determined that the ability to walk through analyzing a lock up on a processor in the lab should be used to establish a viable approach to tool usage. A simulated infinite loop can be used to ensure the processor can be interrogated. Then, given the register and cache contents (especially pointers and program counter), the ability to trace the processor operation to the relevant source code is desired.

6.3.9 Data Presentation

The best way to present data on a processor or SOC is to start by relating the test work to known structures with well-established radiation response, then build from there. This is often done by analyzing the SEE sensitivity of the caches because they are a very large target and test methods are relatively straightforward. Once the cache response is known, the results can be used to assist in normalization of other measurements.

Due to limited development time and lack of access to Maestro systems capable of running normal software, the only additional items tested were the processor registers in both static and dynamic (performing operations) modes. Incidental observation of the performance of the MDN and TDI occurred, along with observation of the UART, but none of these constitutes a meaningful measurement.

6.3.9.1 Cache Results

For Maestro the caches consist of L1 instruction, L1 data, and L2 caches. Because of detection problems, the L1 instruction cache is not testable unless the BTK is used to examine contents of the caches, and that is too slow to be carried out on the beam. The results of cache testing on Maestro are shown in Figure 6-10. Note that the L2 cache appears to have a higher cross section for what are expected to be the same bit-level structures. One potential explanation for this is that the L2 cache only shows upsets when two or more bits in an EDAC-protected byte are flipped due to uncorrelated separate hits in the same EDAC-protected region. But given the feature size, it is possible that these will tend to occur in clusters, even though the bits in a given EDAC-protected byte are physically separated (i.e. the first event upsets multiple bits in multiple EDAC-protected bytes – with no uncorrected errors; then the second event upsets another set of multiple bits in some of the same protected bytes – resulting in multiple events that are not correctible). It is important to note that we are intentionally overwhelming EDAC in order to observe events, and that the artifacts of the EDAC system are thus very important to interpreting the results.

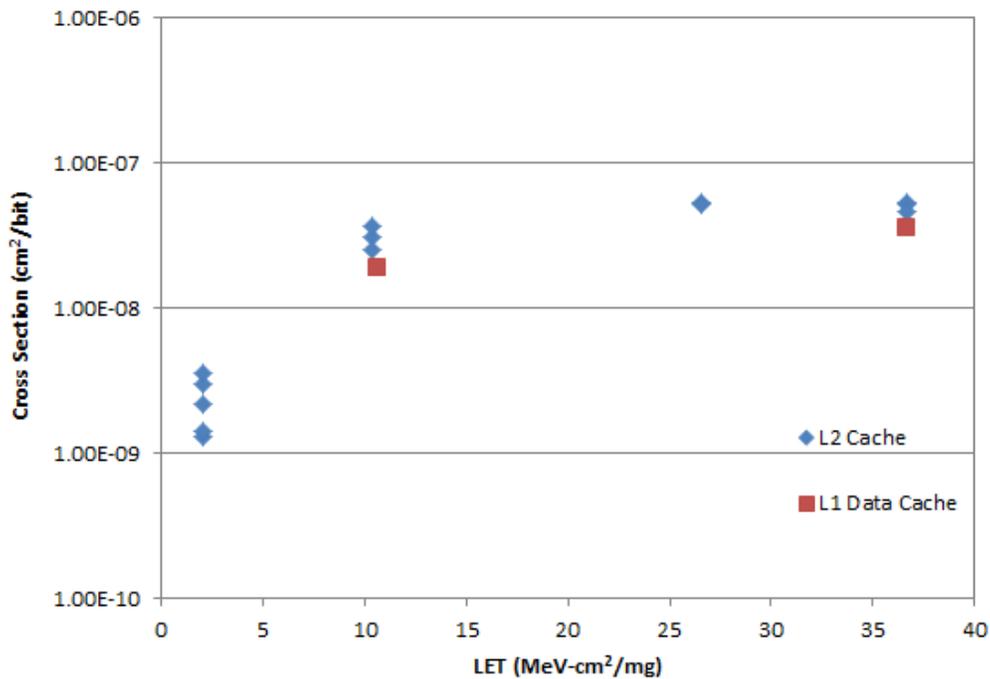


Figure 6-10. L1 data and L2 cache bit sensitivity of the Maestro tile processors. There is considerable spread in the data (especially L2) but these data are consistent with the SRAM sensitivity reported [44]. Figure used with author's permission.

The formula used for calculating the cross section of L2 bits is given by (11). Note this is the same as equation (10) in Section 5.3.5.

$$\sigma = \frac{1}{\Phi} \sqrt{\frac{N_p N_{Err}}{N_{Eff-Tile} M \left(\frac{13 \cdot 12}{2} \right)}} \quad (6-1)$$

Here N_p is the number of passes, N_{Err} is the number of observed errors, $N_{Eff-Tile}$ is the number of effective tiles that contribute to the count, M is the number of bytes being tested, σ is the cross section per bit, and Φ is the fluence. The combinatorial factor in the denominator reflects the fact that an EDAC protected byte consists of 13 total bits so that 5 check bits are used to provide protection for the 8+5 bit “byte”.

6.3.9.2 Register Results

Register testing was performed both statically and dynamically. Static testing showed a handful of events, but correlation with the test algorithm was found. Such correlation is inconsistent with the nature of the sensitizing algorithm, so although the results are right at the sensitivity level, we conclude static register upsets are below the test sensitivity.

Dynamic register results were similar to static except that the events were observed to occur independently of the state of the algorithm. No bit upsets were seen in any registers during dynamic testing. Instead clobbers were seen where the contents of the register are not related to their expected values by just a simple bit flip, and since the operations are not unstable (i.e., like a calculation with sensitive dependence on values encountered during a sequence of calculations). The results for dynamic register sensitivity, compared to the test sensitivity are shown in Figure 6-11. An example of the type of clobber seen during testing is that the rotate operation occasionally had part of a register overwritten by bits that were not rotated or were over rotated.

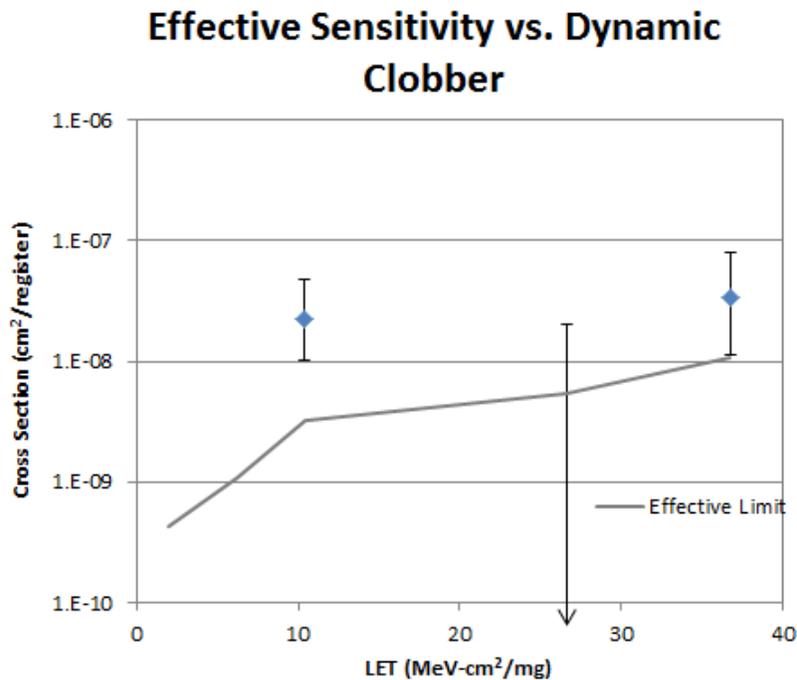


Figure 6-11. The register dynamic clobber cross section compared to the test sensitivity shows a response that is separated from the noise floor. Figure used with author's permission.

6.3.10 Maestro ITC Conclusion

Maestro ITC testing was performed as a significant collaborative effort by multiple NASA centers, government, and industrial groups. The collaboration was mostly on the component test side as opposed to the user side. Collaboration directly included hardware development, test software development, and data analysis, as well as important collaborative discussions and early efforts that were ultimately leveraged into the final test effort reported in [30]. Good test data were collected, and a lot of information was developed regarding how to test complex SOCs.

Maestro is clearly an excellent example for the key elements of this guideline, including manufacturer collaboration, and testing of RHBD, FT, and multicore devices. Each of these was met with special effort to enable testing that would collect relevant results. The participation of the manufacturer and the sponsoring government agency were instrumental in all aspects of the testing. The multicore testing was performed with specialized software that enabled simultaneous testing of all the cores in the Maestro ITC. The FT capabilities of Maestro were tested and formed a major portion of the test software strategy as there was no off-chip memory, and the test code had to reside in the FT L2 cache. On the user side, the FT architecture still needs to be tested as the user thrust is more interested in the ability of Maestro to detect cores that fail and develop software approaches that can work around them, and common mode failures still need to be tested (though none were observed within the test sensitivity). The RHBD architecture of Maestro is very important for future testing, as the frequency and angular dependence of SEE response are both critically important.

In order to perform testing to meet frequency and angular needs of RHBD and FT devices, hardware design for test must include these considerations early. For the Maestro ITC it was known that hardware to support large angle and high frequency testing was simply outside of the plan. Additionally, however, an unexpected difficulty arose in that the FTB was almost too large to be used for testing at TAMU because the DUT was positioned near the middle of the board, so it was not possible to rotate the board to get it lined up better with the targeting system. At high speed, Maestro is expected to have heating problems, and the FTB and test equipment were not configured for high speed operation. The test chips were thinned with relative ease because ICs as large as Maestro are mounted with greater precision and less die flexing than many smaller die. The Maestro ITC testing therefore is a good example of many of the difficulties involved in test hardware development.

A major finding of the Maestro test effort is the potential benefit of debugging software. There are two major places where this software comes in to play. The first is in test software debugging and becoming familiar with how the device operates. But the second, and more important for SEE testing, is that debugging tools can be used to interrogate a crashed processor. The debugging tools available from the Tileria BTK proved invaluable in showing that crashed tiles were indeed attempting to access L2 cache regions that had become corrupt (and since there was no off-chip memory, the corrupt data could not be reloaded). In future devices with more and more on-chip operations without external signaling this type of debugging approach will be more valuable. A potential problem exists in that the device response is unknown at the time of test planning, and a rational approach to how to spend post-beam investigation of crashes must be established.

The Maestro ITC testing would have benefitted from a more global test approach with an end goal of testing a few user applications. There is a common trap in testing a complex device, especially one like Maestro that is clearly built from library elements designed to have common SEE responses. The trap to which we refer is that often initial test results are so difficult to get that testing is not developed beyond testing of basic structures. Maestro, in particular, presents a difficult situation because end users are interested in fault tolerance that cannot really be tested by testing the low level structures. Thus it is important to have a transition plan from low level test structures all the way up to application-like testing. This problem is further stressed by the collaborative test efforts that are needed for SOCs. The collaboration element of the 7-point plan includes having user participation. Thus far, there is very little user collaboration included in this work in a direct manner, but through discussions we have identified key user lines of investigation. We have found that some users accept the low-level test results, while others do not feel comfortable with how that data is extrapolated to full application behavior. For these reasons, we recommend that user-type applications be used for testing.

Limitations of application-like testing also highlight one of the strongest recommendations for additional SEE testing of the Maestro ITC (i.e. some users may desire testing with actual application-like test software). Testing application-like software, in addition to testing more of the support circuits (especially the on-chip networks), rounds out the recommendations for future work.

6.4 Example from Freescale Testing

This section provides sample results from testing of Freescale SOCs. Recent testing targeting the SOC capabilities of these devices has provided insight on commercial device testing. Because they are inherently less expensive than RHBD or aerospace specific devices, they can be used to study SOC radiation phenomena—especially the impact of SEEs on systems—at significantly reduced cost. They are also advanced relative to state of the art aerospace devices, so test methods can be tried out on these devices before transferring to future aerospace devices that will be built on similar architectures in the future. The test examples provided here cover the P2020 dual core e500 and P5020 dual core e5500 SOCs.

6.4.1 Seven Work Points

The Freescale SOCs can be contained as test efforts to individual test groups, and are inherently not RHBD devices. These are reflected in the contribution to the SOC guideline's seven working points as indicated.

6.4.1.1 Collaboration

Collaboration with manufacturers on commercial devices is critically important and usually not available. As an example, the target regions of a device are generally unknown to the SEE test group, but because of thermal issues not all of the device can be irradiated simultaneously. This can lead to wasted effort when trying to characterize portions of the device that are not being irradiated.

For some of our testing we were able to pool resources with other users of these devices in order to streamline processes. Often, however, these devices are inexpensive and readily available along with significant development tools, so that it becomes relatively easy for groups to put together independent test campaigns.

Collaboration on P2020 testing included assistance from Space Micro. Collaboration on P5020 testing included assistance from BAE Systems.

6.4.1.2 Peripheral Approach

The P2020 and P5020 are both complex SOCs with considerable peripheral circuits built into each device. Significant resources exist to enable operation of these peripherals. Thus far we have collected data on the memory management operation of the P2020, and independent operation of two UART ports. These can also be used to study SERDES communication and a host of other peripherals.

6.4.1.3 Fault Tolerance

Both devices provide parity protection for processor L1 caches and EDAC protection for L2 caches. The P2020 has been observed to automatically support L1 miss on cache parity errors (in fact it appears it may be impossible to turn this off, though parity exceptions can be used to catch the upset). No other significant FT is provided on the device aside from FT protocols run over IO devices.

6.4.1.4 RHBD

The Freescale commercial microprocessors are not RHBD. Future developments from BAE, based on the Freescale library, will be RHBD. Thus, knowledge of how to test the Freescale devices and their SEE sensitivity will be valuable for evaluating the effectiveness of methods applied to make future RHBD aerospace microprocessors based on the Freescale library.

6.4.1.5 Multicore

The P2020 and P5020 devices are both dual core. We also examined the P4080 (an 8-core e5500 device) as a possibility for this work but found that our hardware modification requirements resulted in unstable

operation of our test boards due to heat issues. The P2020 and P5020 primarily utilize cache coherency for communication.

Testing of these devices has shown that the P2020 e500 cores do not have a significantly increased probability of common-mode crashes.

6.4.1.6 General Test Methods

Freescale SOCs provide an excellent basis for developing test algorithms for next generation aerospace devices. They are generally quite well-supported in terms of documentation and hardware to enable operation of the various peripheral functions they have. They also require careful hardware selection because of device modifications to both the DUT and the test board required for heavy ion testing and to minimize test cost when many test boards are available (i.e., careful selection from a handful of options may result in significantly reduced cost when multiple DUTs are required). These devices also provide good examples for addressing development of a test matrix, especially as it concerns different ways to run a device and what peripheral tests to run.

6.4.2 Background

Freescale devices, and those from their earlier PowerPC™ partners, are quite common in the space market including many instances of the RAD750 processor from BAE [3]. This device has been used in many different missions including Deep Impact, the Mars Science Laboratory, and many other missions. The RAD750 was leveraged off of the IBM PowerPC™ 750 processor and is produced at the 150 nm and 250 nm feature sizes.

6.4.2.1 Current Products

The current batch of Freescale SOCs are produced at the 45 nm feature size, and some of this technology is slated to be built into a new space computing platform by BAE using the 45 nm e5500 core and associated peripheral devices from Freescale [54].

6.4.2.2 Context with Earlier Testing

Significant information exists on testing PowerPC devices from earlier work on MPC750, MPC744x devices, and other devices over the last several years [1,17,99,100,101]. The earlier testing is useful for insight into test methods on these devices. However, the earlier devices are not SOCs, and very little exists on testing of bridge chips—which would be relevant to the peripherals available in the modern SOCs.

6.4.2.3 P2020 Efforts

P2020 SOCs have been tested for proton and heavy ion SEE. The P2020 is a dual core e500 SOC with direct peripheral connections such as universal serial bus (USB) and secure digital (SD) card connections, and high speed peripheral support for networking and file storage, such as Peripheral Component Interconnect Express (PCIe) and Gigabit Ethernet (GBE). The block diagram of the P5020 is shown in Figure 6-12. The P2020 is made at the 45-nm feature size and is a very important crossover device for this guideline because it bridges a lot of the earlier PowerPC™ testing with modern SOCs.

In this section we will present some of the test efforts on the e500 cores and peripheral devices in the P2020, focusing on the proton and heavy ion test results. Testing targeted for development of this guideline includes standard microprocessor testing of caches and registers, in addition to dual core operations and memory management testing.

The P2020s are also covered in this results section because of efforts related to device preparation, test algorithm development, and test operations. The findings are useful background for other parts of this guideline.

QorIQ P2020/10 Communication Processors

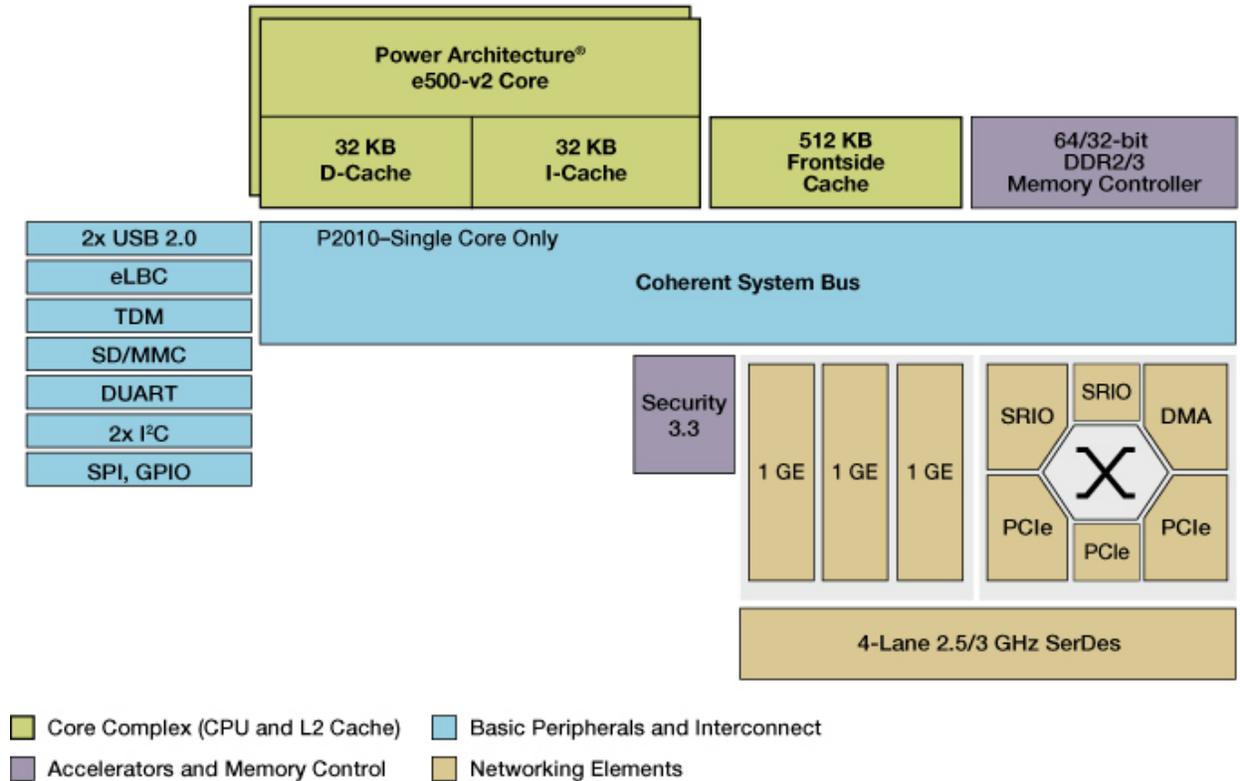


Figure 6-12. P2020 dual e500 core SOC block diagram [8]. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.

6.4.2.4 P5020 Efforts

The P5020 is a dual core e5500 64-bit device that provides a larger set of SOC SEE exploration opportunities than the P2020. The block diagram of the P5020 is shown in Figure 6-13. The P5020 has significantly more high speed IO, at higher speed than the P2020, including a 10-gigabit Ethernet port. It also directly supports Serial Advance Technology Attachment (SATA). By comparison with the P2020, however, it is clear that much of the resources are very similar, and specialized test code for many of the peripherals will probably transfer between the two processors with minimal difficulty.

Limited SEE data from the P5020 are presented in this section. The device was also useful in exploring device preparation issues.

QorIQ P5020 Communication Processor

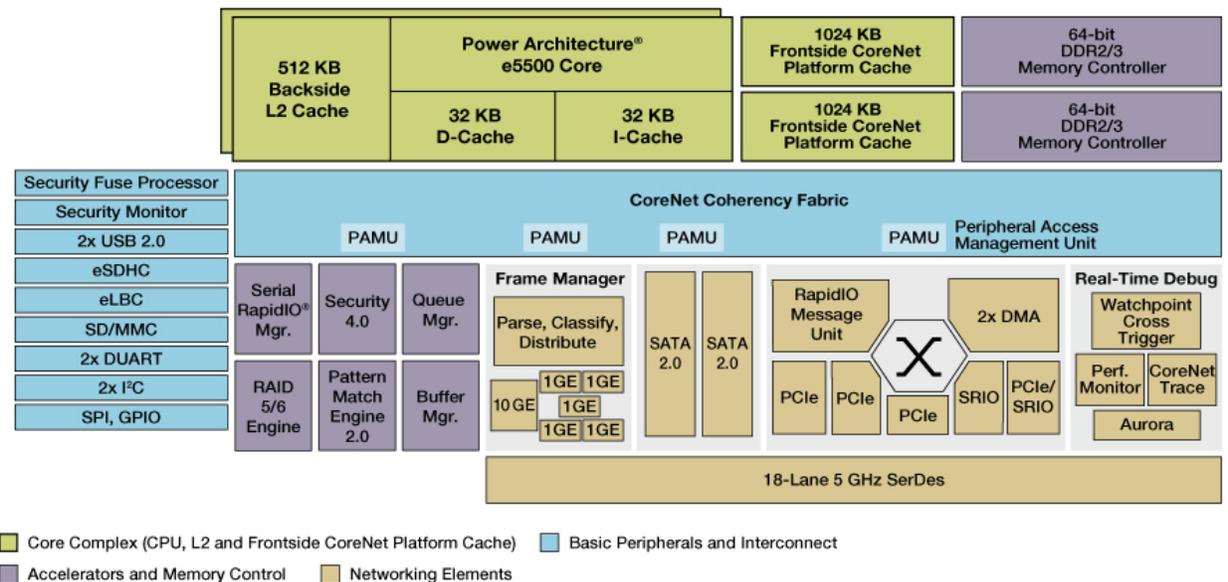


Figure 6-13. Block diagram of the P5020 SOC showing significantly more complexity and more peripherals for examination as compared to the P2020 [11]. Reprinted with permission of NXP Semiconductors, BV. © NXP Semiconductors, BV. 2017.

6.4.3 Commercial Issues

Freescale SOC's are clearly not aerospace devices. As such they do not compromise speed or power for the standard aerospace concerns of radiation performance and extreme power and heat restrictions. These operating conditions mean that the examined devices provide a good source of high speed and performance devices that are also very sensitive to SEE.

Since the devices are not RHBD, some issues must be addressed in order to test them. First, is the question of SEL. 90 nm and earlier microprocessors from Freescale were built on silicon on insulator (SOI) and are immune to SEL. It also appears that the 45 nm devices are not sensitive to SEL (though this has only been verified to an LET of 14 MeV-cm²/mg). There is also the question of TID; however these devices perform very well for TID, and we did not expect any effects until beyond 100 krad(Si). Although SEL and TID performance is good, SEE performance in general does require special consideration.

One issue of the small feature size and lack of RHBD is that clusters of upsets are expected. For this reason it is important to capture information on all observed upsets so that correlation can be ascertained.

The P5020 also required the most consideration regarding heat issues of all devices in this guideline. The P5020 easily goes from room temperature to its maximum operating temperature within 30 seconds if it is not operated with a heat sink. Further, it is encased in a copper heat spreader that is thicker than the P2020's and can only be partially removed in order to allow the rest of the heat spreader to perform properly and regulate the device temperature.

6.4.4 Test Goals

Testing was performed with ions with LET from 1.5 to 14 MeV-cm²/mg. Multiple software algorithms were used to sensitize the DUTs, and the CodeWarrior™ tool was used to provide debugging support to running devices.

The particular goals of the testing of P2020 and P5020 devices in support of this guideline are presented below.

6.4.4.1 Basic Sensitivity Testing

Both the P2020 and P5020 were tested for register and cache sensitivity. These were accomplished by loading a known pattern into the target region, allowing the device to “dwell” and collect upsets for a period of time, then interrogate the target region and note bits that flipped.

6.4.4.2 Peripheral Testing

The P2020 was tested for a high rate of memory access both to the cache and to the external memory.

6.4.4.3 Core Communication Testing

A major part of the benefit of modern SOC is the multicore element. The P2020 test software was designed to allow both cores to operate simultaneously in order to explore common mode SEE.

6.4.5 Hardware Setup

The Freescale multi-core processors present unique hardware setup issues relative to the other example devices explored in this guideline. In this section we present these challenges and the methods used to overcome them for sample testing of Freescale devices.

6.4.5.1 Test Setup

The P2020 and P5020 test setups are similar. A commercial board was used for testing, in the case of the P2020 we used the P2020RDB, while for the P5020 we used the P5020DS. The general test setup is seen in Figure 6-14. Freescale’s P2020 and P5020 processors are in the QorIQ family and built on the 45 nm SOI, which is not sensitive to SEL, thus we used the power supply provided by each test board to operate the board.

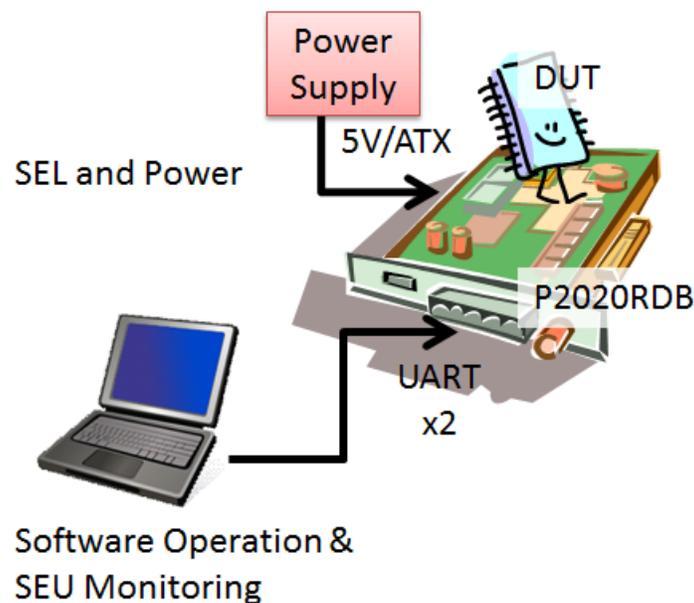


Figure 6-14. The general test setup for operation of the P2020 and P5020 for SEE testing.

DUT communications were conducted using the UART ports, with one port assigned to each core. We also used the Freescale CodeWarrior tools connected through the USB TAP connected to the JTAG port of the processor. CodeWarrior was briefly discussed earlier as an example of a manufacturer debugging tool. It enabled us to explore the P2020 and P5020 processors after crashes.

6.4.5.2 Heat Spreaders

For heavy ion irradiation in terrestrial facilities, access to the sensitive region of the device is necessary. The P2020 and P5020 are both problematic in this regard. They both have a ball grid array (BGA) on the bottom, creating a non-uniform medium the beam must cross. And they both have a copper heat spreader on the top, which is too thick for all but the most energetic beams from the standard test facilities (e.g., TAMU). The P2020 was modified to remove the heat spreader over the entire die, as seen in Figure 6-15.

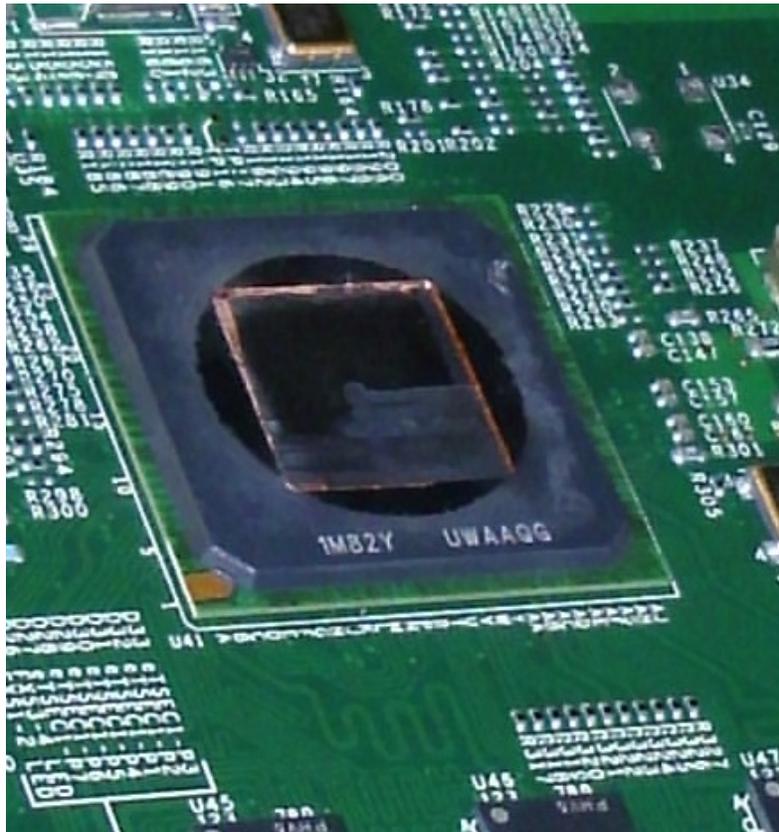


Figure 6-15. A P2020 with the heat spreader opened over the die. Photo used with JPL permission.

The P5020 was more complex. As discussed in the next section, the P5020 thermal management was significantly more important than for the P2020. As a result, the heat spreader could only be opened directly over target regions of the chip. An example of a prepared P5020 is shown in Figure 6-16.

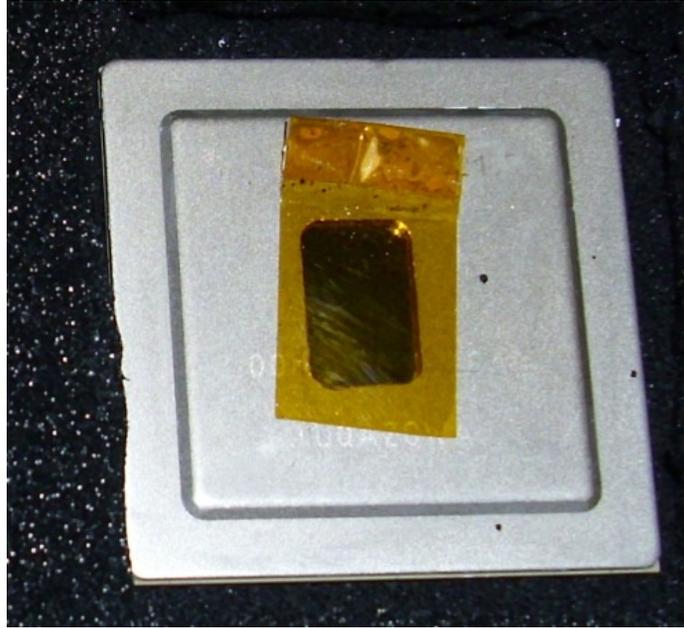


Figure 6-16. A P5020 with only a portion of the heat spreader removed over the die. This photo used with permission and provided by Jet Propulsion Laboratories.

6.4.5.3 Thermal Issues

Two solutions for thermal issues were used for Freescale testing. The less drastic version was applied to the P2020. Because of the physical construction of the P2020, heat transfer still occurs in the filler material between the die and the heat spreader. We used a dry nitrogen line to blow cool air onto the P2020 during testing. This was sufficient, but we also did not operate multiple IO ports of the P2020 during testing. The modified P2020 and the P2020RDB test board with dry nitrogen line are shown in Figure 6-17.

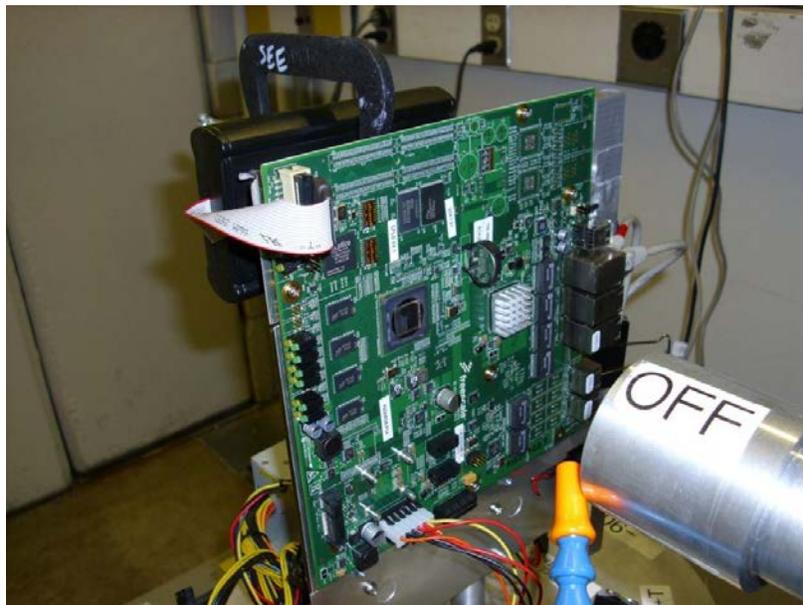


Figure 6-17. A P2020RDB test board at TAMU is shown. Note that when it is moved closer to the beam opening, the dry nitrogen line (blue/orange) blows directly on the opening in the heat spreader. This photo used with permission and provided by Jet Propulsion Laboratories

6.4.5.4 Test Board Cost

Two types of test boards have been examined for testing of Freescale processors. They are exemplified by the test systems used for SEE testing of the P2020 and the P5020. The P2020 was tested using Freescale P2020RDB (Reference Design Boards), which cost less than \$700. The P5020, on the other hand, was tested with P5020DS (Demonstration Systems) which cost approximately \$4000. The P5020 demonstration system is shown in Figure 6-18.

Luckily the P5020DS turns out to use a socket to hold the P5020 processor. And since we were able to purchase loose P5020 processors for about \$600, the per DUT cost turns out to be similar to the P2020. However this is not generally the case.

The significant difference in cost between test systems suggests that an alternative approach should be used in obtaining test boards for SOC testing. We highly recommend shopping around for alternate, inexpensive test boards. There are a few reasons for this. Generally speaking the test goals for evaluating the SEE sensitivity of an SOC do not require a high performance computer system for testing. Thus it is not necessary to have a computer with multiple PCI-express ports, nor is it necessary to have Serial Advance Technology Attachment (SATA) drives installed or a lot of memory (especially considering the heat generated by the chip appears to be closely related to the amount of peripherals that run simultaneously).

At least for initial testing it is highly recommended to use low-cost boards. One alternative which has not been directly explored here but likely would be very useful for controlling cost would be to test a COM Express board such as the COMX-P5020 system from Emerson [102]. Boards may be found in the \$200-400 range, cutting the cost per DUT.



Figure 6-18. P5020 demonstration system. It can easily be seen that this is a general purpose circuit card enabling many different end uses. Image used with permission from Jet Propulsion Laboratory.

6.4.6 Data Presentation

Collected test data is presented here. The P2020 and P5020 data provide examples of heavy ion testing.

6.4.6.1 Effective Sensitivity

For the Freescale devices, the effective sensitivity is set by two elements. The first is the lowest reasonable flux that can be delivered. And the second is the approximate fluence at the point where processor cores crash—the reason for this is that core crashes are often due to double bit errors in the caches, and a double bit error in the L1 cache causes erroneous code execution leading to incorrect counts. For the P2020 the approximate effective sensitivity is $1 \times 10^{-6} \text{ cm}^2/\text{device}$ (more detail can be seen in the core crash section). For the P5020 the sensitivity was not directly analyzed but is expected to be approximately the same (because the sensitivity of both processors is driven by the L1 cache sensitivity).

6.4.6.2 L1 Cache

The L1 cache of the P2020 processor was tested for '0' to '1' and '1' to '0' errors. The results are shown in Figure 6-19. No difference was observed between '0' to '1' and '1' to '0'. No difference was seen between the processor cores. The most interesting element of the L1 testing for the P2020 is that parity protection causes cache lines to be invalidated and reloaded on a parity error, and this behavior cannot be disabled. Hence, events were detected by observing cache content being invalidated before they are written back to main memory. Note that this may cause significant trouble with detecting these events unless the test algorithm is capable of low level device operation to setup custom cache states where invalidation will be detectable.

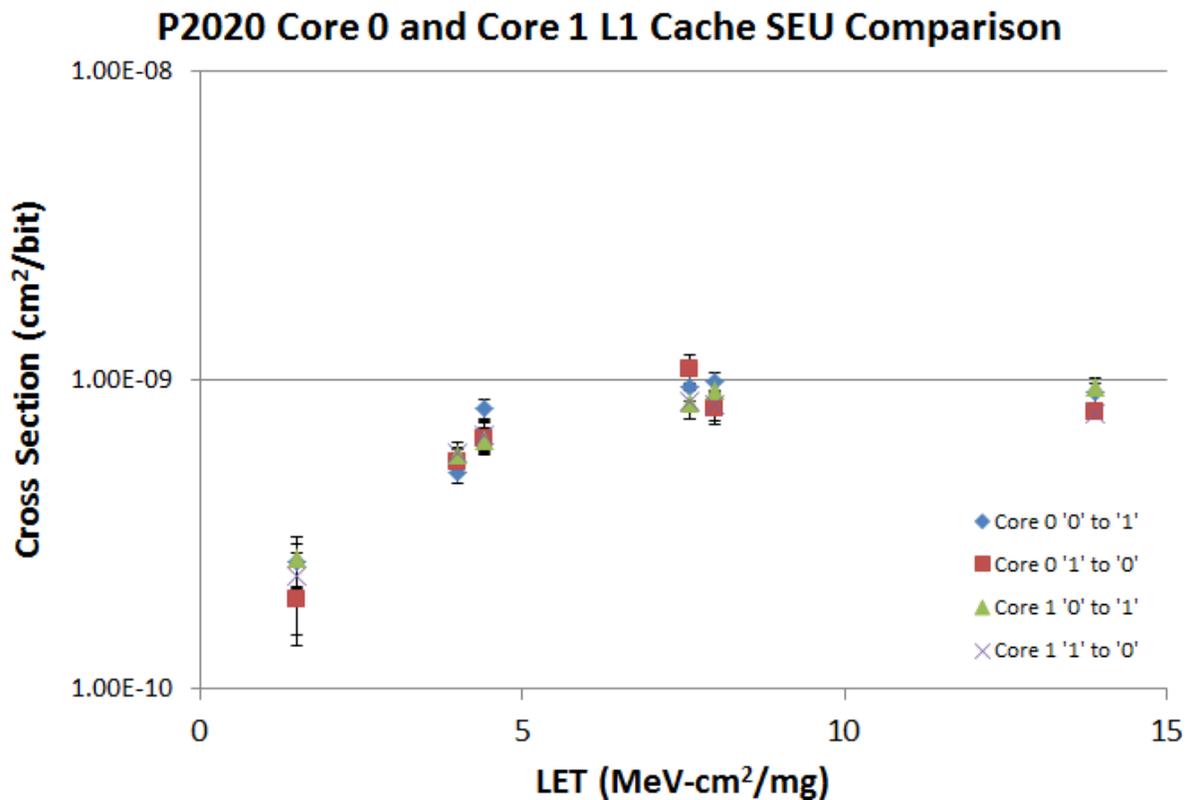


Figure 6-19. P2020 L1 cache results from [103].

6.4.6.3 L2 Cache

The L2 cache of the P2020 can be tested by putting it into cache as RAM mode, this allows us to bypass the EDAC and directly observe upsets. This is a very useful mode to use for testing. The results of this are given in Figure 6-20. Note that the results for the L1 and L2 caches are similar, though the L1 cache has a somewhat higher cross section. The L2 cache in the P2020 is attached to the core complex bus and is shared by the two e500 cores. Thus, it can only be tested by one core at a time.

The P5020 L2 cache was also tested; however, it could not be put into the cache as RAM mode, and the P5020's e5500 cores each have their own L2 cache. For this data, only one of the e5500 cores was tested. The results are shown in Figure 6-21. Note that the results are very similar between the P2020 and P5020 L2 caches.

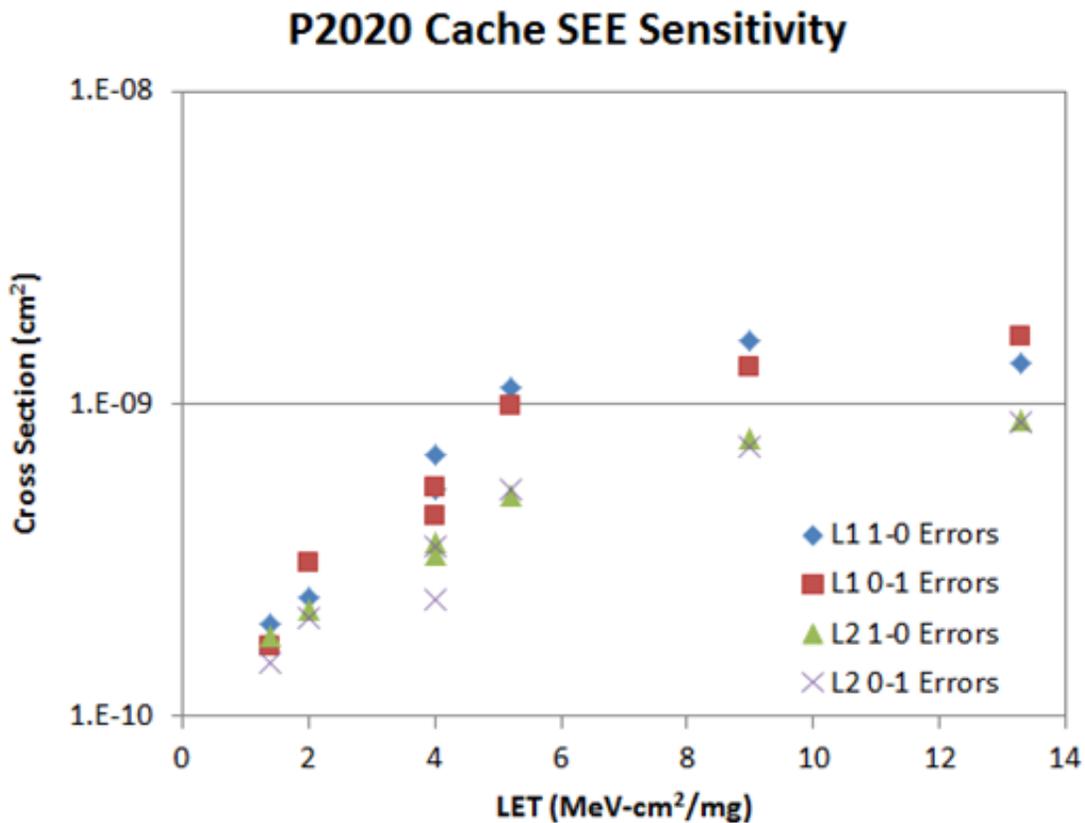


Figure 6-20. Results for L2 testing on the P2020 [19].

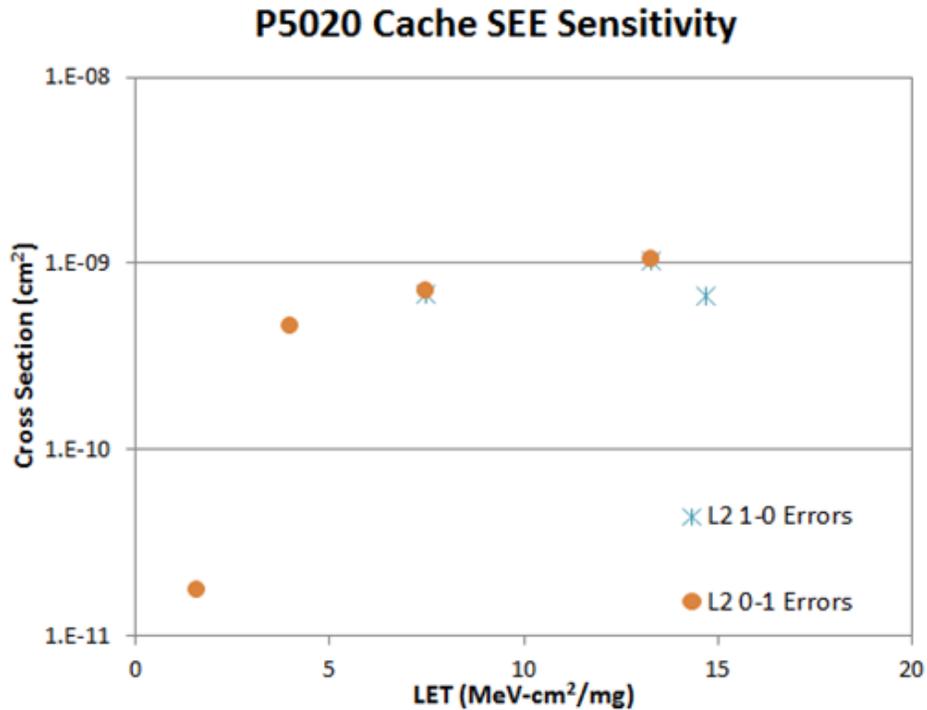


Figure 6-21. P5020 L2 SEE sensitivity, showing similar sensitivity for '0' to '1' and '1' to '0' errors [53].

6.4.6.4 Registers

Not all registers in the Freescale SOCs are used by the test algorithms. The ones that are not used can be passively tested by loading them with a known pattern either before irradiation, or periodically, and then querying their value after irradiation, or periodically after an integration period. The periodic method was used to test 22 general purpose registers (GPRs) on both P2020 cores. The results of the testing are provided in Figure 6-22. Note that no difference was observed core-to-core, but the number of counts is too low to establish statistical significance.

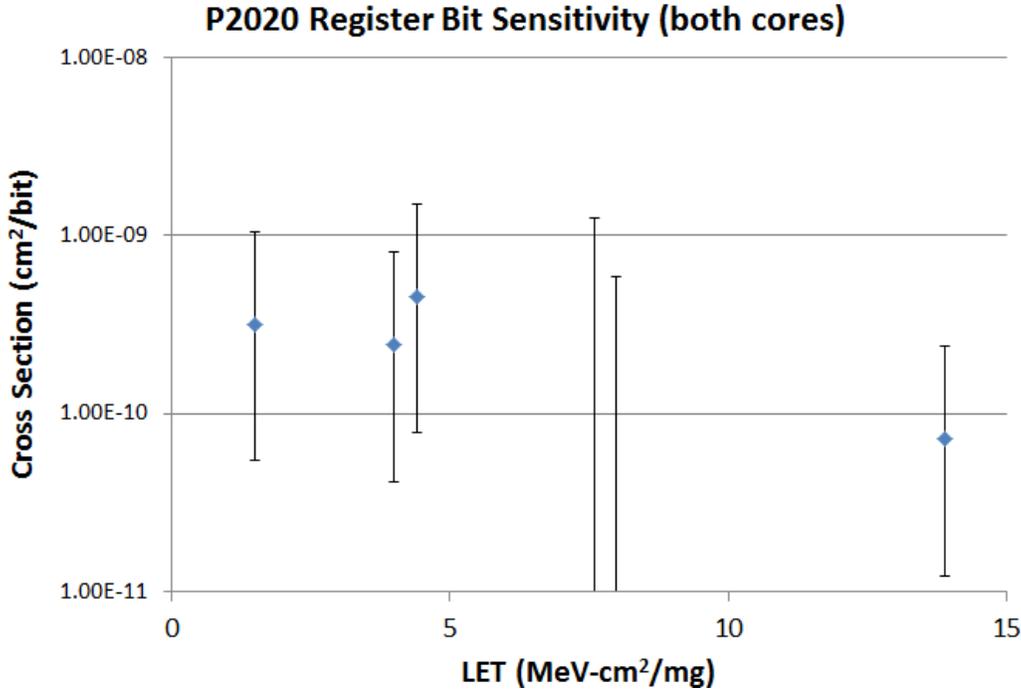


Figure 6-22. The register bit SEE sensitivity for e500 GPRs in the P2020 processor [103]. Testing was performed on both cores.

6.4.6.5 Core Crashes

Processor cores can crash for many reasons. Single event functionality interrupts [SEFIs] and hangs cause crashes. In addition, all other unexpected phenomena cause crashes, since crashing is the absorption state of the individual core state diagram. The test software used, which follows the recommendations in Section 3.6, generally does not crash due to unhandled radiation events that could have been handled. And because of the debugging history at beam facilities, most crashes due to bugs in software that only runs when radiation events occur are minimized. Thus, the remaining crashes are believed to be essentially unavoidable. They mostly fall into two categories: uncorrected exception conditions (the software is not able to recover the state of the processor in every exception circumstance) and alteration of the test software or control variables. (As an example of the latter, we have observed situations where the dwell period of the test code changes permanently—likely the result of a double-bit error in the instruction cache which cannot be detected.)

Results for crashes of cores on the P2020 are shown in Figure 6-23.

6.4.6.6 Write/Read Testing (Memory Controller)

A test algorithm was developed that performs accesses to memory or accesses to the L1 cache of the P2020 based on the configuration setup at the start of the run. Using this algorithm we did not observe any significant sensitivity to either algorithm to the level of sensitivity supported by the testing (approximately $1 \times 10^{-6} \text{cm}^2$).

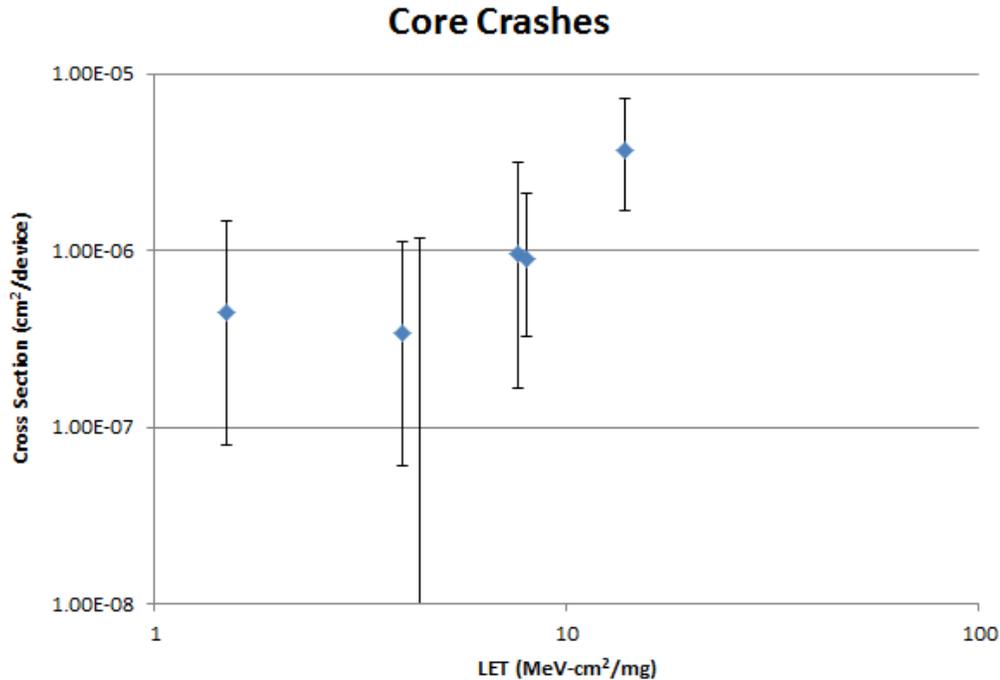


Figure 6-23. Crash sensitivity of the e500 cores in the P2020 processor (includes SEIs and hangs, since both result in loss of operation of the core) [103].

6.4.7 Freescale Conclusion

The Freescale SOCs present a very useful category of example devices for this guideline. They are commercial devices with high processing capability. They include multiple versions spanning single core to eight cores, with varying types of FT. The Freescale SOCs come with specialized high speed IO capability that can be supervised by the on-chip processing cores. These devices are based on the Power™ architecture which has been leveraged for the RAD750 processor and has significant support for aerospace applications, including use in Space Micro's Proton 400k-L SBC [9]. Future implementations of Freescale's e5500 and other 45-nm SOC library elements are expected. Hence, by studying this family of SOCs we are surveying a category of devices that have good traction including existing test methods and implementation, and we are simultaneously looking at devices that will be important in the future.

In this guideline we have specifically studied the P2020 dual core e500 SOC and the P5020 dual core e5500 SOC. Both the P2020 and P5020 cache bits perform in line with earlier PowerPC testing performed on IBM and Freescale/Motorola devices. Because of the small feature size (45 nm) and the low LET_{TH} it is recommended to obtain low energy proton data on these parts, but this was not done for this guideline. Devices continue to have no observed destructive SEE modes. We found that common mode failures do not predominate, and that individual cores can crash independently of each other. In real flight applications the SEE sensitivity of a multicore SOC will depend on the FT capabilities of the hypervisor and its software implementation, and the SEE sensitivity of the individual cores will depend on the exact software structure. Initial testing of hardware interfaces suggest they are probably robust to SEE, but specific testing of application IO is recommended for any interface that has not been tested previously. Core to core communications should be tested, and the Freescale SOCs provide a good example where communication is primarily achieved through memory coherence.

The P2020 and P5020 devices, and Freescale PowerPC devices in general, provide high functioning commercial devices that are inherently inexpensive candidates for developing this test guideline's

approaches. It is expected that future devices and technology trends can be surveyed for SOC test method development by keeping up to date with test methods appropriate to these devices.

6.5 Other SOC Test Efforts

Aside from the testing done in direct support of this guideline, testing has been performed on SOCs and microprocessors. We present some details of these test efforts here.

6.5.1 PowerQUICC III with Whetstone

Quinn et al, [104], report on testing the Freescale MPC8548E PowerQUICC III with a Whetstone benchmark algorithm [105], which is based on the microprocessor test approach from Irom, 2008 [1]. Two test approaches were used. First, the registers and L1 cache bits were initialized and monitored for upsets utilizing a test setup that does not run software on the DUT, but instead performs testing through the JTAG scan chain. Second, the Whetstone benchmark was run continuously, and output was checked against expected results.

Using JTAG to directly access register and cache bits is an excellent alternative to the test approach discussed thus far in this test results section. Such a method was suggested for Maestro but was shown to be too slow to be practical. But for the Freescale SOCs, this method appears to be viable.

The use of Whetstone is an example of a synthetic, flight-like algorithm because it usually is compiled and run in a standard execution environment. This is different from the test approach suggested in this guideline for testing device structures, but it does provide general device response, which is also recommended in this guideline. The particular value of the Whetstone algorithm for mapping to user applications may require further detailed study, but can also be used along with register and cache information to provide bounding SEE rates for a device in an application.

6.5.2 STMicroelectronics 8051-Based SOC

An 8051-based SOC from STMicroelectronics was examined for test methods and SEE response by Rech, 2010 [42]. In that work both test algorithms and programmable built-in self test (pBIST) are used to examine both static and dynamic SEE sensitivity.

Testing showed the viability of test circuits, such as JTAG and the pBIST capability for determining static SEE sensitivity. Dynamic testing was carried out with custom-built algorithms for testing target elements of the SOC under radiation. The algorithms include repeated calculations and verification of the values that result from the calculations, and the performance of code loops versus unrolled code (where loops repeated n times are replaced with n instruction blocks repeated—a mechanism sometimes used to increase reliability or testability in aerospace applications—intended to sensitize a different part of the processor's memory handling).

A key part of the work in Rech [42] was the examination of propagation of errors from static elements to dynamic and system-wide results. In particular they showed that static upsets need not propagate to the system level and system level SEE need not have started with static elements (i.e., SET can be a source of upsets).

As a case study, Rech [42] is a very valuable source for radiation test development, actual testing, and methods that can be used to improve the SEE behavior of a device. The testing is in line with many of the recommendations of this guideline including the use of low cost test boards, testing of fundamental elements of the circuit design, performance of dynamic testing, exploring the relationship between SEE in storage elements and system-level events, and possible methods for achieving increased RHBD or FT in custom versions of the IC.

6.5.3 Using DfT Circuits

Design for testability (DfT) circuits provide another useful way to use hardware tools to assist in SEE testing. This method has been developed in Suh et al., 2012 [106] and can provide insights on other hardware methods for testing.

6.5.4 Other General Microprocessor Testing

The microprocessor ground-based test guideline [1] includes a critical review of SEE testing of microprocessors. In this guideline key topics are feature size and frequency dependence. These are reexamined here with an eye on identifying items of interest to this guideline. The analysis in Irom [1] is drawn from a wide array of microprocessor SEE publications, some of which are reviewed here.

6.5.4.1 Hangs

“Hang” is a term used to describe the state that a microprocessor gets in when it is not executing code in a useful way for external users (i.e., there could be correct internal operations, but no external indication exists that the device is still working properly). We have referred to this as a core crash throughout this document. Hangs were examined in Irom [1] by comparing JPL’s earlier test methods (which are largely still followed and described in this guideline as well as in Irom [1]). Figure 6-24 shows the comparison of PowerPC test code and Intel results when using Windows NT as the platform for test operations. Note that the data presented in Section 6.4.6 for heavy ions on P2020 DUTs, when translated to protons will also be approximately in the $1 \times 10^{-11} \text{cm}^2$ for a core crash (or hang). Thus, we have not observed a significant increase in the achievable crash or hang rate for PowerPC devices, even when the devices are complex SOCs. It is believed that low-level code used to sensitize commercial DUTs for SEE will generally achieve this level of crash rate.

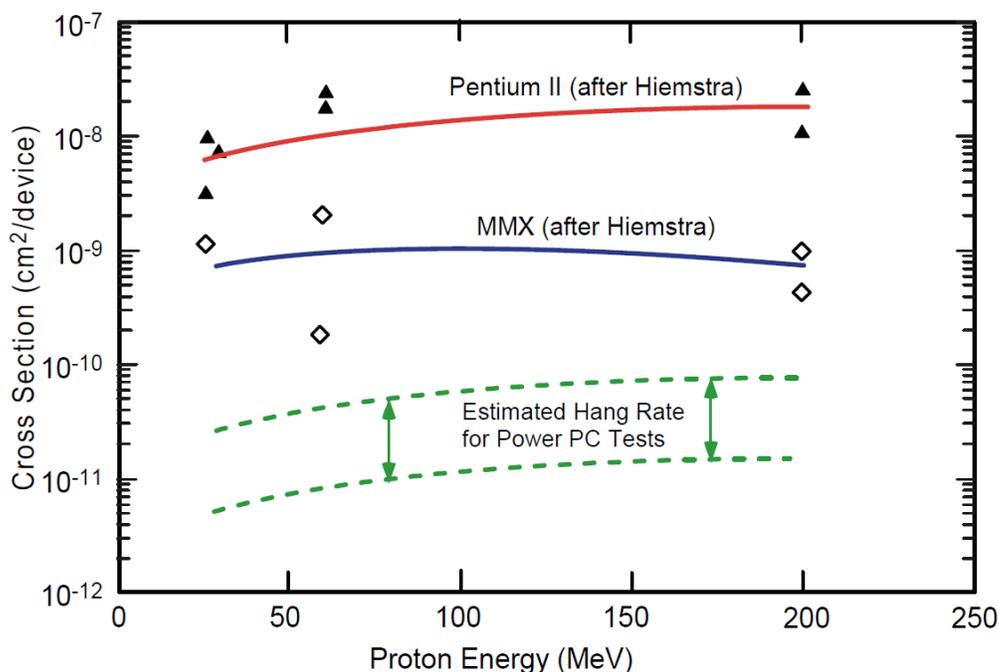


Figure 6-24. Hang results on PowerPC and Intel tests for Power PC, MMX, and Pentium II processors. Note that testing performed on Intel devices used the Windows NT operating system, which is very complex and requires more of the device to function properly than the JPL PowerPC test software [1], with Hiemstra data from [39]. Image used with permission.

6.5.4.2 Feature Size

The response of the data cache bits of PowerPC processors with five different feature sizes are shown in Figure 6-25 and Figure 6-26. The reduction in cross section with feature size is partially explained by the smaller size of the cache bits and therefore the physical size of the cache. However, as can be seen with the smaller feature sizes in Figure 6-25, and the results from Section 6.4.6, the saturated cross section for processor cache bits have remained close to $1 \times 10^{-9} \text{ cm}^2/\text{bit}$ from the 180 to 45 nm nodes.

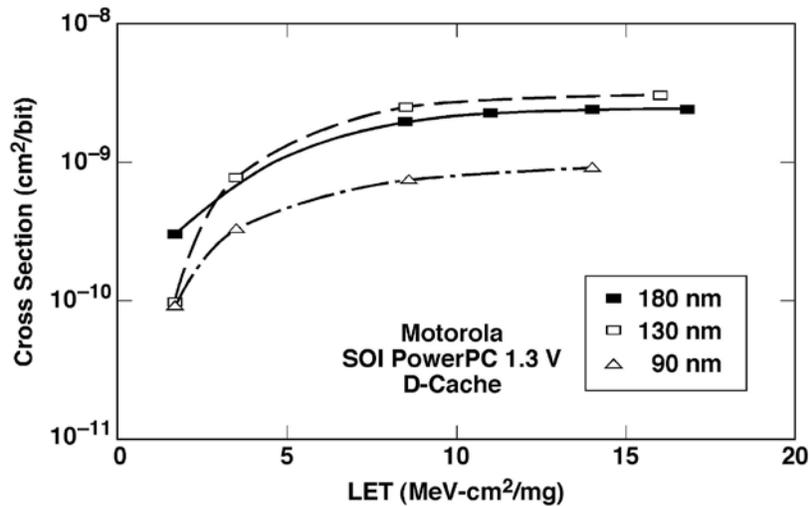


Figure 6-25. Cache bit SEE sensitivity for three PowerPC processors with different feature sizes [101].

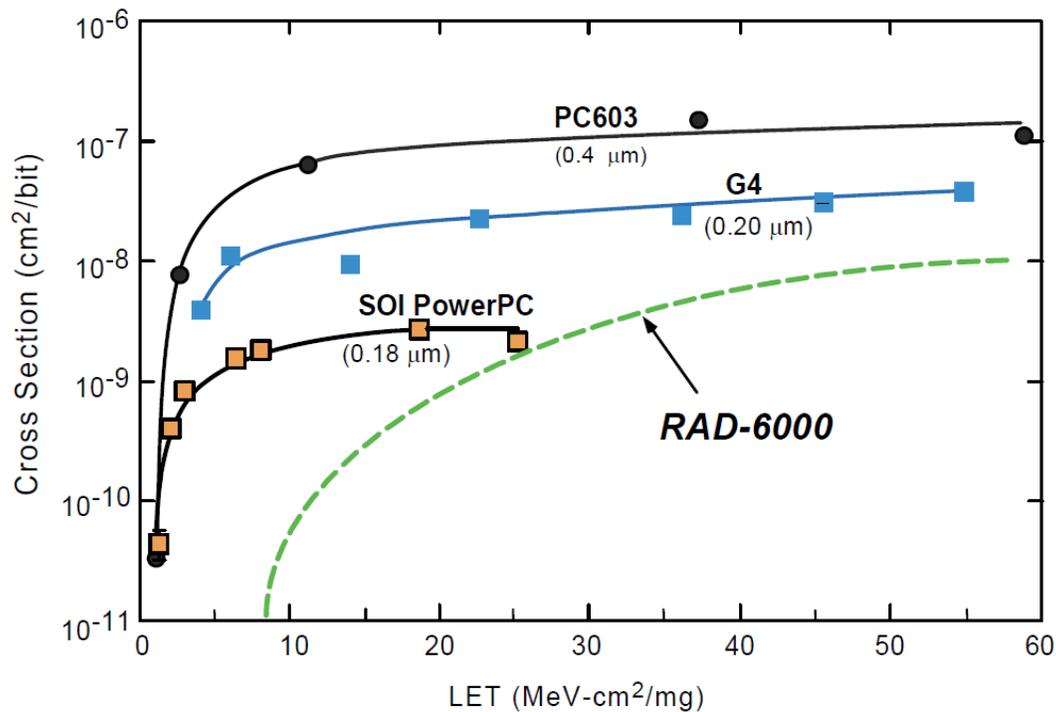


Figure 6-26. Cache results for more PowerPC processors [1]. Note that RAD6000 processors are 0.50 μm or larger.

Although the potential for changes in the saturated cross section of these devices exists, they are expected to remain around the $1 \times 10^{-9} \text{ cm}^2$ from a purely physical constraint perspective.

6.5.4.3 Frequency

Another key concern for SOCs and microprocessors is the frequency dependence of SEE response. Although this is known to be a critical concern for RHBD circuits due to intentionally structuring them to dampen SETs while making clock edges more important, there has not been a significant impact on commercial devices due to operating at different frequencies.

Frequency impact on PowerPC devices was reviewed in [1], and the general results are similar to those found in [22]. These results are shown in Figure 6-27.

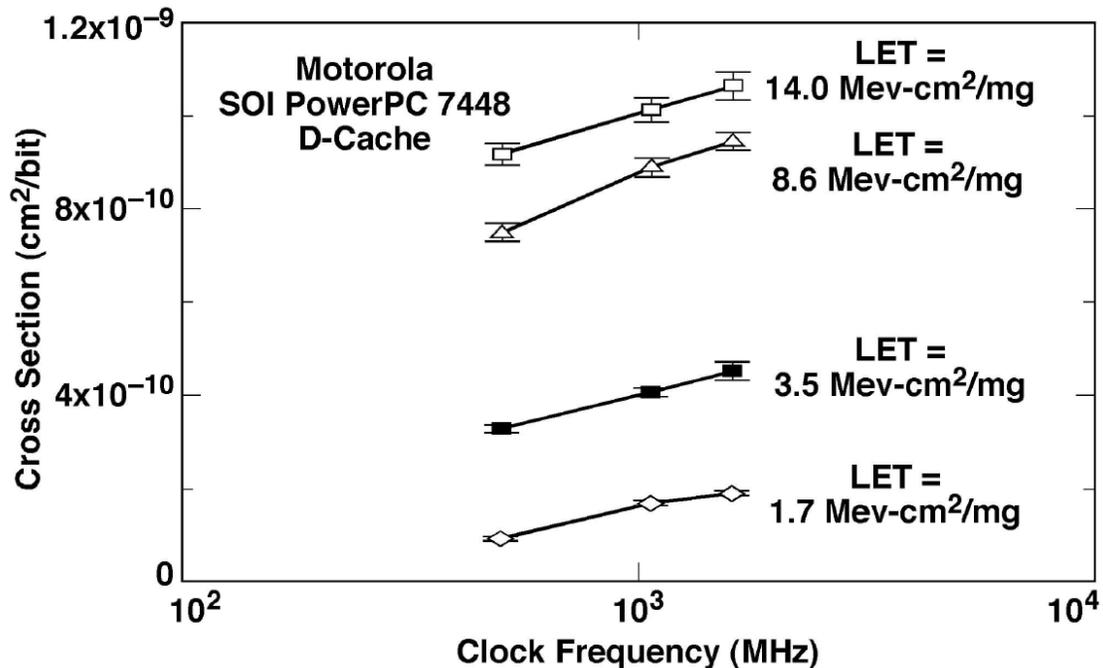


Figure 6-27. Comparison of SEU cross section for the PowerPC7448 operated at 500, 1066, and 1600 MHz, some difference is observed with frequency, but it is limited to about 30% [1].

Another potential implication of frequency was observed in Guertin and Irom, 2010 [17]. During testing of the IBM 750FX the device frequency was set at 400, 600, and 667 MHz. It was observed that parity error identification at 400 MHz resulted in machine check exceptions which could not be recovered, while at 600 and 667 MHz the test software was able to handle these events, count them, and then recover the test code. This suggests that specific use cases may be very important for frequency testing.

6.5.5 Intel/AMD Testing

Intel and AMD processors have been examined previously [39,40]. The data collected were developed under either Windows NT [39], or the Pharlap real time operating system (RTOS). The DUT and test board setup were similar to the P5020 testing except that the modern device heat spreaders are more problematic than the Pentium 3 and AMD K7, but all devices require extreme caution in operation as modified devices or when devices are exposed to allow heavy ions to hit the IC surface.

Cache test results from Howard et al., 2001 [40] show a saturated cross section and overall behavior very similar to the Freescale devices examined earlier in this system. These are shown in Figure 6-28.

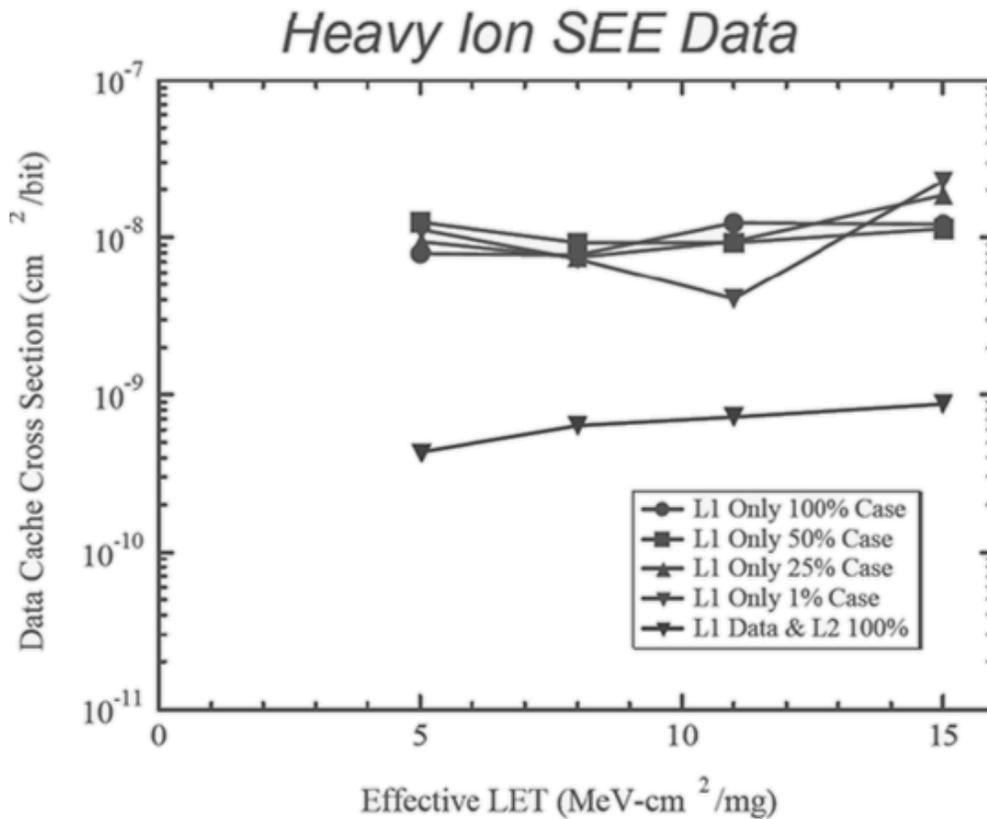


Figure 6-28. Cross section for upsets of data cache bits in Intel Pentium III microprocessors. Note that the lower line corresponds to the L1 Data & L2 100% case [40].

One item of importance here is that the test software was not robust and would not form a solid basis for the development of test software. The crash cross section is shown in Figure 6-29 from Howard et al., 2001 [40]. Here it can be seen that SEFIs (events where a processor core no longer functions correctly, equivalent to a hang or crash, which requires resetting the DUT to continue) have sensitivity on the order of 1×10^{-4} cm². These SEFIs reduced the ability to detect other types of upsets. The overall cross section for these events is approximately 1×10^{-4} cm². Although the feature size is larger for these devices than for the modern Freescale devices, the indication is that the Windows NT operating system obscures errors due to SEFIs.

6.5.6 Testing of FPGAs

FPGAs can provide useful examples for testing of SOCs. FPGAs can easily be used to implement the digital components of a computer system, essentially duplicating the functional behavior of an SOC. Two key areas of interest exist regarding development of test data for FPGAs. The first is testing of embedded hard-core or soft-core microprocessors. The second is testing of peripheral devices. Although the information presented here is limited, FPGA SEE testing is likely to provide complementary data in the future and should be explored by test groups working on future SOC efforts.

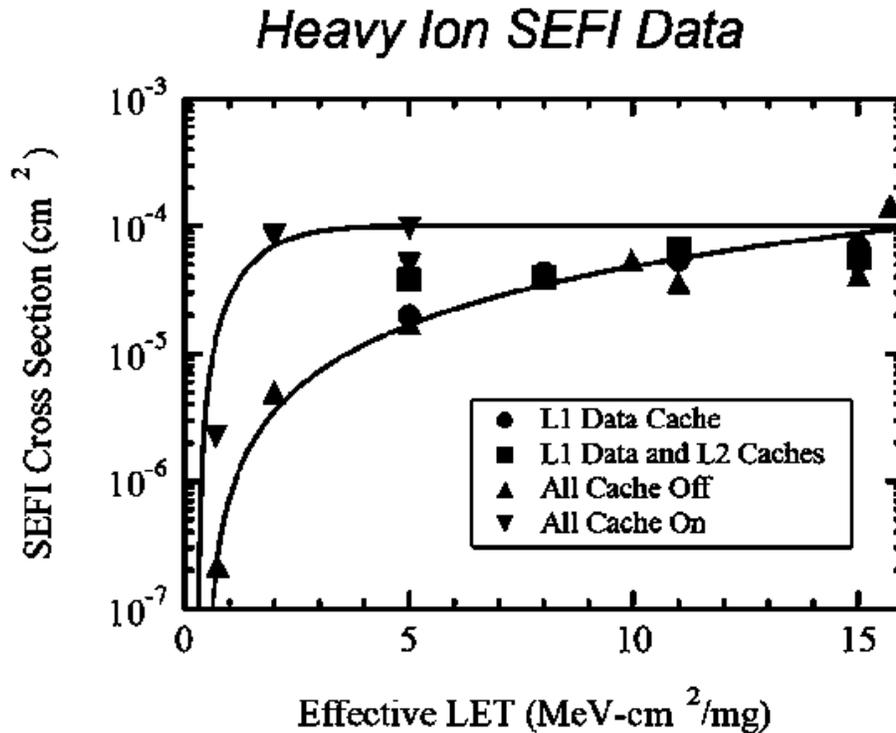


Figure 6-29. Cross section for SEFIs from various test algorithms [40].

6.5.6.1 FPGA Microprocessors

FPGAs sometimes have hard-core embedded microprocessors, and can always use microprocessor IP implemented in the FPGA fabric to provide embedded processing support to a design. These PowerPC processors embedded in the commercial Virtex-II Pro were tested for heavy ion SEE [107]. During this testing SEFIs dominated the data taking, and no good SEE response for the actual processing core could be collected.

The Gaisler Leon3 processor core was studied in the Xilinx Virtex 5-QV by Learn [108]. Many of the details of the effort involve technical details about improving the SEE response of the FPGA that are irrelevant here. However they were able to improve the FPGA performance to the point where the observations likely come from the operation of the processor core. For reasons related to the test setup, SEEs were recorded when the test setup issued a reset to recover operation of the test software in the processor core (a mechanism that might be applicable to an SOC). The results are shown in Figure 6-30, where the various arrangements of FPGA fault handling are compared. Note that for the unhardened Leon3 implementation, the cross section saturates around $5 \times 10^{-5} \text{ cm}^2$.

It is expected that more SEE data on embedded microprocessors will be developed as FPGAs are used more readily in space applications.

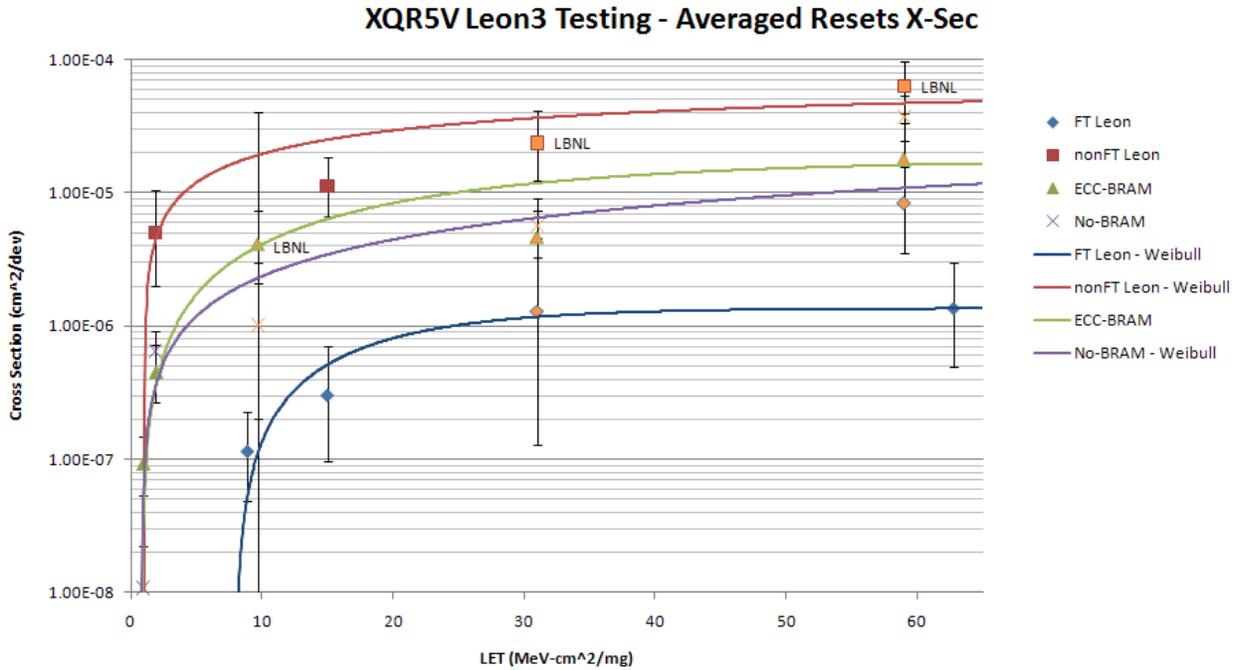


Figure 6-30. The SEE response, device cross section vs. LET, of Leon3 (with FT, and with various amounts of SEE protection) showing that the unhardened setup saturates at about $5 \times 10^{-5} \text{ cm}^2$. [108]

6.5.6.2 Peripheral Testing

A good example of peripheral testing from FPGAs is the case of the multi-gigabit transceivers (MGTs) developed by Monreal et al [109]. These devices are contained in the Xilinx Virtex 5-QV, and considerable SEE data on this part can be found in Swift et al. [110]. Monreal et al identified six different ways that MGTs can have SEEs, including bit errors, and re-sync events/loss of link both on transmitters and receivers. The cross section vs. LET for bit errors on an MGT channel is shown in Figure 6-31.

6.5.7 Simulating SEE to Obtain System Error Rates

Applying test results on complex devices from the test configuration to arbitrary user configurations is necessary in order to estimate flight performance from test results. The fundamental problem is that the most straightforward way to establish SEE rates involves the approach where basic structures are tested and rates are calculated by applying SEE sensitivity to all of the basic structures equally (this was covered in Section 2). This method is known to overestimate SEE sensitivity in complex devices because not all portions of the device are important to SEE sensitivity on any given clock cycle. Although it is outside our scope to discuss how to determine system SEE rates, we do cite two references here [111,112] that can be useful for determining methods that may be relevant to the reader concerning simulation of errors in microprocessors.

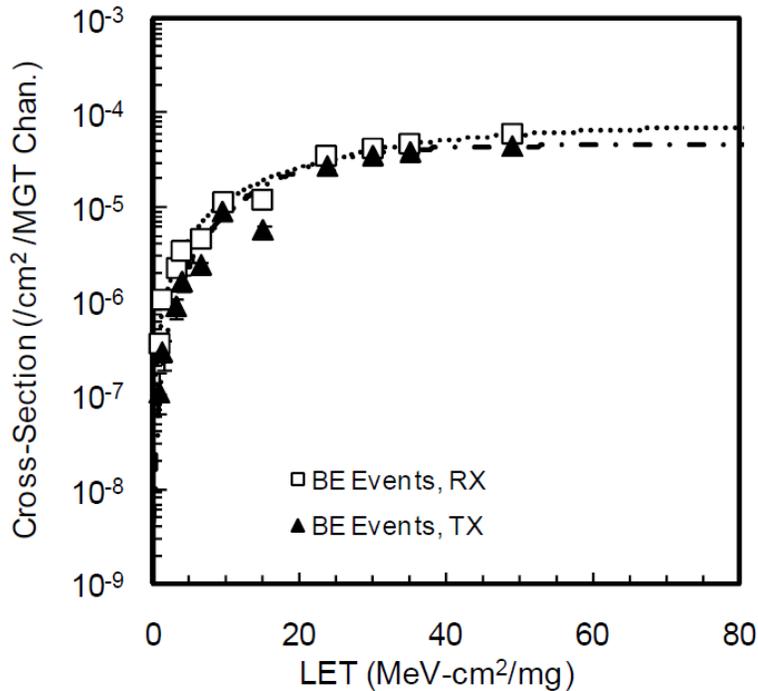


Figure 6-31. Cross section for bit errors in an MGT channel on the Xilinx Virtex 5-QV [109].

6.6 Recommendations

1. Bring a programmer to the remote test facility.
2. Be prepared to collect data sets with anomalies entangled, but only do so if the anomalies are directly caused by SEE.
3. Anomalies should be treated as bugs in the test code or system until they can be reproduced through isolated beam behavior (i.e., multiple types of test code or specialized code that enables direct examination of the error).
4. Automate operations if possible because procedures for complex devices may be too hard to memorize and carry out during testing.
5. Be careful to monitor for thermal issues—consider powering down devices between runs (and leverage automation to reduce on time).
6. Significant lowering of cross section below 180 nm on commercial devices is not expected based on existing test data (test engineers should test devices below this level and not assume that per-bit cross sections reduce with feature size).
7. Frequency dependence of SEE is not expected to be significant on commercial devices but may be very significant on RHBD devices.
8. Complex operating systems make results difficult to disentangle. They will lower the effective sensitivity and obscure SEEs.
9. Beware of potential errata under use conditions. Many of the conditions caused by radiation during SEE testing may have had limited manufacturer testability and may therefore not operate as intended at the hardware level.

7.0 CONCLUSION

This guideline has presented the case for the importance of SEE testing of SOCs. The normal approach to radiation characterization of devices was discussed and specialized to the case of SOCs that are difficult to expose and operate. Details of SEE testing, with emphasis on how they impact efforts to perform testing of SOCs, have been provided. And extensive results have been presented to provide a flavor for all the different types of testing and test development that may be involved in performing SEE testing of these devices.

SOCs of interest here are those where significant amounts of interconnect logic and other digital resources for a computer are put together in a single IC to simplify construction and reliability while improving performance. These devices are not complete SOCs because they will require peripherals, power supplies, and some other circuits that must be provided by their host board. Nonetheless, the commercial market has clearly embraced SOCs including those with multiple memory interfaces, multiple processing cores, and mixed portions of various support circuitries. As space manufacturers move forward, they will embrace the commercial SOC paradigm for several reasons, many of which mimic those that have driven the commercial manufacturers in this direction. As such, it is important to have methods for SEE characterization of these devices, especially as some devices like the P2020 have already made their way into space computers such as the Proton 400k-L.

We identified seven major points for dividing up the focus of this guideline. Each of these points is its own research topic, and details relevant to each point can be found throughout the guideline. The major points are collaboration, peripheral approach, FT, RHBD, multicore, general test methods, and sample testing. The selection of sample devices, test setup, algorithms, and test data reviewed in this guideline cover each of these points through multiple examples. To a large extent all of the key elements have been covered well except for two key areas. The first is the peripheral approach, which involves testing of high speed and complex IO devices where the best methods are not clear and many devices exist—making it difficult, costly, and of limited value to explore the entire space. The second is that very little end user collaboration has been explored. This has traditionally been a difficult interface to cross, and we believe that the combination of simulation tools and radiation testing of actual user software will provide a good means to get this collaboration going.

Many of the findings have been collected into recommendations that can be found at the end of each major section. These recommendations cover everything from how to sensitize the devices with test software, to examples of how to decapsulate parts but still maintain functionality. The reader is encouraged to look specifically at these lists. The sections that the recommendations are in generally include detailed arguments from which the recommendations are derived.

This guideline has provided significant detail regarding methods for testing SOCs. Test examples highlighted the different problems that various types of devices present. Examples include parts that cannot be tested at angle, parts that can only be tested with a subsection of the device exposed at a time, minimizing the expense of test parts, and focusing beam requirements to get the best data given the restrictions due to the parts. These are just some examples of the problems discussed here.

This guideline is intended to help readers establish viable SEE testing protocols on modern SOCs. The microprocessor market has always been fast-moving. But right now, in particular, there are major changes in microprocessor and associated SOC development. These changes include architectural elements from microcode to number of cores. They also include combinations of significant heterogeneous execution units including significant parallel computing resources from on-chip GPUs. Each of these, along with recent and future developments in the seven main research thrusts, is relevant to the future applicability of this document.

Readers are encouraged to forward any questions or other feedback to the author at steven.m.guertin@jpl.nasa.gov.

8.0 REFERENCES

- [1] F. Irom, *Guideline for Ground Radiation Testing of Microprocessors in the Space Radiation Environment*, JPL Publication 8-13, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, April 2008.
- [2] “Mars Science Laboratory,” web page, mars.jpl.nasa.gov/msl, NASA/JPL, (accessed 1/2013)
- [3] “RAD750@ radiation-hardened PowerPC microprocessor,” BAE RAD750 Datasheet, July 2, 2008.
- [4] B. Bornstein, T. Estlin, B. Clement, and P. Springer, “Using a multicore processor for rover autonomous science,” *Aerospace Conference*, 2011 IEEE, pp.1–9, March 5–12, 2011.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5747454&tag=1
- [5] *Atmel Aerospace Rad-Hard Integrated Circuits*, marketing brochure from Atmel, 2012.
http://www.atmel.com/Images/AERO-40151-Integrated%20Circuits-Space%20Rad-Hard_US-E-0912_LR.pdf. (accessed 10/31/13)
- [6] “UT699 32-bit Fault-Tolerant SPARC™ V8/LEON 3FT Processor,” Aeroflex product datasheet, 2012.
<http://ams.aeroflex.com/pagesproduct/datasheets/leon/ut699leon3ftdatasheet.pdf>
- [7] M. Malone, “OPERA RHBD Multi-core,” presented at Military and Aerospace Programmable Logic Devices Workshop,” 2009.
https://nepp.nasa.gov/mapld_2009/talks/083109_Monday/03_Malone_Michael_mapld09_pres_1.pdf
- [8] “Freescale P2020,” Freescale Semiconductor website (Freescale became NXP Semiconductors N.V.).
<http://www.nxp.com/products/microcontrollers-and-processors/power-architecture-processors/qoriq-platforms/p-series/qoriq-p2020-and-p2010-dual-and-single-core-communications-processors:P2020>. (accessed 06/2017)
- [9] “Proton 400k-L™ Single Board Computer,” web page, Space Micro, San Diego, CA.
<http://www.spacemicro.com/assets/datasheets/digital/slices/proton400k.pdf>, April 4, 2013.
- [10] J. Logan, “Migrating PowerQUICC® III Processors to QorIQ™ Platforms,” Freescale Semiconductor presentation (Freescale became NXP Semiconductors N.V.), Nov. 2010.
http://www.freescale.com/files/training/doc/POWERQUICC_TO_QORIQ.pdf (accessed 11/2010).
- [11] Freescale P5020 Website (Freescale became NXP Semiconductors N.V.).
http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=P5020, (accessed 01/2013)
- [12] “RAD750@ radiation-hardened PowerPC microprocessor,” RAD750 product datasheet, BAE Systems, July 2008. http://www.baesystems.com/download/BAES_052281/Space-Products--RAD750-component (accessed 08/22/2013)
- [13] Kyle Bennett, Hardocp benchmark website, Nov. 14, 2011.
(http://hardocp.com/article/2011/11/14/intel_core_i73960x_sandy_bridge_e_processor_review/4) (accessed 01/2013)
- [14] “P2 Platform Series: Dual core performance in a single core power envelope,” Freescale Semiconductor Brochure (Freescale became NXP Semiconductors N.V.), 2009. <http://www.datasheetarchive.com/P2-Series-datasheet.html> (accessed 10/31/2013)
- [15] “How Far We’ve Come: 35 Years of Processing Power,” *Scott’s Soapbox*, website.
<http://scottsoapbox.com/2011/11/30/how-far-weve-come-35-years-of-processing-power/> (accessed 01/2013)
- [16] R. Koga, W. A. Kolasinski, M. T. Marra, and W. A. Hanna, “Techniques of Microprocessor Testing and SEU Rate Prediction,” *IEEE Trans. Nucl. Sci.*, vol. 32, no. 6, pp. 4219–4224, 1985.
- [17] S. M. Guertin, and F. Irom, “Recent Results for PowerPC Processor and Bridge Chip Testing,” *Radiation Effects Data Workshop (REDW) 2010*, IEEE, pp. 8, 20–23, July 2010.
- [18] S. M. Guertin and F. Irom, “Processor SEE Test Design,” presented at *Single Event Effects Symposium*, La Jolla, CA 2009.
- [19] S. Guertin, *P2020 Proton Test Report*, March 24, 2011,” NASA Electronic Parts and Packaging Program (internal document), Goddard Space Flight Center, Greenbelt, MD, 2012.
- [20] M. Malone, “On-board Processing Expandable Reconfigurable Architecture (OPERA) Program Overview,” presented at *Fault Tolerant Spaceborne Computing Employing New Technologies Workshop*, May 29, 2008.
<http://www.zettaflops.org/spc08/Malone-Govt-OPERA-FT-Spaceborne-Computing-Workshop-rev3.pdf> (accessed 10/31/2013)

- [21] N. Seifert, Z. Xiaowei, D. Moyer, R. Mueller, R. Hokinson, N. Leland, M. Shade, and L. Massengill, "Frequency Dependence of Soft Error Rates for Sub-Micron CMOS Technologies," presented at *Electron Devices Meeting*, Dec. 2–5, 2001 2001; also *IEDM Technical Digest International*, pp. 14.4.1–14.4.4, 2001.
- [22] F. Irom and F. H. Farmanesh, "Frequency Dependence of Single-Event Upset in Advanced Commercial PowerPC Microprocessors," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 6, pp. 3505–3509, 2004.
- [23] J. Benedetto, P. Eaton, K. Avery, D. Mavis, M. Gadlage, T. Turflinger, P. E. Dodd, and G. Vizkelethy, "Heavy Ion-Induced Digital Single-Event Transients in Deep Submicron Processes," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 6, pp. 3480–3485, 2004.
- [24] D. G. Mavis and P. H. Eaton, "Soft error rate mitigation techniques for modern microcircuits," in *Proc. Int. Reliability Physics Symp.*, Apr. 2002, pp. 216–225
- [25] M. P. Baze, B. Hughlock, J. Wert, J. Tostenrude, L. Massengill, O. Amusan, R. Laco, K. Lilja, and M. Johnson, "Angular dependence of single event sensitivity in hardened flip/flop designs," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 6, pp. 3295–3301, Dec. 2008.
- [26] E. H. Cannon and M. Cabanas-Holmen, "Heavy ion and high energy proton-induced single event transients in 90 nm inverter, NAND and NOR gates," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 6, pp. 3511–3518, Dec. 2009.
- [27] L. D. Edmonds, *Analysis of Single-Event Upset Rates in Triple-Modular Redundancy Devices*, JPL Publication 9-6, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 2009.
- [28] G. R. Allen, L. D. Edmonds, C. W. Tseng, G. Swift, and C. Carmichael, "Single-Event Upset (SEU) Results of Embedded Error Detect and Correct Enabled Block Random Access Memory (Block RAM) within the Xilinx XQR5VFX130," *IEEE Trans. Nucl. Sci.*, vol. 57, pp. 3426–3431, Dec. 2010.
- [29] R. Velazco, G. Foucard, and P. Peronnard, "Combining Results of Accelerated Radiation Tests and Fault Injections to Predict the Error Rate of an Application Implemented in SRAM-based FPGAs," *IEEE Trans. Nucl. Phys.*, vol. 57, pp. 3500–3505, 2010.
- [30] S. M. Guertin, B. Wie, M. K. Plante, A. Berkley, L. S. Walling, and M. Cabanas-Holmen, "SEE Test Results for Maestro Microprocessor," *RADECS Data Workshop*, Noordwijk, The Netherlands, 2012.
- [31] ASTM F 1192, "Standard Guide for the Measurement of Single Event Phenomena (SEP) Induced by Heavy Ion Irradiation of Semiconductor Devices," ASTM International. <http://www.astm.org/Standards/F1192.htm> (accessed 8/22/2013)
- [32] *Test Procedures for the Measurement of Single-Event Effects in Semiconductor Devices from Heavy Ion Irradiation*, EIA-JESD-57, JEDEC Test Standard, Electronic Industries Association, Arlington, VA, 1996. <http://cds.cern.ch/record/1373656>
- [33] J. R. Shwank, M. R. Shaneyfelt, and P. E. Dodd, "Radiation Hardness Assurance Testing of Microelectronic Devices and Integrated Circuits: Test Guideline for Proton and Heavy Ion Single-Event Effects," *Transactions on Nuclear Science*, vol. 60, issue 3, pp. 2101–2118, June 2013.
- [34] S. M. Guertin, C. Hafer, and S. Griffith, "Investigation of Low Cross Section Events in the RHBD/FT UT699 Leon 3FT," *Radiation Effects Data Workshop (REDW), Los Vegas, NV*, 2011, 978-1-4577-1283-8/11, IEEE pp.1–8, July 25–29, 2011. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6062536>
- [35] G. M. Swift, F. H. Farmanesh, S. M. Guertin, F. Irom, and D. G. Millward, "Single-Event Upset in the Power PC750 Microprocessor," *IEEE Trans. Nucl. Sci.*, vol. 48, no. 6, pp. 1822–1827, 2001.
- [36] F. Irom, F. F. Farmanesh, A. H. Johnston, G. M. Swift, and D. G. Millward, "Single-event upset in commercial silicon-on-insulator PowerPC microprocessors," *IEEE Trans. Nucl. Sci.*, vol. 49, no. 6, pp. 3148–3155, Dec. 2002.
- [37] R. Koga, W. A. Kolasinski, M. T. Marra, and W. A. Hanna, "Techniques of Microprocessor Testing and SEU-Rate Prediction," *IEEE Trans. Nucl. Sci.*, vol. 32, no. 6, pp. 4219–4224 (1985).
- [38] F. Bezerra and J. Kuitunen, "Analysis of the SEU behavior of PowerPC 603R under heavy ions," *Radiation and Its Effects on Components and Systems, Proceedings of the 7th European Conference on*, 2003, SP-536, European Space Agency, IEEE 03TH8776, pp. 289–293, 2003.
- [39] D. M. Hiemstra and A. Baril, "Single Event Upset Characterization of the Pentium MMX and Pentium II Microprocessors Using Proton Irradiation," *IEEE Trans. Nucl. Sci.*, vol. 46, no. 6, pp. 1453–1460, 1999.
- [40] J. W. Howard, Jr., M. A. Carts, R. Stattel, C. E. Rogers, T.L. Irwin, C. Dunsmore, J. A. Sciarini, and K. A. LaBel, "Total dose and single event effects testing of the Intel pentium III (P3) and AMD K7 microprocessors," *Radiation Effects Data Workshop, Vancouver, Canada*, 2001 IEEE, pp. 38–47, 2001. <http://nepp.nasa.gov/docuploads/5EE354A1-EEB8-4D06-957814163F9431DF/w7.pdf>

- [41] D. Czajkowski, M. Pagey, and D. Seidenspinner, "Proton Testing of SEFI Mitigation Technique for Microprocessors," white paper, Space Micro Inc., San Diego, CA. 2004.
- [42] P. Rech, *Soft Errors Induced by Neutrons and Alpha Particles in Systems on Chips*, PhD thesis, University of Padova, Italy, 2010.
- [43] H. Quinn, A. Manuzzato, J. Barton, M. Hart, T. Fairbanks, N. Dallmann, R. and DesGeorges, *High-Performance Computing for Airborne Applications*, Document Number: LA-UR-10-04441, Los Alamos National Laboratory, Albuquerque, NM, 2010.
- [44] T. Amort, W. Snapp, J. Evans, J. Popp, M. Cabanas-Holmen, and E. Cannon, "90nm RHBD ASIC Design Capability," presented at *Respace/MAPLD Conference*, [Military and Aerospace Programmable Logic Devices], 2011.
- [45] A. Agarwal, "The Raw Tiled Processor Architecture," Massachusetts Institute of Technology, Boston, MA, 2004.
- [46] J. Suh, K. J. Mighell, D. Kang, and S. P. Crago, "Implementation of FFT and CRBLASTER on the Maestro Processor," *Aerospace Conference, 2012*, IEEE, pp. 1-6, March 2012.
- [47] E. H. Cannon, M. Cabanas-Holmen, J. Wert, T. Amort, R. Brees, J. Koehn, M. Meaker, and E. Normand, "Heavy Ion, High Energy, and Low Energy Proton SEE Sensitivity of 90-nm RHBD SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 6, pp. 3493–3499, 2010.
- [48] Gaisler Research/Aeroflex Gaisler, web page, <http://www.gaisler.com> (accessed 01/2011)
- [49] J. Andersson, J. Gaisler, and R. Weigand, "Next Generation Multipurpose Microprocessor," *Data Systems in Aerospace*, Budapest, Hungary, 2010. <http://microelectronics.esa.int/ngmp/NGMP-DASIA10-Paper.pdf>
- [50] C. Hafer, "Single Event Effects Qualification Summary for WX07A 0.25 μ m Test Chip," Aeroflex Microelectronic Solutions, 2004
- [51] C. Hafer, "Single Event Effects Qualification Summary for the UT699 LEON 3FT Processor," Aeroflex, 2008.
- [52] C. Hafer, S. Griffith, S. Guertin, J. Nagy, F. Sievert, J. Gaiser, and S. Habinc, "LEON 3FT Processor Radiation Effects Data," *Radiation Effects Data Workshop, 978-1-4244-5092-3/09/*, IEEE, 2009
- [53] S. Guertin, "P2020 and P5020 Heavy Ion Test Report (August 27, 2011)" NASA Electronic Parts Program (internal document), Goddard Space Flight Center, Greenbelt, MD, 2012.
- [54] "Freescale's Power Architecture® Technology Licensed for Space Missions," BAE Systems Newsroom website, March 8, 2012. http://www.baesystems.com/article/BAES_043443/freescales-power-architecture-technology-licensed-for-space-missions, (accessed 12/2012)
- [55] "Apple Introduces iPad Mini," Apple press release, Oct. 23, 2012. www.apple.com, (accessed 01/2013)
- [56] Anand Lal Shimpi; "iPad 4 GPU Performance Analyzed: PowerVR SGX 554MP4 Under the Hood," AnandTech website, Nov. 2, 2012. www.anandtech.com, (accessed 11/2012)
- [57] E. Mack, "iPhone processor way outguns Mars Curiosity rover's," CNET website (news.cnet.com), Aug. 6, 2012. (accessed 08/2012)
- [58] I. Paul, "Apple's iPhone 4 Hired as Tricorder for Space Station," *PCWorld*, June 13, 2011. www.pcworld.com (accessed 07/2011)
- [59] T. Barjarin, "ARM vs. Intel: How the Processor Wars Will Benefit Consumers," *Time Magazine* online, July 16, 2012. <http://techland.time.com/2012/07/16/arm-vs-intel-how-the-processor-wars-will-benefit-consumers-most/>
- [60] J. R. Azambuja, M. Altieri, J. Becker, and F. Kastensmidt, "HETA: hybrid error-detection technique using assertions," *Transactions on Nuclear Science*, vol. 60, no. 4, Aug. pp. 2805–2812, 2013.
- [61] A. J. Tylka, J. H. Adams, Jr., P. R., Boberg, B. Brownstein, W. F. Dietrich, E. O. Flueckiger, et al., "CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code," *IEEE Trans. Nucl. Sci.*, vol. 44, pp. 2150–2160, 1997.
- [62] CREME website, <https://creme.isde.vanderbilt.edu/> (accessed 09/2012)
- [63] SPENVIS website, <http://www.spennis.oma.be/> (accessed 10/2016)
- [64] L. Edmonds, "SEU cross sections derived from a diffusion analysis," *IEEE Trans. Nucl. Sci.*, vol. 43, pp. 3207–3217, Dec. 1996.
- [65] K. P. Rodbell, D. F. Heidel, J. A. Pellish, P. W. Marshall, H. H. K. Tang, C. E. Murray, K. A. LaBel, M. S. Gordon, K. G. Stawiasz, J. R. Schwank, M. D. Berg, H. S. Kim, M. R. Friendlich, A. M. Phan, and C. M.

- Seidleck, "32 and 45 nm Radiation-Hardened-by-Design (RHBD) SOI Latches," *IEEE Trans. Nucl. Sci.*, vol. 58, pp. 2702–2710, Dec. 2011.
- [66] H. R. Schwank, J. R.; M. R. Shaneyfelt, J. Baggio, P. E. Dodd, J. A. Felix, V. Ferlet-Cavrois, P. Paillet, D. Lambert, F. W. Sexton, G. L. Hash, and E. Blackmore, "Effects of Particle Energy on Proton-Induced Single-Event Latchup," *IEEE Trans. Nucl. Sci.*, vol. 52, pp. 2622–2629, 2005.
- [67] W. L. Bendel, and E. L. Petersen, "Predicting Single Event Upsets in the Earth's Proton Belts," *IEEE Trans. Nucl. Sci.*, NS-31, 1201, 1984.
- [68] J. R. Schwank, M. R. Shaneyfelt, V. Ferlet-Cavrois, P. E. Dodd, E. W. Blackmore, J. A. Pellish, K. P. Rodbell, D. F. Heidel, P. W. Marshall, K. A. LaBel, P. M. Gouker, N. Tam, R. Wong, S.-J. Wen, R. A. Reed, S. M. Dalton, and S. E. Swanson, "Hardness Assurance Testing for Proton Direct Ionization Effects" *IEEE Trans. Nucl. Sci.*, vol. 59, no. 4, pp. 1197–1202, 2012.
- [69] D. F. Heidel, J. A. Pellish, P. W. Marshall, K. P. Rodbell, K. A. LaBel, J. R. Schwank, M. Hakey, M. D. Berg, P. E. Dodd, M. R. Friendlich, M. R., A. D. Phan, C. M. Seidleck, M. R. Shaneyfelt, and M. A. Xapsos, "Proton and heavy ion testing of 45 nm and 65 nm SOI SRAMS," *IEEE Trans. Nucl. Sci.*, vol. 56, pp. 3499–3504, Dec. 2009.
- [70] M. Baze, B. Hughlock, J. Wert, J. Tostenrude, L. Massengill, O. Amusan, R. Lacoce, K. Lilja, and M. Johnson, "Angular dependence of single event sensitivity in hardened flip/flop designs," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 6, pp. 3295–3301, Dec. 2008. <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4723728> (accessed 06/30/2017)
- [71] N. A. Dodds, "Hardness Assurance for Low-Energy Proton-Induced Single-Event Effects," Sandia Report, SAND2015-7243, Aug. 2015.
- [72] P. M. O'Neill, G. D. Badhwar, and W. X. Culpepper, "Internuclear Cascade - Evaporation Model for LET Spectra of 200 MeV Protons Used for Parts Testing," *IEEE Trans. Nucl. Sci.*, vol. 45, pp. 2467–2474, 1998.
- [73] C. Pham, H. Malcom, R. Maurer, D. Roth, and K. Strohhahn, "LEON3FT Proton SEE Test Results for the Solar Probe Plus Program," *Radiation Effects Data Workshop (REDW)*, Los Vegas, NV, 2011, 978-1-4577-1283-8/11, IEEE pp.1–4, July 25–29, 2011. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6062535> (accessed 10/31/2013)
- [74] V. Pouget, D. Lewis, H. Lapuyade, R. Briand, P. Fouillat, L. Sarger, and M.-C. Calvet, "Validation of radiation hardened designs by pulsed laser testing and SPICE analysis," *Microelectronics Reliability*, vol. 39, pp. 931–935, 1999.
- [75] F. W. Sexton, "Microbeam studies of single-event effects," *IEEE Trans. Nucl. Sci.*, vol. 43, pp. 687–695, Apr. 1996.
- [76] L. D. Edmonds, "SEU Cross Sections Derived for a Diffusion Analysis," *IEEE Trans. Nucl. Sci.*, vol. 43, pp. 3207–3217, 1996.
- [77] "Radiation Effects Facility," *TAMU Beam Range Information*, website, Texas A&M University, College Station, TX. <http://cyclotron.tamu.edu/ref/beams.php>, (9/2012)
- [78] F. Faure, R. Velazco, M. Violante, M. Rebaudengo, and M. S. Reorda, "Impact of data cache memory on the single event upset-induced error rate of microprocessors," *IEEE Trans. Nucl. Sci.*, vol. 50, pp. 2101–2106, Dec. 2003.
- [79] H. Quinn, D. Black, W. Robinson, and S. Buchner, "Fault Simulation and Emulation Tools to Augment Hardness Assurance Testing," *IEEE Trans. Nucl. Sci.*, vol. 60, pp. 2119–2142, May 2013.
- [80] "Physical Constants of IC Package Materials," *Physical Constants of IC Package Materials: Databook*, Intel. <http://www.intel.com/content/www/us/en/processors/packaging-chapter-05-databook.html>, (01/2013)
- [81] "Alcoa 6063 Alloy datasheet," "Al-392, ASM International, Materials Park, OH, April 2005. <https://www.scribd.com/document/206631152/ALCOA-6063-Material-Data-Sheet> (accessed Aug. 2, 2017)
- [82] B. von Przewoski, T. Rinckel, W. Manwaring, G. Broxton, M. Chipara, T. Ellis, E. R. Hall, A. Kinser, K. M. Murray, and C. C. Foster, "Beam Properties of the new Radiation Effects Research Stations at Indiana University Cyclotron Facility," *Radiation Effects Data Workshop*, Atlanta, GA, 0-7803-8697-3/04, IEEE, July 2004.
- [83] "Usage – Radiation Effects," UCD Crocker Nuclear Laboratory Radiation Effects Usage Web Page, University of California, Davis. <http://crocker.ucdavis.edu/cyclotron/usage-radiation-effects/> (accessed 10/31/2013)

- [84] E. W. Blackmore, "Operation of the TRIUMF (20-500 MeV) Proton Irradiation Facility," *Radiation Effects Data Workshop*, Atlanta, GA, pp. 1-5, 2004.
- [85] "Most Commonly Used Ions," Brookhaven National Laboratory Tandem Van de Graaf website, Upton, NY. <http://tvdg10.phy.bnl.gov/species.html> (accessed 9/2012)
- [86] UCB Ion Website: <http://cyclotron.lbl.gov/base-rad-effects/heavy-ions/cocktails-and-ions> (9/2012)
- [87] "Heavy Ion Irradiation Facility (HIF)," website of Université catholique de Louvain, Louvain-la-Neuve, Belgium. <http://www.cyc.ucl.ac.be/HIF/HIF.html> (accessed 9/2012)
- [88] "Beam Ion Species and Energies Used Previously at NSRL, NASA/BNL Space Radiation Program website, Brookhaven National Laboratory, Upton, NY. http://www.bnl.gov/medical/NASA/CAD/Beam_Ion_Species_and_Energy.asp (accessed 9/2012)
- [89] J. F. Ziegler, "The Stopping and Range of Ions in Matter", software application, <http://www.srim.org/> (accessed 10/05/2016)
- [90] "ASAP 1," UltraTec Manufacturing, Inc., website. <http://www.ultratecusa.com/asap-1> (accessed 10/31/2013)
- [91] J. H. Elder, J. Osborn, W. A. Kolasinski, and R. Koga, "A method for characterizing a microprocessor's vulnerability to SEU," *IEEE Trans. Nucl. Sci.*, vol.35, no. 6, pp.1678–1681, Dec. 1988.
- [92] D. Limbrick, *Mitigation of Radiation-Induced Soft Errors Using Temporal Embedded Signature Monitoring*, Master's Thesis, Vanderbilt University, Nashville, TN, 2009.
- [93] Dhrystone Source Code can be found here: <http://classes.soe.ucsc.edu/cmpe202/benchmarks/standard/dhrystone.c> (accessed 10/2013)
- [94] J. Howard, K. LaBel, M. A. Carts, R. Stattel, C. E. Rogers, and T. L. Irwin, "Summary of the Radiation Testing of the Intel Pentium III (P3) Microprocessor," MAPLD 2002 [Military and Aerospace Programmable Logic Devices Workshop], 2002. http://nepp.nasa.gov/DocUploads/8A499785-90F7-4A77-B90913144A0402D1/MAPLD02_Howard_P3.pdf (accessed 10/31/2013)
- [95] M. Traxler, R. Becker, I. Frohlich, W. Kuhn, J. Lehnert, C. Lichtblau, E. Lins, M. Petri, M.-A. Pleier, and J. Ritman, "The multi-level trigger system of the HADES detector," Real Time Conference, 1999, *1999 IEEE Conference on Real-Time Computer Applications in Nuclear Particle and Plasma Physics*, 11th IEEE NPSS Santa Fe, NM, pp. 529–532, 1999.
- [96] GRMON2 User's Manual, GRMON-UM Version 2.0.30, Aeroflex Gaisler, September 2012, <http://www.gaisler.com/doc/grmon2.pdf>
- [97] CodeWarrior Development Tools, website, Freescale (Freescale became NXP Semiconductors N.V.). http://www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME (accessed 9/2012).
- [98] K. J. Mighell, "Benchmarking CRBLASTER on the 350-MHz 49-core Maestro Development Board," archived at Cornell University website for submission to ADASS XXI, Paris, 2011. <http://arxiv.org/abs/1201.4788>, 2012
- [99] G. M. Swift, F. H. Farmanesh, S. M. Guertin, F. Irom, and D. G. Millward, "Single-Event Upset in the Power PC750 Microprocessor," *IEEE Trans. Nucl. Sci.*, vol. 48, no. 6, pp. 1822–1827, 2001.
- [100] F. Irom and F. H. Farmanesh, "Single-event upset in highly scaled commercial silicon-on-insulator PowerPC microprocessors," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 5, pp. 1524–1529, Oct. 2005.
- [101] F. Irom, F. Farmanesh, and C. K. Kouba, "Single-Event Upset and Scaling Trends in New Generation of the Commercial SOI PowerPC Microprocessors," *IEEE Trans. Nucl. Sci.*, vol.53, no.6, pp.3563–3568, Dec. 2006.
- [102] "COMX-P5020 High Performance Freescale QorIQ™ Module," Emerson Network Power sales website. www.emersonnetworkpower.com, accessed 12/2012.
- [103] S. M. Guertin, *March 2012 P2020 Dual Core Test Report*, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, Jan. 2013. https://nepp.nasa.gov/files/25659/12_126_JPL_%20Guertin_%20P2020%20Dual%20Core%20SEE%20Test%20Report%20rec%2012%209%2013.pdf (accessed 8/2/2017)
- [104] H. Quinn, A. Manuzzato, J. Barton, M. Hart, T. Fairbanks, N. Dallmann, and R. DesGeorges, *High-Performance Computing for Airborne Applications*, Document Number: LA-UR-10-04441, Los Alamos National Laboratory, Los Alamos, NM, 2010.
- [105] Example c source code for the Whetstone benchmark can be found at <http://www.netlib.org/benchmark/whetstone.c> (1/2013)

- [106] P. Bernardi, M. Grosso, P. Rech, M. Sonza Reorda, D. Appello, S. Gerardin, and A. Paccagnella, "DfT Reuse for Low-Cost Radiation Testing of SoCs: a case study," in *Proc. IEEE VLSI Test Symposium 2009*, Santa Cruz, CA, pp. 276–281.
- [107] D. Petrick, W. Powell, J. W. Howard, Jr., and K. A. LaBel, "Virtex-II Pro SEE Test Methods and Results," MAPLD04_Petrick_paper, *MAPLD Proceedings [Military and Aerospace Programmable Logic Devices Workshop]*, 2004. http://radhome.gsfc.nasa.gov/radhome/papers/mapld04_petrick_paper.pdf (accessed 10/31/2013)
- [108] M. W. Learn, *Evaluation of the Leon3 Soft-Core Processor Within a Xilinx Radiation-Hardened Field-Programmable Gate Array*, Sandia Report, SAND2012-0454, Sandia National Laboratories, Albuquerque, NM, 2012.
- [109] R. Monreal, C. Carmichael, and G. Swift, "Single-Event Characterization of Multi-Gigabit Transceivers (MGT) in Space-Grade Virtex-5QV Field Programmable Gate Arrays (FPGA)," Radiation Effects Data Workshop, IEEE, July 2011. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6062534>
- [110] G. Swift, C. Carmichael, G. Allen, G. Madias, E. Miller, and R. Monreal, "Compendium of XRTC Radiation Results on All Single-Event Effects Observed in the Virtex-5QV," *MAPLD 2011 [Military and Aerospace Programmable Logic Devices Workshop]*, Aug. 2011. https://nepp.nasa.gov/respace_mapld11/talks/tue/MAPLD/1420%20-%20Swift.pdf (accessed 10/31/2013)
- [111] P. Peronnard, R. Ecoffet, M. Pignol, D. Bellin, and R. Velazco, "Predicting the SEU Error Rate through Fault Injection for a Complex Microprocessor," *2008 IEEE International Symposium on Industrial Electronics*, Cambridge, UK, June 30–July 2, 2008, pp. 2288–2292, 2008.
- [112] H. Guzman-Miranda, J. N. Tombs, and M. A. Aguirre, "FT-UNSHADES-uP: A platform for the analysis and optimal hardening of embedded systems in radiation environments," *2008 IEEE International Symposium on Industrial Electronics*, Cambridge, UK, June 30–July 2, 2008, pp. 2276–2281, 2008.
- [113] D. Jaeckle and A. Sikora, "Thermal modeling of homogeneous embedded multi-core processors," *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, New Delhi, 2014, pp. 588-593.

9.0 ACRONYMS

AHB	Advanced High-performance Bus (AMBA)
ALU	arithmetic logic unit
AMBA	Advanced Microcontroller Bus Architecture
amu	atomic mass unit
APB	Advanced Peripheral Bus (AMBA)
Aramica	poly (p-phenylene terephthalamide)-aramid
ARM	ARM Holdings plc (also owner of former Artisan Components, Inc.)
ASIC	application specific integrated circuit
ASTM International	(organization formerly known as) American Society for Testing and Materials
BAE	BAE Systems
BERT	bit error rate tester
BGA	ball grid array
BNL	Brookhaven National Laboratory
BTK	(Tilera) board test kit
CAN	Controller Area Network
CLK	clock
cPCI	compact peripheral component interconnect
CPU	central processing unit
CRBLASTER	(algorithm and a parallel-processing image-analysis application)
CRC	cyclic redundancy check
CRÈME	Cosmic Ray Effects on Micro-Electronics code
DARPA	Defense Advanced Research Projects Agency
DBE	double-bit error
DDR2	double data rate 2 (2 nd generation)
DfT	design for testability
DICE	dual interlocked cell
DMA	direct memory access
DSU	debug support unit
DTRA	Defense Threat Reduction Agency
DUT	device under test

ECC	error correcting code
EDAC	error detection and correction
FF	flip-flop
FFT	fast Fourier transform
FIFO	first in, first out
FPGA	field programmable gate array
FPU	floating point unit
Freescale Semiconductor, Inc. (became NXP Semiconductors N.V.)	
FT	fault tolerant
FTB	functional test board
GBE	Gigabit Ethernet
GCR	galactic cosmic ray
GEO	geosynchronous Earth orbit
×GOPS	giga-operations per second (billions of operations per second)
GPIO	general purpose input/output port
GPR	general purpose register
GPU	graphics processing unit
GRMON	Gaisler Research Monitor
Hex	indicates a number is base 16, or hexadecimal
HI	heavy ion
HPI	hardware platform interface
IC	integrated circuit
IIC/I2C (-H/-S)	Inter-Integrated Circuit interface (-high-speed/-standard)
IDN	Input/Output Dynamic Network
I/O (IO)	input/output
IrqCtrl	interrupt controller
IUCF	Indiana University Cyclotron Facility (see also ISAT)
ISAT	the Integrated Science and Accelerator Technology Hall (formerly IUCF)
ISS	International Space Station
ITC	(Maestro) Interim Test Chip

JEDEC Solid State Technology Association	(formerly known as) Joint Electron Device Engineering Council
JPL	Jet Propulsion Laboratory
JTAG	Joint Test Action Group (also refers to hardware used to operate a port defined under the group)
LEON	32-bit microprocessor core – not an acronym
LET	linear energy transfer
MAC	Media Access Controller
MBE	multiple-bit error
MBU	multiple bit upsets
MDN	Memory Dynamic Network
MeV	mega electron volts
MGH	Massachusetts General Hospital
MGT	multi-gigabit transceivers
MIPS	millions of instructions per second (used to measure performance but not as accurate in the era of multi-operation instructions—see MOPS)
MMU	memory management unit
MOPS	millions of operations per second
NASA	National Aeronautics and Space Administration
NEPP	NASA Electronic Parts and Packaging Program
NRL	NASA Research Laboratory
NSRL	NASA Space Radiation Laboratory
NVM	nonvolatile memory
OPERA	Onboard Processing Expandable Architecture
OS	operating system
pBIST	programmable built-in self test
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PSR	processor state register
PWR	power
RADEF	Radiation Effects Facility (Finland)
RAM	random access memory

RDB	(Freescale) reference design board
RHA	radiation hardness
RHBD	radiation hardened by design
RMGII	Reduced Gigabit Media Independent Interface
ROCKSTER	Rock Segmentation Through Edge Regrouping (application)
RPP	rectangular parallapiped
RTOS	real time operating system
SATA	Serial Advance Technology Attachment (computer bus drive)
SD	secure digital (connector)
SEB	single-event burnout
SECDDED	single-error correction, double-error detection
SEE	single-event effects
SEFI	single event functional interrupt
SEL	single-event latchup
SERDES	serializer/deserializer
SET	single-event transient
SEU	single-event upset
SNR	signal-to-noise ratio
SOA	safe operating area
SOC	system on a chip [device]
SOI	silicon on insulator
Space Micro	Space Micro Inc.
SpaceRad	(type of rate-calculation software)
SPENVIS	SPace ENVironment Information System
SPI	Serial Peripheral Interface
SRAM	static random-access memory
SRIM	stopping and range of ions
SSED	(Boeing) Solid State Electronic Development Static Network
STN	Static Network
TAMU	Texas A&M University Cyclotron
TDN	Tile Dynamic Network
TID	total ionizing dose
TMR	triple-module redundant

TRIUMF	Tri-University Meson Facility
UART	Universal Asynchronous Receiver/Transmitter
UCB	University of California at Berkeley
UCD	University of California at Davis
UCL	Université catholique de Louvain (Belgium)
UDN	User Dynamic Network
USB	universal serial bus (connector)
USG	United States Government
UT699	SPARC V8 compliant compiler
wim	window invalid mask
XAUI	X [for 10 Gigabit] attachment user interface
ZIF	zero insertion force