Final Report for the Remote Exploration and Experimentation (REE) Program's Testing of the Intel Pentium III (P3), AMD K7 Microprocessors and the Myrinet Network Hardware[†]

James W. Howard Jr., Kenneth A. LaBel, Martin A. Carts, Ronald Stattel, Charles E. Rogers, Timothy L. Irwin, Zoran Kahric and J. Anthony Sciarini

I. Introduction

Many future NASA missions will require extensive on-board computation capability which raises the issue of availability, cost and capability of radiation hardened or radiation tolerant microprocessor systems. Radiation hardened computer systems are often costly and are actually two or three generations behind in computational capability, a significant shortfall for missions that may require state-of-the-art (SOTA) capability.

To confront these issues, NASA instituted the Remote Exploration and Experimentation Project (REE) with the goal of transferring commercial supercomputer technology into space using SOTA, low power, non-radiation-hardened, commercial-off-the-shelf (COTS) hardware and software to the maximum extent possible.

As part of this project, the radiation response of the Intel Pentium III (P3) and AMD K7 microprocessors and their associated bridge chips were evaluated. This final report summarizes the Total Ionizing Dose (TID) response (both proton and cobalt-60) and single event effects (proton and heavy-ion) observed. This report also summarizes the results observed for the Myrinet network hardware. Not included in this report is a final proton test conducted after the conclusion of the project. The test report for this test is included in the CD distribution and should be referenced for final proton data.

Included are the test reports for all testing, all raw data, all test and analysis software developed for the testing and a copy of all papers and presentations based on this project.

II. Testing

A. Overview

To investigate the radiation sensitivity of the Pentium III and K7 microprocessors, total ionizing dose (both Cobalt-60 and protons) and single event effects (both with protons and heavy ions) testing must be done. Our plan was to utilize the GSFC total dose facility for the Cobalt-60 exposure and external facilities for the proton and heavy-ion testing. The advantage is that the TID testing at GSFC can be done in parallel with the single event testing. Indiana University Cyclotron Facility was used for the proton testing mainly for its convenience and availability. UC Davis was used for the Myrinet testing. The Texas A&M University Cyclotron was used for the heavy-ion testing, as it is the only currently available facility in the United States that can supply ion beams with sufficient energy to penetrate the die substrate.

The devices used in this project are detailed in three tables. Table I gives the pertinent information for all the Pentium III devices, Table II for the AMD K7 devices and Table III for the Myrinet network devices.

[†] This work was performed at NASA/GSFC for the Remote Exploration and Experimentation Project. This project is part of NASA's High Performance Computing and Communications Program, and is funded through the NASA Office of Space Sciences.

TABLE I
Pentium III DEVICE UNDER TEST (DUT) TABLE

Device	Vendor	Rated Speed	Operating Speed	Package	Source	Test Type	Package Markings
Pentium III	Intel	550 MHz	550 MHz	SC242	Protons	SEE	550/256/100/1.65V S1 90050493-0099 MALAY imc '99 SL3V5
Pentium III	Intel	550 MHz	Unbiased	SC242	Co-60	TID	550/256/100/1.65V S1 90050493-0307 MALAY imc '99 SL3V5
Pentium III	Intel	650 MHz	650 MHz	SC242	Protons	SEE	650/256/100/1.65V S1 10100418-0293 PHILIPPINES imc '99 SL3KV
Pentium III	Intel	650 MHz	Unbiased	SC242	Protons	TID	650/256/100/1.65V S1 10100418-0176 PHILIPPINES imc '99 SL3KV
Pentium III	Intel	650 MHz	650MHz	SC242	Protons	TID	650/256/100/1.65V S1 10100418-0161 PHILIPPINES imc '99 SL3KV
Pentium III	Intel	650 MHz	Unbiased	SC242	Co-60	TID	650/256/100/1.65V S1 10160248-0411 PHILIPPINES imc '99 SL452
Pentium III	Intel	650 MHz	Unbiased	SC242	Co-60	TID	650/256/100/1.65V S1 10160260-0075 PHILIPPINES imc '99 SL452
Pentium III	Intel	700 MHz	700 MHz	SC242	Protons	SEE	700/256/100/1.65V S1 90160187-0108 MALAY imc '99 SL454
Pentium III	Intel	700 MHz	700 MHz	SC242	Protons	SEE	700/256/100/1.65V S1 90160187-0055 MALAY mc '99 SL454
Pentium III	Intel	700 MHz	700 MHz	SC242	Protons	TID	700/256/100/1.65V S1 90160187-0057 MALAY imc '99 SL454
Pentium III	Intel	700 MHz	Unbiased	SC242	Co-60	TID	700/256/100/1.65V S1 90160187-0060 MALAY imc '99 SL454
Pentium III	Intel	750 MHz	500, 750 MHz	SC242	Protons	SEE	750/256/100/1.65V S1 90260050-0092 MALAY imc '99 SL456
Pentium III	Intel	800 MHz	533, 800 MHz	SC242	HI	SEE	800/256/100/1.65V S1 90240221-0006 MALAY imc '99 SL457
Pentium III	Intel	800 MHz	800 MHz	SC242	Co-60	TID	800/256/100/1.65V S1 90240169-0091 MALAY imc '99 SL457
Pentium III	Intel	850 MHz	850 MHz	SC242	Protons	SEE	850/256/100/1.65V S1 10280400-0071 Philippines imc '99 SL47M
Pentium III	Intel	850 MHz	566, 850 MHz	SC242	Protons	SEE	850/256/100/1.65V S1 10280400-0293 Philippines imc '99 SL47M
Pentium III	Intel	850 MHz	Unbiased	SC242	Co-60	TID	850/256/100/1.65V S1 10280400-0262 Philippines imc '99 SL47M
Pentium III	Intel	850 MHz	566, 850 MHz	SC242	Protons	SEE	850/256/100/1.65V S1 10280400-0308 Philippines imc '99 SL47M
Pentium III	Intel	933 MHz	466, 700, 933 MHz	SC242	Protons	SEE	933/256/133/1.7V S1 00280415-0224 COSTA RICA imc '99 SL47Q
Pentium III	Intel	933 MHz	466, 700, 933 MHz	SC242	HI	SEE	933/256/133/1.7V S1 00280415-0269 COSTA RICA imc '99 SL47Q
Pentium III	Intel	933 MHz	466, 700, 933 MHz	SC242	HI	SEE	933/256/133/1.7V S1 11040081-0098 Phillippines imc '00 SL4KK
Pentium III	Intel	933 MHz	Unbiased	SC242	Co-60	TID	933/256/133/1.7V S1 00280415-0068 COSTA RICA imc '99 SL47Q
Pentium III	Intel	933 MHz	Biased	SC242	Co-60	TID	933/256/133/1.7V S1 11040081-0488 Phillippines imc '00 SL4KK
Pentium III	Intel	933 MHz	700, 933 MHz	PGA370	Protons	SEE	933/256/133/1.7V L045A581-0194 SL4ME
Pentium III	Intel	933 MHz	700, 933 MHz	SC242	Protons	SEE	933/256/133/1.7V S1 11040081-0098 Phillippines imc '00 SL4KK
Pentium III	Intel	933 MHz	933 MHz	PGA370	HI	SEE	933/256/133/1.7V L109A567-0326 Malay SL4ME
Pentium III	Intel	933 MHz	933 MHz	PGA370	HI	SEE	933/256/133/1.75V 3125A341-0245 Costa Rica SL50W imc '01
Pentium III	Intel	1 GHz	0.75, 1.0 GHz	PGA370	Protons	SEE	1000/256/133/1.75V Q112A279-0442 SL4MF
Pentium III	Intel	1 GHz	0.75, 1.0 GHz	PGA370	Protons	SEE	1000/256/133/1.75V Q111A242-0067 SL4MF
Pentium III	Intel	1 GHz	1 GHz	PGA370	HI	SEE	1000/256/133/1.75V Q118A892-1019 Malay '01 SL5DV
Pentium III	Intel	1 GHz	1 GHz	PGA370	HI	SEE	1000/256/133/1.75V Q118A892-1177 Malay '01 SL5DV

TABLE II
AMD K7 DEVICE UNDER TEST (DUT) TABLE

Device	Vendor	Rated Speed	Operating Speed	Package	Source	Test Type	Package Markings
K7	AMD	600 MHz	600 MHz	SC242	Protons	SEE	AMD-K7600MTR51B C 219949147583
K7	AMD	650 MHz	650 MHz	SC242	Protons	SEE	AMD-K7650MTR51B A 210017540094
K7	AMD	650 MHz	650 MHz	SC242	HI	SEE	AMD-K7650MTR51B A 210017540094
K7	AMD	650 MHz	Unbiased	SC242	Protons	TID	AMD-K7650MTR51B A 230015009833
K7	AMD	700 MHz	700 MHz	SC242	Protons	SEE	AMD-K7700MTR51B A 210019614073
K7	AMD	900 MHz	900 MHz	SC242	Protons	SEE	AMD-K7900MNR53B A 210036542751
K7	AMD	1 GHz	1 GHz	SC242	Protons	SEE	AMD-K7100MNR53B A 710026014044
K7	AMD	1 GHz	1 GHz	SC242	Protons	SEE	AMD-K7100MNR53B A 710026147861
K7	AMD	1 GHz	1 GHz	SC242	HI	SEE	AMD-K7100MNR53B A 710026019051

TABLE III
Myrinet Device Under Test (DUT) Table

Device	Vendor	Location	Model Number	Serial Number	Other Part Markings
Switch 1	Myricom	Switch Board 1	M3-SW16-8S	84312	A-0041 0124
Switch 2	Myricom	Switch Board 2	M3-SW16-8S	87091	A-0041 0125
NIC	Myricom	NIC	M3S-PCI64B-2	90110	B-0111 0128
Lanai9	Myricom	NIC	9.1	118	
SerDeSer	Myricom	NIC	1.1	123	
PCIDMA	Myricom	NIC	1.3	126	
Transceiver	Vitesse	NIC	VCS7146RH		0113LUBAD
SRAM	Samsung	NIC	K7N803601M		TKLB53BA KOREA TKG012DA KOREA

B. Testing Completed

Testing was done in all three areas: TID, proton single event effects and heavy-ion single event effects. The TID testing is still in progress as an in-house project in an attempt to find failure levels. The results as of 2/5/2002 are summarized in the following results section.

Three separate proton tests were completed on the Pentium III and K7 processors. An initial test was completed in June of 2000 to understand the issues with testing the complex processors and to do the initial screening for latchup. In December of 2000 a follow-on proton test was conducted that built on the knowledge gained from the first test by greatly enhancing the software so as to be able to fully handle all exceptions. This allowed for longer duration runs that revealed some limitations in the test software for the cache tests. These routines were re-written and a final proton test was conducted in May of 2001. All of these test reports are included with this report.

Two heavy-ion tests were completed. Testing conducted in March of 2001 was based on the test software from the December 2000 proton testing. The data analysis that required the re-write of the cache test software for the proton testing was completed after the March testing. With that in mind and the knowledge gained from the re-written proton testing software, an additional heavy-ion test was deemed necessary. A second heavy-ion test for the "Flip-Chip" Pentium III processors utilizing the latest software was done in October. Both test reports are included in this package.

Finally, testing was done, as an initial proton test, of the crossbar switch device of the Myrinet network system. In addition, five devices on the Myrinet Network Interface Card (NIC) were exposed to the proton beam and the system response measured. The details of this testing is covered in the completely in the Myrinet test report included with this package. The remainder of this final report will deal exclusively with the P3/K7 testing.

III. Testing Systems

A. Test Hardware

The system hardware for all P3 and K7 tests conducted is approximately the same. Details of specifics and differences are found in the appropriate test reports. This section describes the overall test configuration, as is shown in Figure 1.

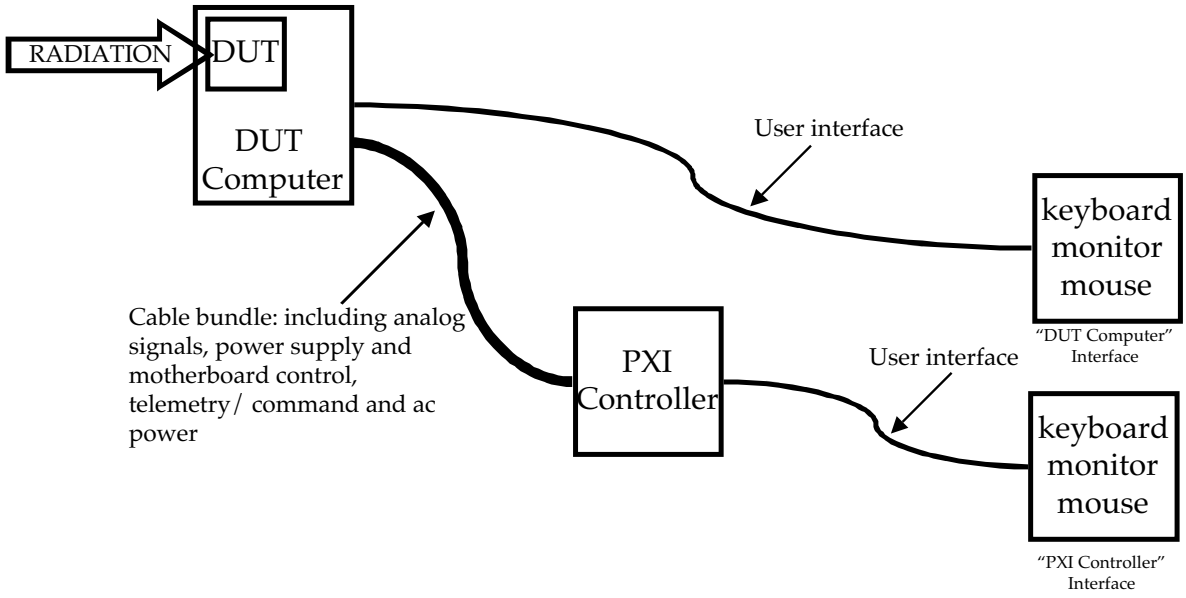

Figure 1. Overall, block diagram of all P3 and K7 tests.

Test Hardware Configuration

Either during or after each irradiation, Devices Under Test (DUTs) were tested for functionality and parametrics in a DUT Computer, consisting of a motherboard and related components. The user interface for this computer resided some distance (from immediate vicinity to ~50′ depending on the test) from the DUT Computer. The DUT Computer was also connected to the PXI Controller for collecting data and controlling the DUT Computer.

The PXI Controller resided at some distance from the DUT Computer, which had similar connections. The location of the PXI Controller, with respect to the DUT Computer and the user interfaces varies for each test.

The DUT was tested during heavy-ion irradiation. The DUT Computer resided a minimum of 10 feet from the PXI Controller and the user interfaces. A keyboard, monitor, and mouse extension was used for the DUT computer but not the PXI Controller.

The DUT was also tested during proton irradiation. To mitigate spurious neutron event, the PXI Controller resided from 15 to 45 feet away from the DUT Computer. It was not located further away, in order that analog sample fidelity be maintained. The user interfaces were located an additional significant distance further from the PXI Controller and thus keyboard/monitor/ mouse extensions were used for both it and the DUT Computer.

For biased irradiations, the setup was similar to the heavy-ion testing setup. For unbiased TID irradiations, the DUTs were tested after each increment in exposure by placing them in a DUT Computer sitting immediately next to the PXI Controller and user interfaces.

DUTs

The Pentium III (P3) is/was available in three different physical form-factors. The original, the SC-242 form-factor was a module with 242 card edge contacts and integrated heatsink and fan. By breaking open the module, the bare die was exposed, though with active layer facedown and 900 micron thick substrate exposed. SC-242 DUTs stand with the plane of the die normal to the motherboard. This and the existence of pre-manufactured extender boards for the SC-242 make it the most desirable form-factor for SEE testing.

Unfortunately, the SC-242 module was apparently a more expensive concession to the inability to efficiently manufacture on-die cache and was eventually obsoleted. All SC-242 processors tested actually did contain on-die cache. Early in this testing program the SC-242 module became unavailable.

The second form-factor of the P3 was adopted for all testing except for TID. The FC-PGA form-factor is a ceramic 370 pin grid array with the die mounted flip-chip and epoxy-sealed onto the PGA ceramic. The FC-PGA fits into a motherboard zero insertion force (ZIF) socket with the surface of the die lying parallel to the surface of the motherboard. An extender for accessing the analog signals was complex beyond the program's needs, so analog measurements, which required shunt-resistor insertion, were abandoned when testing the FC-PGA.

A third form-factor of the P3, the FC-PGA2, is essentially the same as the FC-PGA except that it has a metal heat spreader on top of it, which also protects the die (the corners and edges are particularly susceptible to chipping).

B. Test Software

The DUT software is compiled in Microsoft Visual C++ with a Pharlap Linker Add-in. The tests are written in a combination of C and assembly language. The software is executed using the Pharlap Real-Time Operating System. The DUT communicates with the user through a VGA/keyboard interface and a serial port to the test controller system. All telemetry is echoed to a memory area that is not destroyed upon soft-reset or short power cycles. This memory is dumped through the serial port to the test

controller after a reboot following a crash of the system. This is performed in an attempt to recover information lost due to the crash.

The P3 DUT Software is designed to execute one thread at a time with no preemptive task switching. This is accomplished in the following statement:

```
EtsSetTimeSlice(0);          // disable preemption
```

There is an array (radTests) in the main thread that contains information about the various tests, including the test letter, the test description, a pointer to the test's executable, a repeat count, a pointer to memory block for the test to use and its size. The repeat counts are currently always -1 indicating that the test should be repeated until it is terminated by an external command.

When the DUT Software begins execution, it creates a menu using the test letter and description from the radTests array. It also adds three hardcoded selections: "@: Toggle cache enable", "L: Loop through all tests", and "N: Loop through all tests continuously".

The first hardcoded selection switches the cache state between four states: "Disabled", "Enabled - L1 Data", "Enabled - L1 & L2 data", and "Enabled - All". There is a memory allocation (malloc) statement at the start of the main program that forces all of the test memory into a different page than the program executable:

```
void *fillA=malloc(0x800000),*fillB;
```

The cache can then be enabled or disabled on these pages separately. Model Specific Registers (MSRs) in the Pentium are used to achieve this. MSR BBL_CR_CTL3 is used to enable/disable the L2 cache and MSRs MTRRphysBase0 and MTRRphysMask0 are used to enable/disable the code page cache.

The second hardcoded selection, "L", executes each test except "M" in the radTests array two times and then returns to the menu.

The third hardcoded selection, "N", executes "L" continuously pausing once every ten minutes for the test controller to take voltage/current measurements. When pausing the DUT Software outputs "MEASURE …" to the test controller, waits 30 seconds, and then outputs "END …" to the test controller.

The tests contained in the radTests array are executed by starting a new thread at the location pointed to in the array. The main thread turns over execution to this thread and does not execute again until the test is terminated and the test thread returns execution to the main thread.

The first thing that each test does is to call MarkExceptionReturn(). This subroutine saves the execution location and stack information as of this call. Any exceptions encountered while executing the test cause a jump to this location after reporting the error information. The standard action is to set the restart flag to 1 and then terminate the thread. After the thread is terminated, the main thread will continue to execute, check the restart flag, and, if the restart flag is non-zero, will restart the test.

Three types of errors are encountered when executing the test software. Test Specific Errors are errors that the software is designed to look for based on the function it is testing (Desired). Program Exceptions are errors that cause the CPU to stop executing the code immediately. Terminal Errors are those that terminate a test. At the end of this document is a chart that shows possible reported errors.

Test Descriptions

There are eight tests available for testing the Pentium III components. Each test sends a keep-alive at approximately a one hertz rate (0.1 Hz for test "H") and sends error results as they happen. One test is selected for each exposure to the beam and the software repeatedly performs the test until a pre-determined dose (or fluence) is reached or the software stops communicating to the user.

Test "A" checks the eight general-purpose registers of the CPU. Three registers are used by the program to keep track of execution parameters. The ESI register is used to point to a data area in memory where the values of the five registers under test are mirrored. The EAX register is used to keep track of the number of executions so that the keep-alive can be sent at proper intervals. The ESP register keeps track of the stack, which is used to store information when logging errors.

This test sets five CPU registers (EBX, ECX, EDX, EBP, and EDI) to a baseline value of 0AAAAAAAA Hex, then continuously checks to see if any of the register values change. If any values change, an error is reported to the user and an attempt is made to reset the register to its baseline value. The register is read again to form a new baseline value. The error report includes the following: the name of the register that changed, the value it changed to, the baseline before the error and the baseline after the error. The test then continues. At each keep-alive, the baselines are reset to 0AAAAAAAA Hex.

This test is designed to catch four error types: "Bit error: 1 changed to 0", "Bit error: 0 changed to 1", "Miscompare: data different by >1 bit", and "Miscompare: data identical". At TAMU the first two error types were reported as "Miscompare: data different by 1 bit". To determine how to classify this error the telemetry is examined to compare the values read and expected. The following example is from test "A" telemetry output from March 2001 at TAMU:

```
A.A.A.A.A.A.A.A.A.A.A.A.A.A.
FAIL Sat Mar 24 03:44:14 2001

00000004
AAAAAABA
AAAAAAAA
AAAAAAAA
END
A.A.A.A.A.A.A.A.A.A.A.A.A.A.
```

The first line after the FAIL statement indicates that the error occurred in the ECX register. The second line is the value read from the ECX register after the miscompare. The third line is the expected value for ECX and the fourth line is the value of ECX after trying to reload ECX with the expected value. In this case the value read from ECX has a bit that is 1 where a 0 was expected. This error was classified as "Miscompare (data different by 1 bit)". If the second line were "FFFFFFFF" then it would be classified as "Miscompare (data different by >1 bit)".

In test "A", an error in one of the registers is expected to result in a "Bit error: 1 changed to 0" or a "Bit error: 0 changed to 1". A hit that causes a register to be reset is expected to result in a "Miscompare: data different by >1 bit". An error in performing the compare operation is expected to result in a "Miscompare: data identical".

Test "B" checks the Floating Point Unit (FPU) with a maximum of data transfer to the FPU. A buffer is loaded with the arguments and expected results for the five operations tested (fadd, fsub, fmul, fdiv, and fsqrt). For each cycle through the test, the following sequence is followed: The first operand is transferred to the FPU from memory. The second operand is transferred to the FPU from memory. The operation is performed. The result is transferred to memory. The result is transferred back to the FPU and compared with the expected result. Any errors are saved to a log in memory and reported to the user at the next keep-alive. The EBX is used to keep track of the number of cycles run. When EBX passes a constant number of cycles a keep-alive is sent and any errors are reported.

An error consists of three quadwords: The first is the function identifier (Hex 0:fadd 18:fsub 38:fmul 48:fdiv 60:fsqrt), The second is the result of the operation. The third is the expected result.

This test is designed to catch four types of errors: "Bit error: 1 changed to 0", "Bit error: 0 changed to 1", "Miscompare: data different by >1 bit", and "Miscompare: data identical". The classification is determined in the same manner as that of test "A". An error in the performance of an operation, the transfer of operands, or a bit change in the operand registers is expected to result in a "Miscompare: data different by >1 bit". An error in the transfer of the result or a bit change in the result register is expected to result in a "Bit error: 1 changed to 0" or a "Bit error: 0 changed to 1". An error performing the compare function is expected to result in a "Miscompare: data identical".

Test "C" performs a memory test or cache test. If the cache is turned off, it performs a memory test otherwise, it performs a cache test. If the cache is on, then the cache is turned off, the memory is loaded with an incrementing pattern and the cache is turned back on before entering the test. The entire range is loaded with a baseline of 0AAAAAAAAH. The range is 131072 words for the memory test and the L1&L2 cache test and 8192 words for the L1 only test. The memory is checked word by word. After each word is checked, its value is changed to the bitwise complement of the baseline. If the value is not as expected then an error is reported and an attempt is made to reset the value to the baseline. The error report includes: the location address, the value read, the baseline and the value after attempting to reset to the baseline. If the value cannot be restored to the baseline, then checking is disabled for that location until the next keep-alive. After the entire range is checked, the baseline is changed to its bitwise complement and checking starts again from the beginning.

Two types of errors occur during the cache tests. Tag errors cause 16 consecutive words to be in error and bit errors cause one or two bits to be in error in a word. We have seen three tag error scenarios. The first is that a tag miss occurs and the pattern from memory is read. The second is that the wrong baseline is read. In addition, the third is that a random pattern is read. The output from the first two scenarios has been compressed so that there is one line output that describes what happened for all 16 words. Most tag errors occur once but we have seen some tag errors recur several consecutive times within a run when testing with L1&L2 cache on.

When the cache test is started, the user is prompted for a test size. The available sizes are 100%, 50%, 25%, and 1%. The actual size of the test is computed as full cache (256K for L2 and 16K for L1) times this percentage. When testing the L2 cache, 1% is not an option since this would reduce the size such that only the L1 cache is utilized.

Because of the high number of bits checked in the cache test, it has the high number of reported errors. The number of bits tested in the L1&L2 cache test is 2097152*x% data bits plus 180224*x% tag bits (2277376*x% bits). This test is designed to test the L2 cache

by always retiring data in the L1 cache when it overflows. The data flows through the L1 cache. The data is transferred to the L1 cache across a 64-bit data bus. The data then stays in the L1 cache for an average of .008% of the time it spent in the L2 cache. The L1&L2 cache test tests the entire bit range in 0.464/x%-0.928/x% seconds on a 1GHz part and multiplied by x/1GHz for an x GHz part.

The number of bits tested in the L1 cache test is 131072*x% data bits plus 12800*x% tag bits (143872*x% bits). This test tests the entire bit range in 0.0288/x%-0.0577/x% seconds on a 1GHz part and multiplied by x/1GHz for an x GHz part.

In all cache tests, the data is transferred to the execution unit from the L1 cache across a 64-bit bus. In order to determine whether transfer errors are occurring as opposed to bit flips in storage, we shall add cache tests that test smaller areas of the cache. We will find error rates at three different sizes for each cache. From this data, we will determine the part of the rate that is independent of the size (transfer errors) and the part that is proportional to the size (storage errors).

For test "C" May IU data, there were two possible classifications for a memory/cache error: "memory error", "recurring memory error". To determine how to classify this error the telemetry was examined for repeating entries. If an error repeated more than 50 times between keep-alives it was classified as "recurring memory error". All other errors were classified as "memory error".

At the October TAMU, improvements were made to test "C" to give more visibility into memory/cache errors and to lessen the number of errors from instruction cache hits. The software was placed in a non-cacheable segment of memory and the telemetry output was expanded to include the location and contents of memory found in error. Also test "C" was changed from a process of loading all memory, checking all memory, loading all memory… to a process of checking one location loading that location checking the next location… This allowed the data to sit in its location (cache or memory) for the entire time cycle between checks. In the previous version, the data was loaded at about the halfway point between checks.

In addition, at TAMU, some data output reduction was added. If several consecutive locations read the wrong baseline or the ram pattern (for cache tests), then the telemetry output was reduced to one error message followed by "repeated xx times". Most tag errors now show up as follows (taken from May 2001 IU data):

FAIL Thu May 24 22:05:57 2001
memory test error 01750 1750 repeated 16 times END

For test "C" IU data, there are four possible classifications for a memory/cache error: "recurring memory error", "tag error", "Bit error: 1 changed to 0", "Bit error: 0 changed to 1". If an error repeated more than 50 times or in more than 50 consecutive locations, it was classified as "recurring memory error". If an error appeared in 16 consecutive locations (1 line in the cache), it was classified as "tag error". If an error showed that a bit was 0 but was expected to be 1 then it was classified as "Bit error: 1 changed to 0". If an error showed that a bit was 1 but was expected to be 0 then it was classified as "Bit error: 0 changed to 1". There were several cases where one line of telemetry showed more than one bit error or tag error as in the following example:

FAIL Wed May 23 04:51:55 2001
memory test error 149EE 08AEA 0AAAA 0AAAA END
FAIL Wed May 23 23:37:36 2001

memory test error 02010 2010 repeated 32 times END

The first report shows 2 bit errors and the second shows 2 tag errors.

Test "D" launches seven subthreads each with a counter that is reset to zero. Each thread increments its counter if the counter is less than 11 and then passes control to the next thread. The main thread then checks, after 50 milliseconds, to see if all of the counters have reached 11. If not an error is reported to the user. The test repeats continuously.

The "Task Switch Error" is the only error that this test is designed to catch. This error will occur if any of the subthreads stop executing.

Test "E" runs through 16K of instructions repeatedly. The instruction sequence is to increment the eax register from 0 to 3 checking in between each increment to see if the value is as expected, then to decrement the eax register 3 times and check to make sure it returns to zero. Any errors are reported to the user and the cache is invalidated. Thirty of forty-six bytes in the code are continually used during proper execution; 65% of the 16K are exercised. The ECX register is used to keep track of the number of times the instructions are repeated. When ECX passes a constant number of cycles, a keep-alive is sent. The cache is flushed and the test is restarted whenever an error is detected.

Because the test is extremely sensitive to the value in EAX, the test will report a cache error as soon as the proper sequence is interrupted. The following example shows how the program responds to such interruption:

This is the error free operation of the test "E" (eax starts at 0):

```
40              inc eax
3D01000000      cmp eax,1
0F85E03F0000    jnz erout       ;jump not taken
40              inc eax
3D02000000      cmp eax,2
0F85D43F0000    jnz erout
40              inc eax
3D02000000      cmp eax,3
...
```

The following two scenarios show how one bit change can affect the operation of test "E" (eax starts at 0):

```
41              inc ecx
3D01000000      cmp eax,1
0F85E03F0000    jnz erout       ;jump taken
...
003D0100000F    add [ecx][eax]+0F000000H,bh
85E0            test esp,eax
3F              aas             ;adjust al after subtract
0000            add [eax],al
40              inc eax
3D02000000      cmp eax,2
0F85D43F0000    jnz erout       ;jump taken
…
```

This test is designed to report only one type of error: "cache error". Any sequence that causes a jump to "erout" will report a "cache error".

Test "F" checks the Floating Point Unit (FPU) with a maximum of operations in the FPU. A buffer is loaded with the arguments and expected result. The operation tested is:

cos(cos(cos(sin(sin(sin(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(a*b))))))))))))

where a=0.123456789 and b=0.987654321. For each cycle through the test, the following sequence is followed: The first operand is transferred to the FPU from memory. The second operand is transferred to the FPU from memory. The operations are performed. The result is transferred to memory. The result is transferred back to the FPU and compared with the expected result. Any errors are saved to a log in memory and reported to the user at the next keep-alive. The EBX is used to keep track of the number of cycles run. When EBX passes a constant number of cycles a keep-alive is sent and any errors are reported.

An error consists of three quadwords: The first is the function identifier (Hex 78), the second is the result of the operation, and the third is the expected result.

This test has the same expected errors and causes as test "B". However, since there are fewer transfers and more time spent with data sitting in the FPU, the frequency of errors should be different.

Test "G" checks the Matrix Math Extensions (MMX). A buffer is loaded with the arguments and expected results for the four operations tested (pxor, por, pmul, pmulh, padds, addps, divps, and mulps). For each cycle through the test, the following sequence is followed for each operation: The first operand is transferred to the FPU from memory (FPU registers are used by the Pentium for performing MMX functions). The second operand is transferred to the FPU from memory. The operation is performed. The result is transferred to memory. The result is compared with the expected result. Any errors are saved to a log in memory and reported to the user at the next keep-alive. The EBX is used to keep track of the number of cycles run. When EBX passes a constant number of cycles a keep-alive is sent and any errors are reported.

An error consists of three quadwords or five quadwords depending on the function: The first is the function identifier (0:pxor 1:por 2:pmul 3:pmulh 4:padds, 5:addps, 6:divps, and 7:mulps). For functions 0 through 4, the second quadword is the result of the operation and the third is the expected result. For functions 5 through 7, the second and third quadwords are the result of the operation and the fourth and fifth quadwords are the expected result.

There are four errors that this test is designed to catch: "Bit error: 1 changed to 0", "Bit error: 0 changed to 1", "Miscompare: data different by >1 bit", and "Miscompare: data identical". The classification is determined in the same manner as that of test "A". An error in the performance of an operation, the transfer of operands, or a bit change in the operand registers is expected to result in a "Miscompare: data different by >1 bit". An error in the transfer of the result or a bit change in the result register is expected to result in a "Bit error: 1 changed to 0" or a "Bit error: 0 changed to 1". An error performing the compare function is expected to result in a "Miscompare: data identical".

Test "H" measures the passage of time in CPU cycles against the ISA bus clock of 8.33MHZ. A timer board has been set up to provide an interrupt every 16383 cycles of the ISA bus clock. 5000 samples of the number of CPU cycles between interrupts are

recorded and sent to the user after the test is complete in about 9.8 seconds. Then another test is started.

This test outputs a distribution of the number of CPU cycles between interrupts. The format of this output is as follows (each entry is in hex followed by a comma):

- the number of cycles for the first value in the list
- the number of samples below the lowest value in the list
- the number of samples beyond the highest value in the list
- the increment in number of cycles between entries in the list
- the list of values (comma separated) where a series of zeros is output as 0:v (v being the hex number of times to repeat the 0 entry). There is no comma after the last value in the list.
- The keep-alive: "H"

Exception Errors

In addition to the errors that the tests are designed to catch, there are several errors that can occur in any of the tests. The most common of these is the "General Protection Fault" (GPF). This error occurs when the program is not executing as designed. There are several events that could cause this:

- The Global Descriptor Table could have been corrupted. We now run all tests with the GDT stored in un-cacheable memory to minimize the occurrence of a corrupt GDT.
- The instruction could have been corrupted. We now run all tests with instructions stored in un-cacheable memory to minimize the occurrence of an unintended instruction being executed.
- A register could have been corrupted.
- The Arithmetic Unit could have computed an illegal address.

Other errors that can occur in any test are: illegal instruction, divide-by-zero exception, debug exception, non-maskable interrupt, one byte interrupt (INT3), interrupt on overflow, bound interrupt, device not available exception, double fault, coprocessor segment overrun, invalid TSS, invalid segment exception, stack fault, page fault, coprocessor error, hang, bomb, self reset. The last three of these errors are classifications that are determined by the manner in which a test terminates. A hang is the classification for when the DUT stops sending keep-alives and stops responding to the user. A bomb is the classification for when the DUT display is loaded with a random pattern. A self-reset is for when the DUT reboots from the disk.

Any of the exceptions can be caused by an unintended instruction sequence due to a corrupt instruction or the execution of instructions from the wrong location (corrupt instruction pointer, segment register or GDT). This is very unlikely however since the instruction sequence for the exception is one in 65536 possible instruction codes.

The "illegal instruction" exception can also be caused by execution of an illegal instruction but is less likely because most of the values are valid instructions and executing an unintended instruction is more likely to end up in a GPF.

The "divide-by-zero exception" can also be caused by a divide by zero but is very unlikely because the event would have to load a zero into the divisor immediately before a divide instruction or an unintended divide would have to be executed with a

divisor value of zero. Test "H" is designed to perform two divides per keep-alive. Test "A","B","F", and "G" are designed to perform only one divide per keep-alive. Test "C","D","E" are designed not to do any divides.

The "Debug exception" has no additional causes.

The "non-maskable interrupt" can also be caused by the NMI input signal being set.

The "one byte interrupt (INT3)" can also be caused by an INT3 instruction but is less likely because there is one code in 256 that performs the instruction and it would have to be an unintended instruction.

The "interrupt on overflow" can also be caused by an INTO instruction but is less likely because there is one code in 256 that performs the instruction that triggers this and the overflow flag must be set when it is executed.

The "BOUND interrupt" can also be caused by a BOUND instruction but it is less likely because there is one code in 256 that performs a boundary check and it would have to be an unintended instruction. In addition, the actual boundary check would have to fail.

The "Device not available exception" can also be caused by a failure in detecting the math coprocessor. This would have to be a transient that occurs at precisely the same time that the CPU is trying to detect the coprocessor.

The "Double fault" can also be caused by an exception in the exception handler. Since we are using the same exception handler for the "double fault", the exception in the exception handler is likely to cause an infinite number of calls to the handler and either continuously report an incomplete failure message or hang. The following combinations can cause a "Double fault":

1. Contributory Exception followed by Contributory Exception.
2. Stack Fault followed by either a Contributory Exception or a Stack Fault.

Contributory Exceptions	divide-by-zero exception
	Invalid TSS
	Segment not present exception
	Stack Fault
	General Protection Fault

The "Coprocessor segment overrun" has no additional causes.

A corrupt GDT or TSS can also cause the "Invalid TSS" or "Segment not present exception". Since these are stored in memory, it is less likely to occur.

The "Stack Fault" can also be caused by a corrupt GDT, TSS, stack segment register or stack pointer. This is more likely than "Invalid TSS" and "Segment not present exception" because if the stack is accessed outside of valid memory this fault is generated instead of the GPF that is generated for other segments.

A corrupt GDT or TSS can also cause the "Page Fault". There are three causes of a page fault: Writing to a read only page, Accessing a page which is not in physical memory, or Accessing a page to which the current program does not have access. Additional information is present in the error code:

- Bit 0 indicates whether (0) the page was not present or (1) there was an access rights violation or use of a reserved bit.
- Bit 1 indicates whether (0) the program was attempting a read or (1) the program was attempting a write.

- Bit 2 indicates whether (0) the program was executing in supervisor mode or (1) the program was executing in user mode.
- Bit 3 indicates whether (0) the fault was not caused by a reserved bit violation or (1) the fault was caused by a reserved bit violation.

At IU, we recorded one "Page Fault". It was caused by a write to a page that was not present.

The "coprocessor error" has no additional causes because the coprocessor is disabled from generating this error.

IV. Testing Methodologies

A. TID Test Process

To completely characterize the P3/K7 DUT for TID effects requires numerous parametric measurements, too numerous to measure without specialized test equipment beyond the scope of this project. This level of characterization is not necessary for the needs of this project. To this end, it is sufficient to monitor the voltages and currents to the microprocessor, the instruction timing (to monitor the processor for timing critical operations), and microprocessor functionality.

This total ionizing dose response was measured for both protons and Cobalt-60. Proton testing was carried out at the Indiana University Cyclotron Facility (IUCF) using 198 MeV protons incident on the test structure with fluxes ranging from 10^6 to 10^9 protons/cm^2/sec. This proton TID testing was done in conjunction with the proton SEE testing. Cobalt-60 testing was done at the GSFC Radiation Effects Facility (REF) with dose rates ranging from 3 to 10 krads(Si)/day.

The same test hardware used for SEE testing was used to accomplish the measurements mentioned above. For biased testing, the entire motherboard assembly is placed in the Cobalt chamber with all but the DUT heavily shielded. For unbiased testing, the DUT is placed in a test jig with grounded pins and removed from the chamber periodically into the full test setup for data collection.

Data collection consists firstly of the seven voltages and currents available through the extender card. Secondly, the DUT is fully exercised utilizing all the tests developed for SEE testing. Finally timing is monitored using a measurement of the access time of a data line and through the use of the Timer Card hardware and software.

B. Single Event Effects Test Process

The SEE test process includes methods to test for all aspects of single event effects (latchup, functional interrupts, upsets, etc.). As a number of these effects are sensitive to the software being run and may be sensitive to numerous other conditions, detailed control of the Device Under Test is required. To this end, an extensive operating system would serve no purpose. Therefore, testing is done with a minimal operating system and a test executive. This is to allow for low-level testing of sections of the processor.

Testing uses various software routines to isolate sections of the microprocessor (e.g., registers, cache memory, floating–point and MMX units). Additionally, numerous processor speeds are tested in an attempt to investigate frequency dependent events. This is done using processors of various rated clock speeds and running these processors at a lower than rated Front Side Bus (FSB) frequency.

The main part of the test flow is placing the DUT in a known operating state, waiting for something to happen and then dealing with it. Figure 2 shows a flowchart of the methodology used once an event happens. The flowchart shows that five event types are expected: end of test fluence, single event functional interrupt (SEFI), system resets (radiation induced), single event latchup (SEL), and non-fatal errors (some error is produced but it does not immediately induce a functional interrupt or system reset). Once one of these conditions is seen and identified, the test goal is to gain information about what exactly happened and to recover the DUT to a known state.

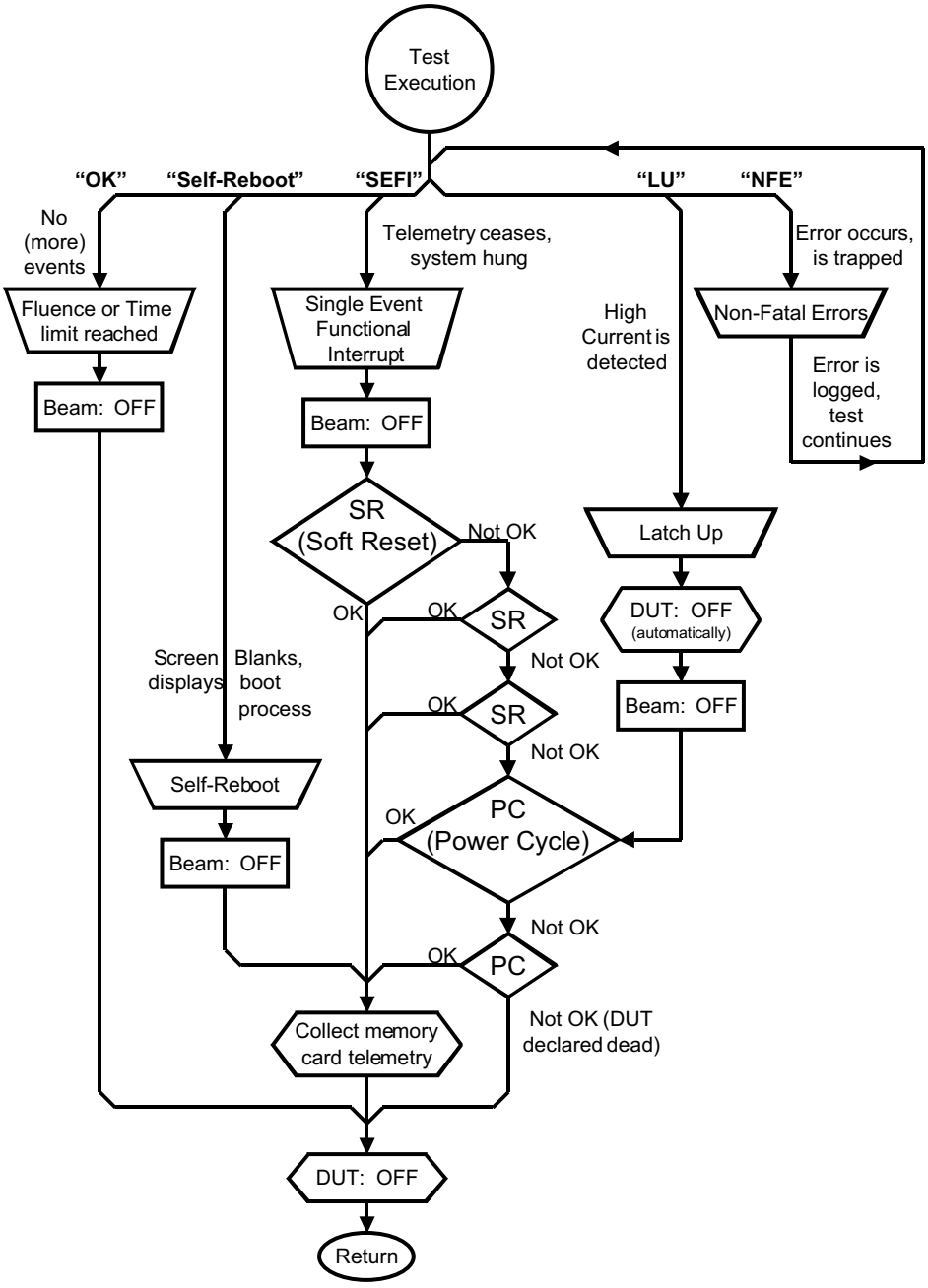

Figure 2. SEE Testing flowchart showing the expected possible outcomes from a Single Event and the methodology used in dealing with these events.

It is important to understand the definition of a functional interrupt as used here. The main routine of the test executive software was written to handle all the exceptions the processor could throw. So, in addition to looking for errors produced in the specific test being run, the test software handles any other generic exception. In general, if the exception can be handled (after the exception is recorded the process restarts correctly) it is classified as a non-fatal error (NFE). It should be noted that if these exception-handling routines were not present these events would result in the test software being halted. So, within this work, a SEFI is an event that causes the processor to lock up, reset, have continuous exceptions, or go into some unknown, unrecognizable state.

C. Thermal Considerations

When operating under normal conditions with the caches enabled, the Pentium III draws in excess of 20 watts of power. If left in that state with no cooling, the processor will not even boot.

Also, The Pentium III and AMD K7 die, as procured as a COTS parts, are flip chip solder bubble bonded die to the DUT daughter cards. Since the beam must hit the die directly, the packaged heat sink and cooling fan must be removed. In its place a water-cooled jacket, which is thinned to 10 mils over the die, is used.

The large thermal source is also the reason that the die cannot be thinned, as has been done with other flip chip parts. The thick substrate is the main thermal path for removal of heat from the junctions. Thinning this would place excessive thermal stresses on the die and most likely lead to structural failures.

D. Die Thickness Issue

As pointed out above, these parts are flip chip solder bubble bonded die. This places the sensitive regions of the processor approximately 900 microns deep in the silicon die with respect to the heavy-ion incidence point (See Figure 3). Thermal issues compound this by requiring cooling material in the beam line, as well. Therefore, only high energy and high Z beams are capable of penetrating and giving higher Linear Energy Transfer (LET) values in the sensitive regions.

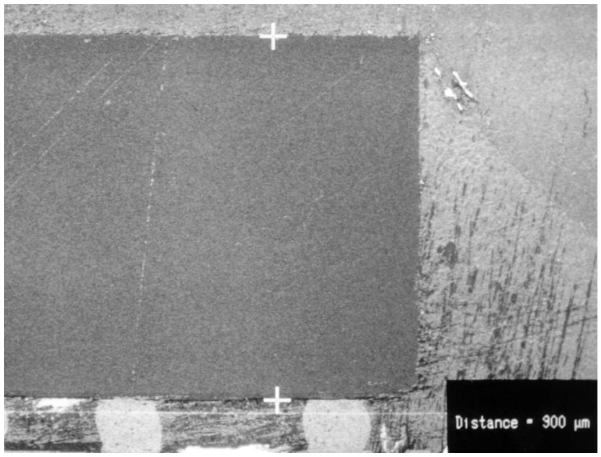

Figure 3. Photograph showing a close up cross section of the P3 die. Depicted here is the die thickness (900 microns) between the backside of the die and the front edge where the solder bubbles are clearly visible.

V. Data Analysis Software

The data sets generated from single event testing contain enormous amounts of data due to the large test matrix. It was necessary to generate software to deal with these large data sets and to analyze the data under all test conditions. A database form was chosen for this analysis. The database initially reads all of the test conditions, then allows to user to scan through the telemetry files. The user marks locations within the telemetry files with error annotations that are then stored in the database with those associated test conditions.

After each telemetry file has been annotated, the database can be queried via Structured Query Language (SQL) commands to extract only those conditions to be analyzed. Depending on the detail of the SQL commands, either the event rate (or cross-section) can be calculated directly or the selected data from the database can be exported in tabular format for other software to continue the analysis.

Figure 4 and Figure 5 show a sample screen shot for the telemetry file analysis and the SQL data extraction process, respectively.

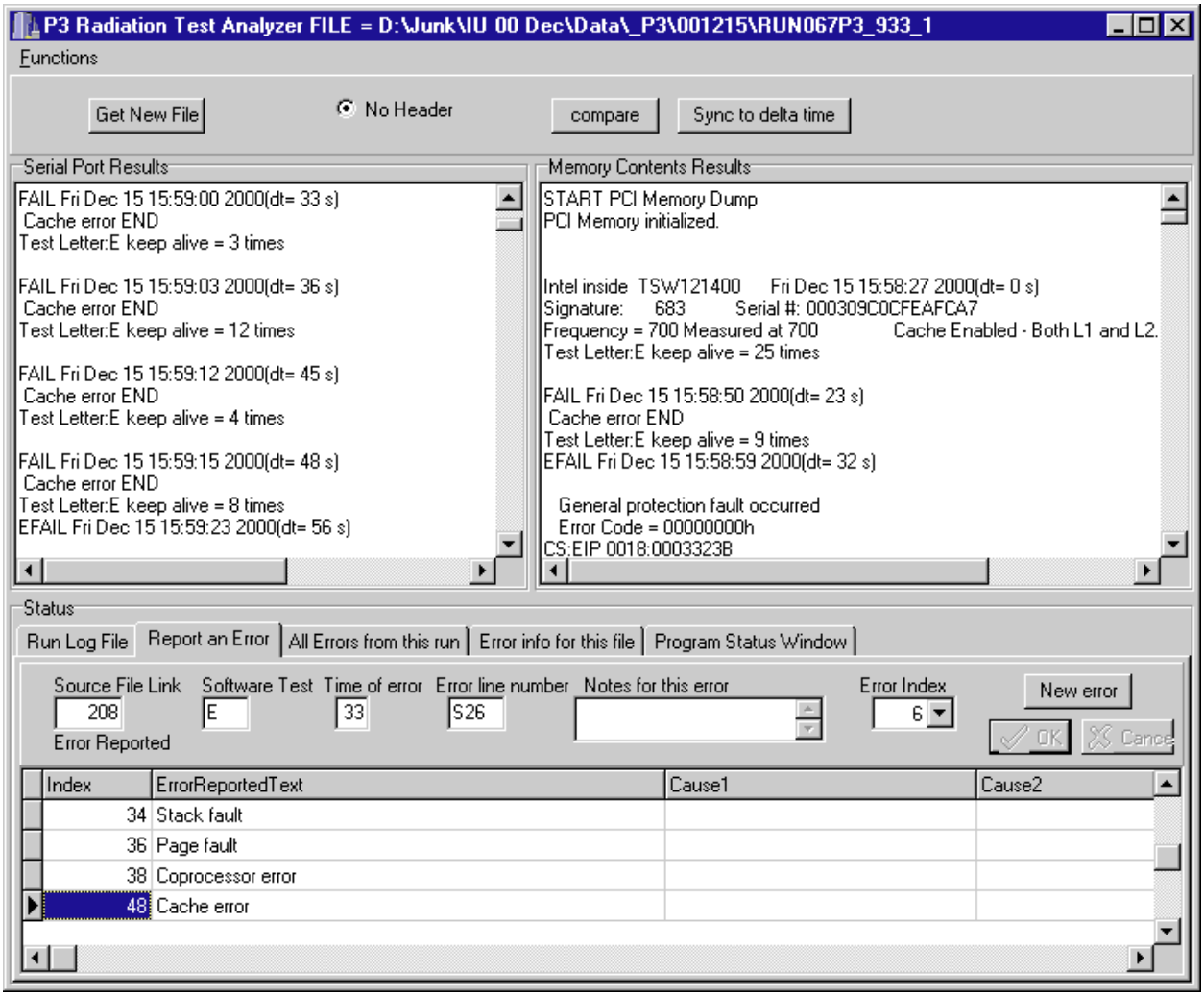

Figure 4. Screen shot of the telemetry analysis software.

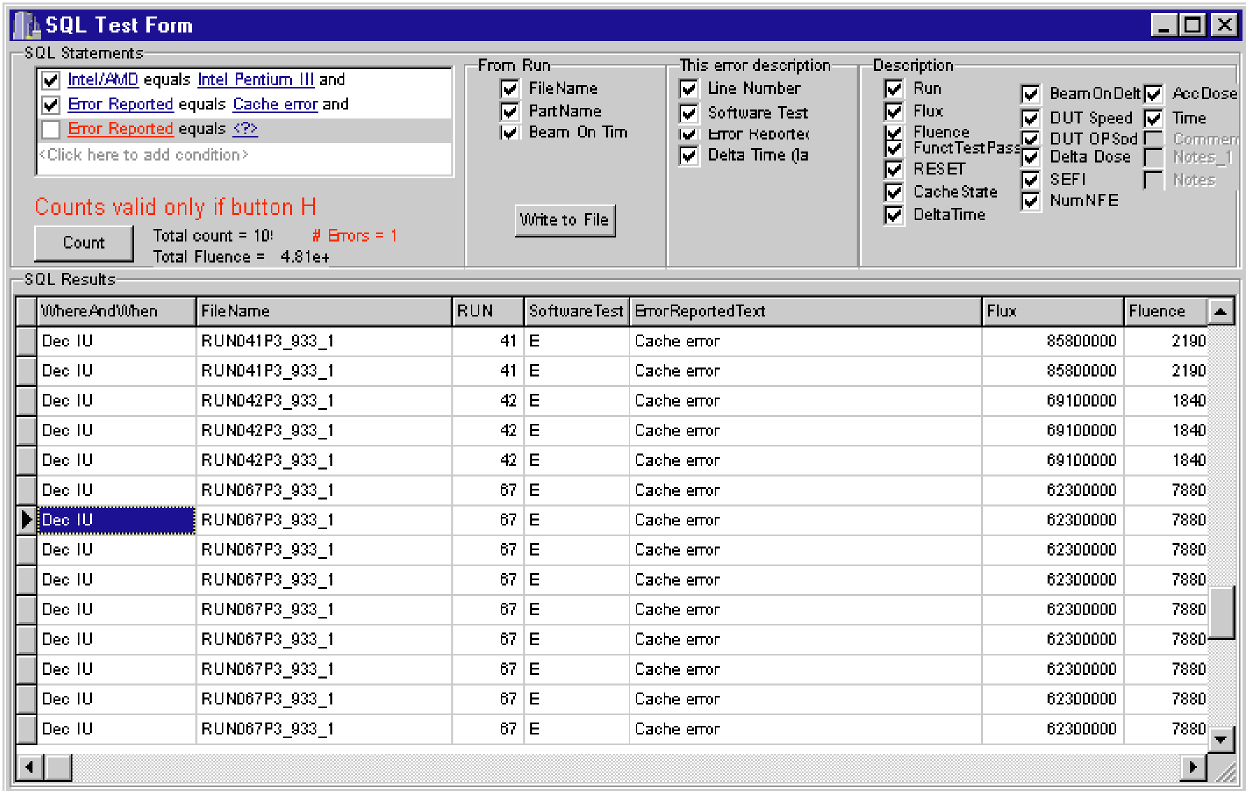

Figure 5. Screen shot of the SQL data extraction software.

VI. Results

A. *Total Ionizing Dose (TID)*

Intel P3 and AMD K7 parts were exposed to the total dose environment at the IUCF proton facility and the GSFC Radiation Effects Facility (Cobalt-60). The results of this testing are summarized in Table IV for the P3 devices and Table V for the K7 devices. It should be noted that the one DUT rated at 550 MHz is 0.25 μm technology, while all other DUTs tested are 0.18 μm technology.

The parts in unbiased and biased states tested at IUCF were exposed to proton doses with various increments up to approximately 100 krads(Si). After each dose, all parts passed all functional tests and the monitored voltages and currents did not change. No parametric timing measurements were done in these tests. These tests, if possible, would be expected to be sensitive to dose. The parts, however, did not degrade in timing sufficiently to fail any of the functional tests that were performed.

Total dose testing using the Cobalt-60 source at GSFC was stopped on 3/8/02, due to the GSFC Facility shutting down for maintenance. Several P3 devices, in an unbiased condition, have been exposed to various doses, one in excess of 3700 krads(Si). They have shown little sign of degradation in either supply currents or timing and functionality testing. Biased testing of one Pentium III DUT did functionally fail after exposure to an approximate dose of 511 krads(Si). A replacement part was tested under bias, and had exceeded a dose of 573 krads(Si) when it was finally removed.

TABLE IV
Pentium III DEVICE UNDER TEST (DUT) TABLE

Device	Rated Speed	Test Condition	Source	Exposure Levels (krads)
P3	800 MHz	Biased	Co-60	*511
P3	933 MHz	Biased	Co-60	573
P3	550 MHz	Unbiased	Co-60	336
P3	650 MHz	Unbiased	Co-60	336
P3	650 MHz	Unbiased	Co-60	3700
P3	700 MHz	Unbiased	Co-60	336
P3	850 MHz	Unbiased	Co-60	697
P3	933 MHz	Unbiased	Co-60	2100
P3	550 MHz	Biased	Protons	4.9
P3	650 MHz	Biased	Protons	52
P3	650 MHz	Biased	Protons	4.4
P3	700 MHz	Biased	Protons	100
P3	700 MHz	Biased	Protons	5.5
P3	700 MHz	Biased	Protons	8.9
P3	750 MHz	Biased	Protons	14.3
P3	850 MHz	Biased	Protons	0.3
P3	850 MHz	Biased	Protons	13.3
P3	850 MHz	Biased	Protons	47.5
P3	933 MHz	Biased	Protons	16.9
P3	933 MHz	Biased	Protons	46.7
P3	933 MHz	Biased	Protons	34.4
P3	1 GHz	Biased	Protons	45.3
P3	1 GHz	Biased	Protons	32.5
P3	650 MHz	Unbiased	Protons	26

** Functional Failure Dose.*

TABLE V
AMD K7 DEVICE UNDER TEST (DUT) TABLE

Device	Rated Speed	Test Condition	Source	Exposure Levels (krads)
K7	600 MHz	Biased	Protons	4.4
K7	650 MHz	Biased	Protons	3.6
K7	700 MHz	Biased	Protons	0.5
K7	900 MHz	Biased	Protons	7.2
K7	1 GHz	Biased	Protons	3.2
K7	1 GHz	Biased	Protons	0.1
K7	650 MHz	Unbiased	Protons	100

There were plans to expose the AMD K7 processors, both biased and unbiased, to Cobalt-60. However, after the poor showing the parts made at the heavy-ion facility (see next section), it was determined to remove the AMD K7 parts from this study.

The limited proton TID testing seems to indicate that these generations of AMD K7 processors are TID hard to greater than 100 krad(Si). However, based on substantial data collected via proton and Cobalt-60 exposure, the Intel Pentium III processors are extremely tolerant to total dose, with unbiased parts surviving in excess of 3.7 Mrads(Si) and biased parts surviving in excess of 500 krads(Si).

B. Single Event Effects

Both P3 and K7 processors were evaluated for SEE response. This included exposure to protons at the IUCF and heavy ions at Texas A&M University (TAMU) Cyclotron. The result summaries from the test reports are reported here.

Protons

Single Event Latchup

Sixteen different P3 processors (one 550 MHz, three 650 Mhz, three 700 Mhz, one 750 Mhz, three 850 MHz, one 933 MHz in SC242, two 933 MHz in PGA370, and two 1 GHz in PGA370) were run at fourteen different clock speeds. Seven different AMD K7 processors (one 600 Mhz, two 650 MHz, one 700 MHz, one 900 Mhz, and two 1 GHz) were run at five different clock speeds. During these tests, the processors were running one of the tests in the test executive (tests were varied) and exposed to proton fluences (per run) that varied from 1.5×10^5 to 3.3×10^{11} protons/cm^2. The parts were tested in 683 conditions (processor speed, various cache on/off, ECC on/off, and software executing). In all of the testing, no evidence of latchup was observed (presence of latchup would be indicated by a sharp increase in I_Vcc_core). There were two of the 883 cases that required a hard boot (power cycle) to resume normal operations.

Single Event Functional Interrupts (SEFI)

Figure 6 shows the Pentium III (P3) SEFI cross-sections as a function of the DUT operating speed and cache state. The curves in blue are the "cache on" data and the curves in red are for "cache off." The solid symbols are data from the first complete test run (proton test #2). The plus and star symbols are from the third proton test run. The solid symbol data show a rather dramatic difference between cache on and off. The third proton test (with the better DUT software) still shows a difference but not as dramatic and not as a large a cross-section for the "cache on" condition. This third data set possibly shows a slight speed dependence, but the variation in the data does not allow that to be said with any confidence.

Figure 7 shows the same cache on/off data from the second proton test run for the AMD K7 parts. Again, the data shows a higher sensitivity in the "cache on" case and no speed dependence.

Single Event Upsets – Non-SEFI

Next to be considered is the exception cross-section (i.e., those events, if not handled, would lead to a SEFI). Figure 8 shows the data for the exceptions as a function of the DUT operating speed and cache state. As above, the blue data is the cache on condition and the red data the cache off condition. The solid symbols are the older data and the plus and star data are the most recent data.

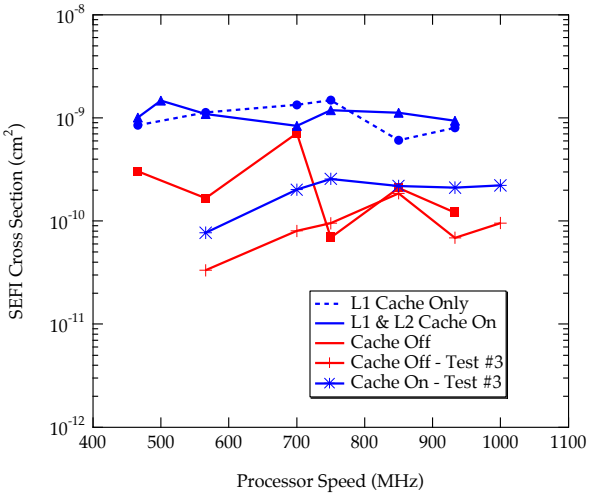

Figure 6. SEFI cross-section of the Pentium III processor as a function of the operating clock speed.

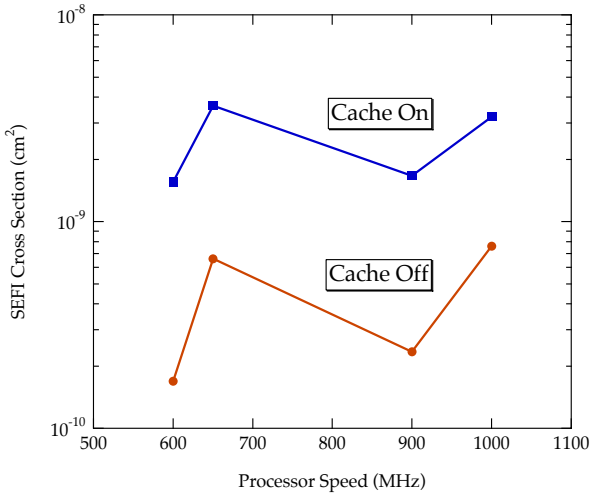

Figure 7. SEFI cross-section of the AMD K7 processor as a function of the operating clock speed.

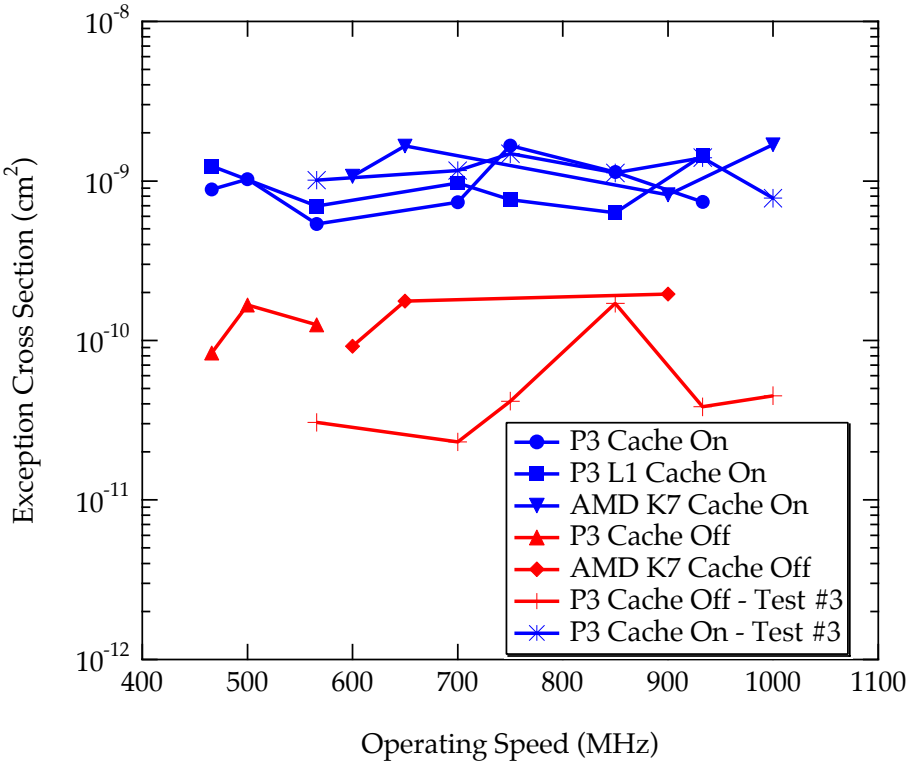

Figure 8. Exception cross-section of the Pentium III processors as a function of DUT operating speed.

The final data to be presented is for upsets based on tests A through G. This data is shown in Tables VI and VII.

The data shown in Table VI for test C shows that not only are the cache bits themselves sensitive to upset, but the cache tags are as well. Additionally, the cache tags

have an upset mode that appears to be multiple bit upsets in the tags. Of interesting note is that this mechanism is only present for the L1 & L2 Cache case. This case, as discussed in the software section, is a test of the L2 cache only. This seems to indicate that the L1 cache and L2 cache are not of the same technology or different architecture is employed when the L2 cache is enabled.

Another issue of concern is the difference in bit error rates in the cache that is not seen in the tags. For the per-bit tag errors for the L2 cache, upset rate is about one and a half to two times that for the L1 cache (more testing would be required to verify whether this factor is statistically significant). For the actual cache bits, though, the difference is on the order of 10 to 20 times. However, here the L1 cache is the 10 to 20 times more sensitive than the L2 cache. This same difference is observed in the heavy-ion testing that is discussed later. It turns out that the ratio of L2 bits tested (or at least thought were tested) to the 100% L1 bit size is about 12. If it were assumed that only 100% L1 bits were actually tested in L2, then the cache bit errors would be very similar between the L1 and L1 & L2 cases.

Investigation into architecture difference and possible software errors is still ongoing but it is possible that the differences are real and that the L2 cache is a different technology from the L1 cache and therefore may have a different upset rate. However, no discussion on the validity of the different cache claim can be made without knowledge of the hardware from Intel. This information was not immediately available and may not be available at all. Additionally, this upset rate difference can change with new technology devices that may change cache technologies as well.

TABLE VI
Per Bit Cache Cross-sections from Test C

DUT Speed	Cache State	Single Tag Errors	Multiple Tag Errors	Cache Bit Errors
1000	Off	0	0	0
1000	L1 Data	2.8×10^{-14}	0	3.88×10^{-14}
1000	L1 & L2	4.5×10^{-14}	9.36×10^{-17}	2.64×10^{-15}
933	Off	0	0	0
933	L1 Data	2.33×10^{-14}	0	4.07×10^{-14}
933	L1 & L2	3.39×10^{-14}	2.11×10^{-16}	2.43×10^{-15}
850	Off	0	0	0
850	L1 Data	1.9×10^{-14}	0	2.89×10^{-14}
850	L1 & L2	3.83×10^{-14}	3.54×10^{-16}	1.57×10^{-15}

The main item to be seen in Table VII is that very few events were actually observed, except for Test E. Collection of this data was problematic, as the SEFI rate was

sufficiently high as to impact the lengths of the runs. This data for all tests but test E (since test E tests the Instruction Cache, its operation is required) was taken with the Cache Off or the SEFI would have been too high to collect any significant data.

An important item to note is that the Test E cross-section shown in Table V is the per-bit cross-section and that it is very close to the per-bit cross-section for the L1 data cache from Test C. This result is consistent with the concept that both L1 caches should be the same technology and therefore have the same upset sensitivity.

TABLE V

Total Cross-sections for Other Tests

DUT	Test	Number of Upsets	Fluence (p/cm^2)	Cross-section (cm^2) (cm^2/bit) for Test E
P3	A	1	2.39×10^{11}	4.19×10^{-12}
P3	B	0	2.56×10^{11}	$< 3.91 \times 10^{-12}$
P3	D	1	3.96×10^{11}	2.52×10^{-12}
P3	E	562	3.22×10^{11}	1.33×10^{-14}
P3	F	6	3.92×10^{11}	1.53×10^{-11}
P3	G	4	2.62×10^{11}	1.53×10^{-11}

Heavy Ions

Single Event Latchup

Seven different P3 processors (one 800 Mhz, two 933 Mhz, two 933 Mhz Flip Chip and two 1 Ghz Flip Chip) and two K7 processors (one 650 MHz and one 1000 MHz) run at seven different clock speeds (Flip Chip P3 processors and K7 processors were only run at rated speed). During these tests, the processors were running one of the tests in the test executive (tests were varied) and exposed to heavy ion fluences (per run) that varied from 3.1×10^2 to 6.0×10^7 ion/cm^2. The P3 parts were tested in 600 different conditions (Linear Energy Transfer (LET), processor speed, cache on/off, and software executing) and the K7 parts were tested in 16 different conditions. In all of the testing, no evidence of latchup was observed (presence of latchup would be indicated by a sharp increase in I_Vcc_core).

It should also be mentioned that the K7 processors had a high current transient on the core power supply. While these transients were very high (tens of amperes in some cases), no destructive events were ever observed. In fact, the processors for most transients continued to work through a series of transients before a reset event would occur. It should be noted that the same high current transients observed in the proton testing on the K7 processors were observed with the heavy ions. They were observed on the same parts that demonstrated them in the proton testing and not seen on the same parts that did not demonstrate them with protons.

Single Event Functional Interrupts (SEFI)

Figure 9 shows the Pentium III (P3) and AMD K7 SEFI cross-sections as a function of the effective Linear Energy Transfer (LET). Figure 10 shows the Pentium data, as taken from Figure 9 (now in pluses and stars), and adds in the most current data set (shown with solid symbols).

The first observation to make from Figure 9 is the roll-off with LET. The K7 processor shows a significant roll-off after an LET of 15. It should be noted that the thickness was determined using a P3 part and the K7 part could have a substantially thicker substrate. The P3 parts do not show this roll-off but most likely would show a higher saturation cross-section if a longer range, high LET ion were used. As for LET threshold, every ion used in this test eventually led to a SEFI event. Therefore, the SEFI LET threshold is less than 0.7 MeV-cm^2/mg.

The next observation to make is the order of magnitude difference between the K7 and the P3 processors. This is also true for the most recent data set shown in Figure 10, where the Pentium data sets across the two test trips are shown to be consistent. The K7 difference is for the cache off condition only as the cache on condition for the K7 led to a single event induced reset as soon as the beam was turned on. This did not allow for an accurate determination of a SEFI cross-section for the K7 cache on state. For even the cache off state, the rate was high enough that the time to turn the beam off was a substantial error in the actual SEFI cross-section.

The significant part-to-part variation noted in the high current response, with the very high SEFI rate, led to the decision to remove the AMD K7 parts from further consideration (i.e., no additional SEE testing or any Cobalt-60 testing).

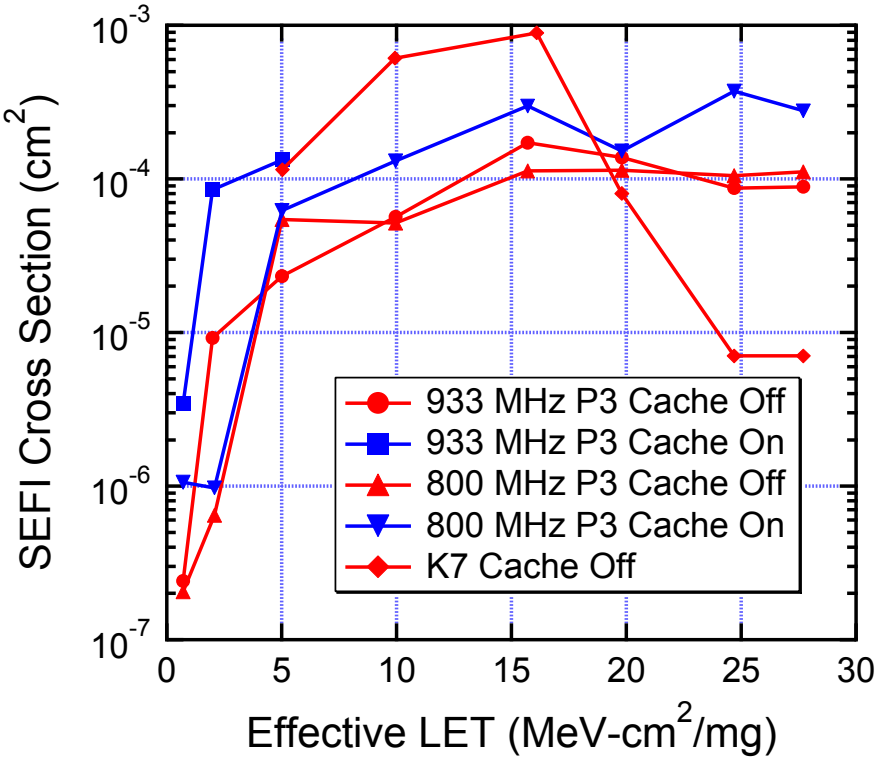

Figure 9. SEFI cross-section of the Pentium III and AMD K7 processors as a function of the effective Linear Energy Transfer (LET).

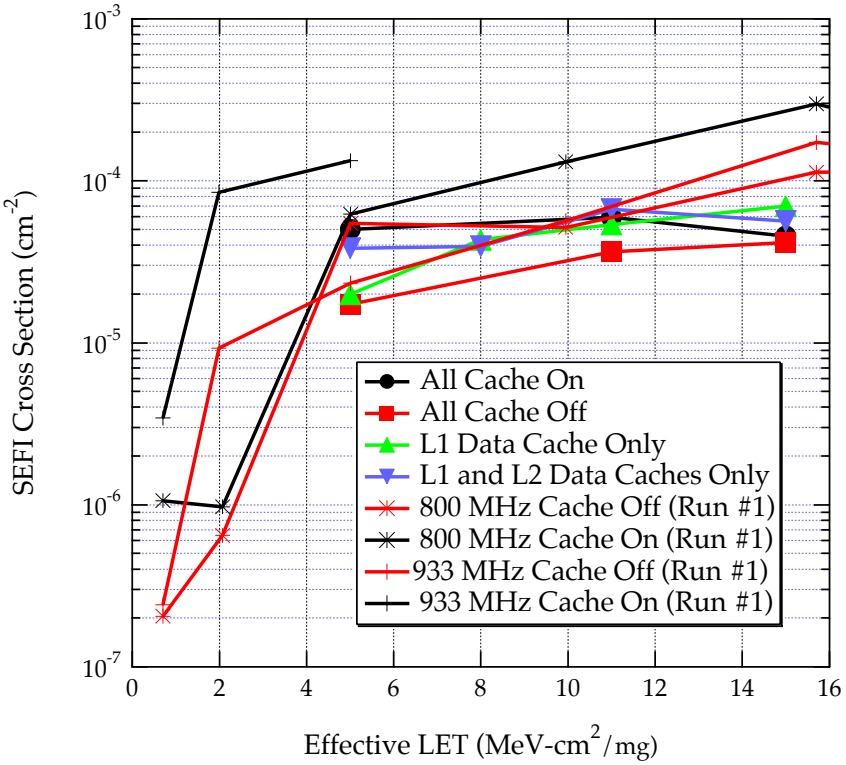

Figure 10. SEFI cross-section of the Pentium III processors as a function of the effective Linear Energy Transfer (LET) from both tests.

Single Event Upsets – Non-SEFI

With the above determination in mind, the remainder of this report deals exclusively with the Pentium III processor. Also, from this point forward, processor speed is not explicitly shown in the figures. As was stated for previous figures, it is true for the following figures that no speed sensitivity was observed for any of the tests.

Next to be considered is the exception cross-section (i.e., those events, if not handled, would lead to a SEFI). Figure 11 shows the data for the exceptions as a function of LET and cache state for the two heavy-ion tests.

As with the SEFI, the LET threshold is less than 0.7 MeV-cm^2/mg. There does appear, however, to be a more significant roll-off with LET above an LET of 10 to 15. The fact that events shown in Figure 10 and Figure 11 are still observed at the higher LETs, indicates that as long as charge is deposited in the region between 800 and 900 microns, events can occur. It is not necessary for the ion to actually reach the sensitive region.

Unlike the SEFI cross-section, it can be seen in Figure 11 that there is a strong dependence on the cache state for the exception rate. There is about an order of magnitude difference between the cases of "cache off" and any of the "cache on" cases. There is a small difference when the data caches are enabled, but the primary effect is seen when the instruction cache is enabled.

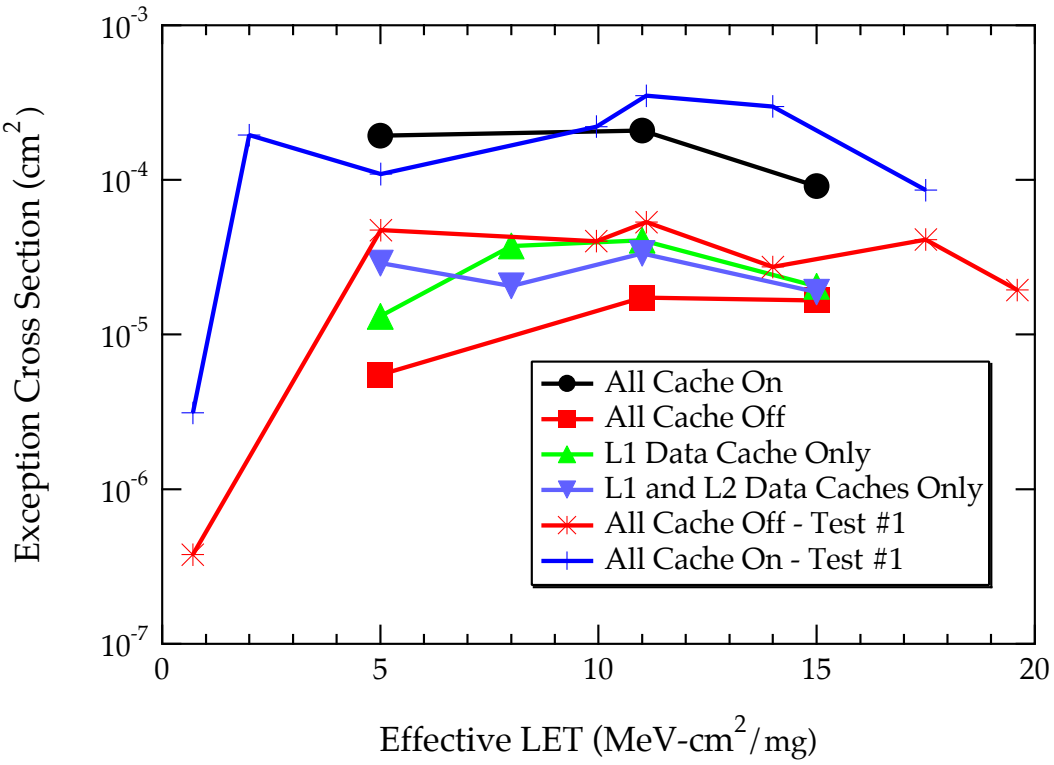

Figure 11. Exception cross-section of the Pentium III processor as a function of LET.

The next in line to consider for non-SEFI events are errors during the individual software tests (Tests A-G, see Software Section). Tests A, B, D, F, and G will be considered first as in all these cases very few to no errors were observed. Fluences during these tests were typically in the range of 10^5 to 10^6 ions/cm^2. Therefore, the registers, the floating-point unit registers and combinatorial logic and the MMX unit have very small cross-sections, on the order of a few time 10^{-6} cm^2 or less.

Tests C and E test the data and instruction caches, respectively. These areas of the die do show a high sensitivity to upsets and will be dealt with separately. The first to be considered in the instruction cache test, Test E.

Figure 12 shows the per-bit cross-section for the instruction cache as a function of effective LET for the two device speeds tested. It is easy to observe in Figure 12 that the cross-section is in the saturation regime over the LET range tested. The possible exception to this is for the 1000 MHz parts, roll-off may be coming into play for the effective LET 15 point. As in the SEFI and exception cases, it is difficult to say whether this is truly roll-off or just statistics. It should be noted, though, that the difference in the 933 and 1000 MHz parts shown in Figure 12 is not seen in the exception or SEFI data. That is why that data was not plotted as separate device speeds.

We can say with some confidence, though, that the threshold LET will be less than 1.0 MeV-cm^2/mg and the saturation cross-section is on the order of 2 x 10^{-8} cm^2/bit.

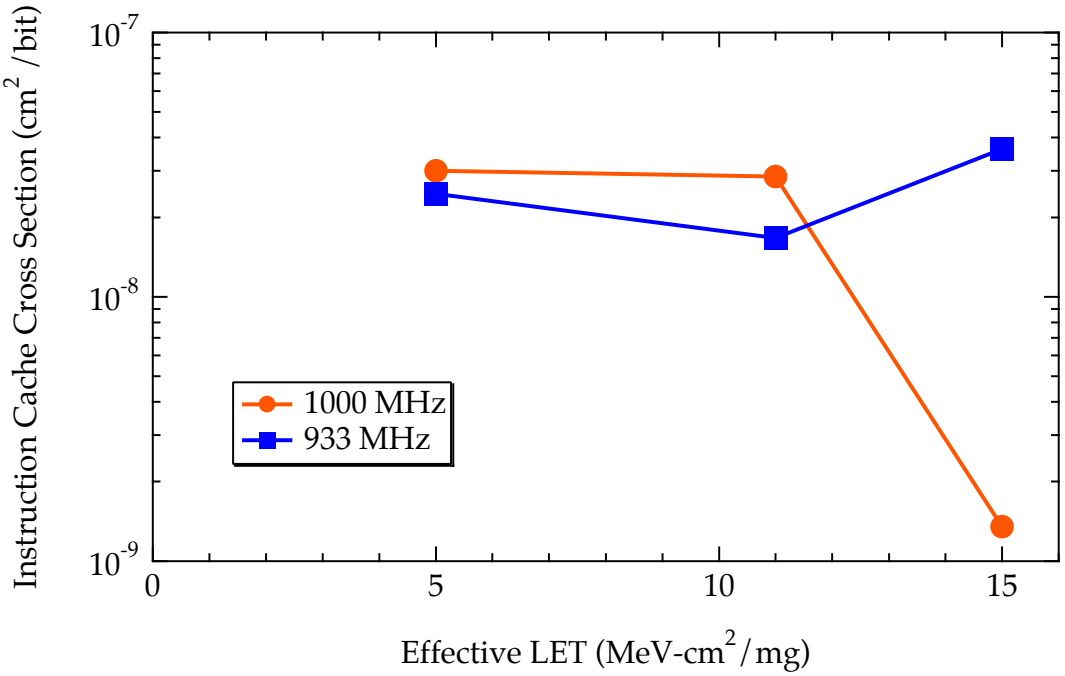

Figure 12. Data from Test E showing the instruction cache per–bit cross-section as a function of the effective LET. Shown is the data for both the 933 and 1000 MHz parts.

The final test to be discussed, Test C, tests the data caches. As in the previous proton testing, the data caches were controlled individually so that two cases were possible, testing with the L1 Data cache only and testing with both the L1 and L2 Data caches. Unlike previous testing, these cases were able to be further divided into percentages of the total cache size (either L1 only or L1 and L2). These percentages were 100, 50, 25 and 1% (except for the L1 and L2 case where the 1% run was not possible as this would make the L2 cache tested smaller than the L1 cache and the L2 cache would therefore not be sampled).

As stated in the software section, there are three types of errors that can occur in Test C. They are recurring memory error, tag error, and bit error (both 1 to 0 and 0 to 1 cases are tracked). With this improved software, recurring memory errors were not observed in this testing. We believe that previously observed recurring memory errors were an artifact of the software rather than a true stuck bit.

In addition, in this testing, different types of tag errors were observed, such as single tag errors, multiple tags in error within the same test loop, and multiple occurrences of the same tag error across consecutive test loops. The mechanisms for the latter two of these tag errors are not yet understood or why they almost exclusively only occur for the case of the L1 and L2 data cache (only two events occurred for all the L1 only cases as compared to 139 events for the L1 and L2 cache cases, with similar fluence levels of 10^5 to 10^6 cm^{-2}). Therefore, for the L1 Data cache only, the cross-section for these multiple tag errors is $\leq 10^{-10}$ cm^2. For the L1 and L2 Data cache case, the cross-section for each of these multiple tag errors is approximately $2 - 4 \times 10^{-10}$ cm^2.

The final tag error case to consider is the single tag error. Figure 13 shows the per-bit cross-section for a tag error as a function of effective LET for the two cache cases tested. The error bars on this graph are the one–sigma statistical variation across the parts tested (both part-to-part and speed differences). The error bars on the L1 only case are

significantly larger than the L1 and L2 case (they almost fall within the plot symbol). This is due mainly to the much smaller number of errors seen in the L1 only case.

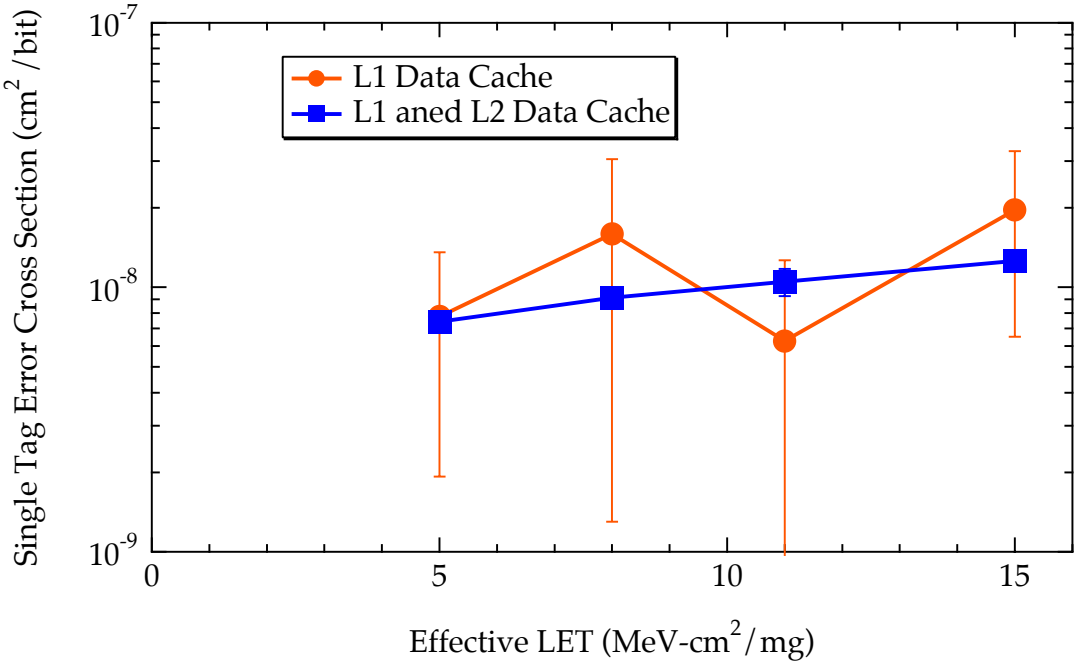

Figure 13. Data from Test C showing the per–bit tag cross-section as a function of effective LET. Shown are the two cases of L1 data cache only and the L1 and L2 data cache.

It can also be seen in Figure 13 that the cross-section is near, if not at, saturation levels across the entire range of LETs used. This would indicate that the threshold level is small (< 1) and that the saturation cross-section is in the range of 1 – 2 x 10^{-8} cm^2. These numbers are consistent with the data from the instruction cache test. Since the tag bits are physically the same technology and manufactured within the same array as the actual caches we should expect to see similar numbers between the actual cache bit error rate (both instruction and data) and the tag bit error rate.

The final result to be considered for Test C is the cache bit upset cross-section. When the data is collected during the testing, bit upsets are classified as either a 0 to 1 or a 1 to 0 upset depending on their initial state. These results collected during this testing showed that there is no statistically significant difference between a 0 to 1 and a 1 to 0 upset (i.e., they occur at the same rate). It should also be noted here that the total number of upsets allowed during a single loop through the cache being tested was much less than 1% of the total cache size being tested, thus the possibility of a bit upsetting out of and back to its original state is extremely unlikely. With no difference in the upset sensitivity due to initial state, data presented from this point will be for the total bit upset cross-section (i.e., the sum of the 0 to 1 and 1 to 0 cross-sections).

The next issue to discuss is the process of the cache bit testing and its time implications. After the appropriate portion of the cache is loaded with the pattern (alternating 1's and 0's), each bit is read, its complement written back and read again. The purpose of this sequence is to first determine if the bit was upset since the last visit to the bit. The second reason is to determine if the bit is possibly stuck by writing a

different value and making sure that the write actually happened. The disadvantage of this scheme is the amount of time it takes to do these three steps. While this may seem to occur very quickly, when it has to be done 2,097,152 times (for the 100% L1 and L2 case), the loop time can be significant. This is especially true considering the rate at which the processor can have an exception or SEFI event.

To determine how significant this timing issue is, a correction factor was determined based on a number of factors. For each run at the accelerator, besides collecting the number of bit upsets, exceptions and whether a SEFI occurred, the time that run took is also recorded. Using a typical time for a loop through the portion of the cache being tested (as determine in bench-top testing), a number of loops and errors per loop can be determined. The correction factor comes in whenever a loop is interrupted. This will occur every time either an exception or a SEFI occurs. The case for the SEFI is obvious but for the exceptions, it must be stated that the process control for an exception is to deal with the exception by resetting the processor to a known state at the beginning of a run and restarting the process. While this does not take a large amount of time, it does interrupt the loop that was currently running.

When a loop is interrupted, there is, in effect, an entire loop that has not been done. This is due to all the bits logically after the bit tested when the loop stopped and all the bits logically prior to the bit being tested that has been reset and may have already upset again. Therefore, a correction factor for each case is determined based on the average upset per loop for that case and on the total number of SEFI and exception events that occurred for that case. As would be expected, the cases were the number of bit being tested is small (all L1 only cases and the 25% L1 and L2 case), the correction factors were completely negligible (<< 0.1%). For the larger percentage cases for the L1 and L2 cache, the correction factor was still small (on the order of 1 to 3%). While not significant for this data set, this process can become important as the cache sizes continue to increase. Currently available P3 and P4 processors are being distributed with 512KB of L2 cache and Motorola PPC processors are being distributed with 2MB of L3 cache. Even though little difference is shown in the data, the correction factor has been kept for the remainder of this report.

The final issue to consider is the number of bits for each of the cases. We had thought that this would not be an issue and the correct number of bits would just be the fraction of the total for each of the different percentage cases run. While this seems to be a good assumption for the L1 only case, Figure 14 seems to indicate differently for the L1 and L2 cache cases.

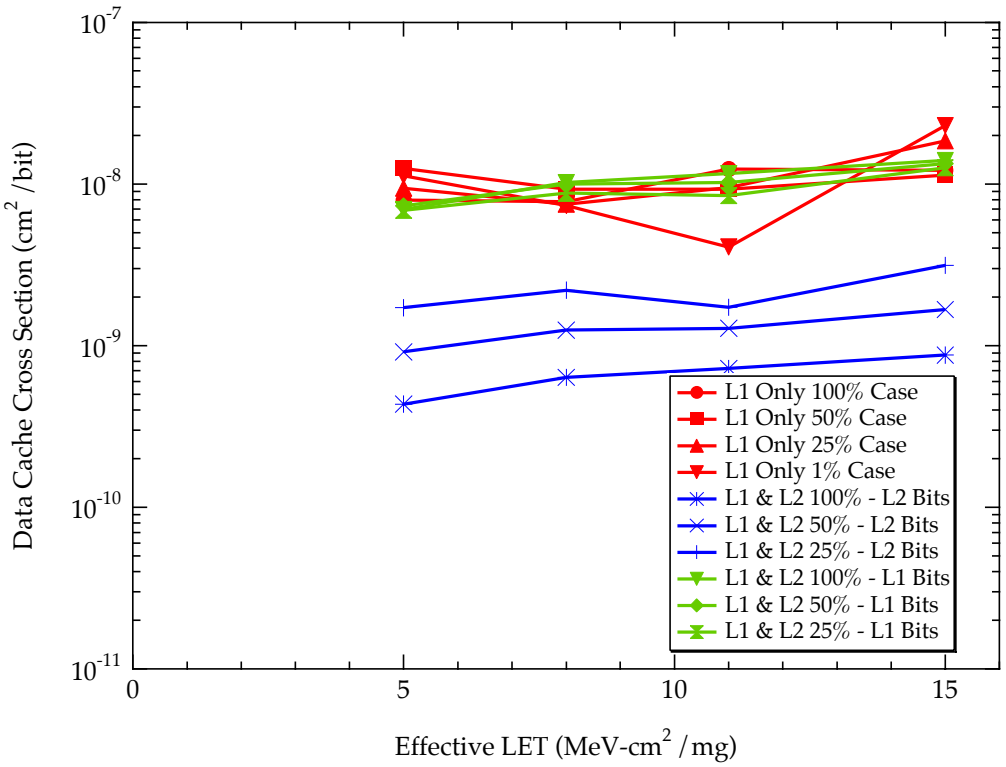

Figure 14. The per-bit cross-section for the data caches is shown as a function of the effective LET. The data for L1 and L2 case is shown with two bit cases, the proportional L2 bits and the 100% L1 bits.

The four red traces represent the four percentage cases for the L1 only case. As can be seen in Figure 14, these show a saturation cross-section trend across the effective LET region tested and a saturation cross-section of approximately $1 - 2 \times 10^{-8}$ cm^2/bit. This number is in line with the results from Test E on the instruction cache and the results from the tag errors from Test C. This is what one would expect if all the caches are of the same technology and layout (as is typically done).

The problem arises when the per bit cross-section is calculated for the L1 and L2 cases using the proportional number of bits based on 100% of the L2 cache test. These three percentage cases are shown in blue in Figure 12. As can be seen, the curves fall below the L1 only case by an order of magnitude or more. More significantly is the spread in the curves as a function of the percentage of the L2 cache tested. This could possibly be explained by having two mechanisms that lead to upsets, one that is not related to the number of bits tested (e.g., in the periphery circuits) and the bit cell upsets. An attempt was made to determine what single curve could be subtracted from each of the three total cross-section curves for the L1 and L2 percentage cases that would lead to the three per-bit percentage cases falling on top of each other. While there is no physical basis for this methodology, for the postulate given above, this curve must exist.

The best attempt at finding this curve and the resultant per bit curves is shown in Figure 15. In this figure, the red curves are the same as in Figure 14. The single green curve is the mystery mechanism that has not bit number dependence. It is shown here at the same level as the L1 data for convenience only. The actual cross-section values are

five orders of magnitude larger, as it is a per-device cross-section rather than a per-bit. The three blue curves in Figure 15 are the remainders of the three L1 and L2 percentage cases after subtracting out the mystery mechanism curve and dividing by the proportional number of L2 bits for the percentage case. While the curves have grown somewhat closer together, the grouping is still not very tight and the values are almost two orders of magnitude lower than for the L1 cases (and for that matter the instruction cache and tag bits). This difference does not seem to make a good fit or reasonable explanation.

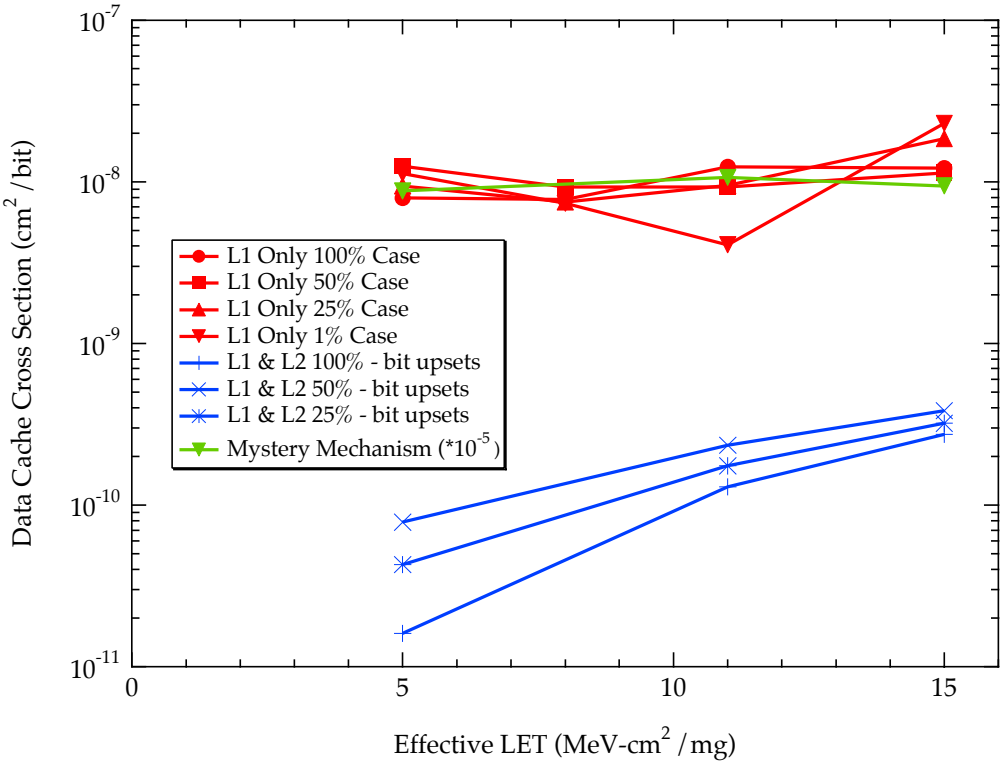

Figure 15. Data cache per-bit cross-section curves showing possible two upset mechanism for the L1 and L2 cases.

To investigate an alternative explanation, we can return to Figure 14. In this figure the three green curves represent the three percentage cases for the L1 and L2 data cache that have been made per-bit by dividing the cross-section by the total number of bits in the L1 cache (L1 Data 100% case). It is easy to see that these curves now fall directly on top of each other and on top of the L1 only curves, indicating the same per-bit cross-section for all of the caches, under all percentage cases and their respective tag bits as well. This situation makes the most sense. The problem is trying to explain why the 100% L1 data cache size is the correct number of bits to be dividing by. At present, we have no explanation for this but it is still under investigation.

Assuming this is the correct situation, the per-bit cross-section for the data cache is the same as the instruction cache and the data cache tag bits at approximately 1-2 x 10^{-8} cm^2/bit. All of these curves are relatively flat over the entire effective LET range tested, indicating that the threshold should be small. A good estimate from this and earlier testing is that threshold will be at an effective LET of less than one.

VII. Lessons Learned

Throughout the 18 months that this project has been in place, numerous changes in procedures, hardware, software, etc. were done as a result of learning how to test these state-of-the-art microprocessors. These lessons learned are grouped into three categories: hardware and testing, DUT software and analysis software.

A. Hardware and Testing

The approach taken in the earliest testing turns out to be quite on-target. Specifically, the choice of commercial hardware (motherboards especially) as opposed to special versions (e.g. VME or cPCI versions) allowed the modification and evolution of the testing to occur without huge materials cost and development time.

- The choice of the SC-242 module form-factor was also prescient in allowing access to more analog signals. All analog signals were not available upon shifting to the Pin Grid Array package.
- Originally, the DUT operating system and application resided on a floppy disk. It was discovered that boot time was considerably improved by having it reside on a hard drive.
- The choice of a minimalistic operating system such as Pharlap with minimum boot time was also proved to be important. Besides the software benefit of more control of the processes, the reboot time is an important aspect of testing. The longer the reboot, the more time required for a given set of tests (or less testing in a given time period).
- The issue of whether static or dynamic testing of registers and other components is more informative is still unresolved. It seems, though, as if dynamic testing is more similar to actual processor performance, and thus will yield better predictive efforts, with much greater confidence.
- In future endeavors a method of capturing boot-time monitor output would be helpful. This was found to be outside of the field of expertise of this team when using off-the-shelf commercial technology, and thus a careful eye was kept on, but no permanent log was kept of, the booting of the DUT Computer.
- Initially certain types of SEE-induced errors were anticipated and code was written to accommodate them (by trapping execution errors and reporting machine state information) but it was not anticipated that things like general protection faults (GPFs) would occur nor how they would affect operation. By having the DUT Computer software development team attend testing, on-the-fly code modifications were made and data capturing improved dramatically.
- The PXI form-factor Test Controller was found to be a good choice. Due to the frequency of switching, in future tests solid state switches would be considered advantageous over the reed relays used in these tests.
- The RS-242 ports were chosen for telemetry and command because of their ubiquitousness. Because they are simple and served well in initial tests, they were retained as the primary link for all testing. However, in future testing a look at front side bus (FSB) telemetry would be warranted. Proximity of FSB monitoring equipment in TID testing would be a concern, however.
- The original method of physically fixturing the DUT Computer for SEE tests was cumbersome—essentially, it sat on a static mat. A fixture, which holds ATX form-factor motherboards (ATX apparently defines a standard mounting hole

pattern) either horizontally (for SC-242) or vertically (for FC-PGA), alleviated this problem.

- In TID testing, fixturing of the motherboard inside the Lead-Aluminum box remained cumbersome. Future testing should have better mounting of the motherboard and allow for such things as cable bend radius. Shielding of non-DUT components (stacking of Lead bricks) was not done in a specialized way; perhaps this would reduce the downtime and mortality in the future with more design time.
- Originally the ATX power supply was modified to extend the 20 conductors by ~6'. While this was necessary due to the difficulty (leadtime) in obtaining the connectors necessary to build an extension cable, the eventual construction of the extender allowed for easy replacement of power supplies.
- Cooling was an issue which required re-attending each time a higher-speed and power DUT was tested or form-factor changed or heavy-ion testing was done. No thermal solution involving thermal grease was allowed in proton testing, and cooling solution thickness limitations of ~10 mil for heavy-ion testing exacerbated the problem. Water-cooled thermal plate ended up being the solution of choice, and eventually the components most compatible with the circumstances (testing facilities and DUT fragility and heat production) was found.
- The process of producing the cooling plates could use some improvement. The plates were locally manufactured so that there was a limitation on how thin and uniform the material over the die could be made. Possible use of commercially available metallic foils secured to the cooling plate would be a better option.
- JTAG, or boundary scan test access standard, is a standard for testing complex circuitry without having to build extensive hardware and software components for each different circuit. It involves functionality designed into individual circuits, and protocols for communication with the JTAG functions built into circuits. Essentially, complex circuitry (in this case a microprocessor) has built inline with each input and output bit, and various internal registers, a boundary scan bit register. All boundary scan bit registers are connected in series, with a simple serial port (test access port, or TAP) to read or write data to/from the circuit's boundary scan bit registers. Testing involves clocking into the TAP a series of bits that set an initial condition at each of the input pins and internal registers and then operating the device. Monitoring functionality involves stopping the device and reading the state of the bit registers. Very complex circuits, and entire assemblies of complex circuits, can be daisy-chained to be tested in this method.

  JTAG was initially considered for testing of the P3 processors. The primary reason for not using JTAG was that was difficult to obtain information from Intel regarding the P3's JTAG functionality.

  Beyond that, though, there is the issue of realism—JTAG seems to have been most useful in radiation effects testing for static, as opposed to dynamic, testing. The efforts of NASA GSFC REA Group have been primarily directed towards dynamic testing, at actual processor speed, since that will reflect real sensitivities as opposed to static register sensitivities that cannot be assumed to reflect any separable sensitivity in an operating processor.

  However, there was a data cost to not having JTAG monitoring functionality. In cases where the processor suffered a SEFI without first communicating the

SEFI's proximal cause, boundary scan probably could have provided information on the state of the processor, and thus the manifestation of the SEFI.

It is not known how complex the use of JTAG and the analysis of the results would have been. In summary, it is suggested that future endeavors consider JTAG as a possible tool, especially if SEFI events dominate SEE testing.

B. DUT Software

- The Device Under Test (DUT) Software was originally designed to run flawlessly reporting errors only in that part of the CPU that was under test. We quickly found out that most of the errors encountered were causing exceptions and terminating the program. A solution was partially implemented during the first IU testing in 2000 (i.e., on-site software support is critical).
- Our second trip to IU in 2000 had this error fixed and we were able to run for significant periods of time without the program crashing. The exceptions were handled, an error message was output and the program continued normally. This extended run time allowed for test results that demonstrated other weaknesses in the test software, especially in the bit testing of the caches. This type of testing needs an in situ test process that can inject errors that would allow for software development outside of the expensive test facilities. A laser based SEE system would have been very useful in this regard.
- On our first trip to TAMU in March of 2001, we discovered that the cache was causing most of our errors and that when the cache was on, the program would terminate before we could get a large volume of data. This led to the understanding of the fact that the architecture allowed for control of all the individual caches. It is probably worth the time and expenses to investigate and take any architecture courses that the processor vendor may offer. More technical discussions with motherboard vendors would also have been useful if they had been more forthcoming.
- On our third trip to IU in May of 2001, we added the capability to cache only the data area of the program. We also enhanced the error report on the cache test to include the location of cache errors within the buffer. During the trip, we further enhanced the cache test to compress the enhanced output and to add the value of the location in error to the report. This enabled us to distinguish cache tag errors from bit errors. It also allowed us to distinguish 1 to 0 transitions from 0 to 1 transitions for the bit errors.
- On our second trip to TAMU in November of 2001, we added the capability to utilize different percentages of the cache. We wanted to be able to distinguish between errors in the delivery of data from the cache and errors in the cache memory.

C. Analysis Software

Throughout the process of creating and using this very useful tool, a number of items were thought of that were not implemented either due to insufficiency of the software originally chosen for the development or time constraints. A list of these items is as follows:

- Create more simplified data entry tools. These would be "short cuts" to data analysis. By creating more automated "short cuts", it becomes less tedious and less prone to error.

- Add the artifact data from the “trip” to the install. (Problem install file get huge and hard to transfer). Define a known location to store the data.
- Understanding and developing a method to translate all possible principal investigator’s Excel spreadsheets (one spreadsheet for each run) and test runs into a database used to create the analysis outputs. When changes in formats are made, it creates the need for custom programming.
- Be able to analyze across multiple facility runs (e.g., all proton testing).
- Need checking routine that verifies that all data, errors, etc. make it into the final database (we always had to iterate the final database until all the data was there, at least all the ones that were caught!).
- Most times the only data needed was number of events and fluence, but for a large number of SQL cases. It would have been nice to be able to click a button that recorded that info into a file (i.e., the file is kept open across multiple SQL cases and whenever the button is pushed the event count and fluence is appended to the file).

VIII. Conclusions

Extensive data has been collected on the total dose and single event response of the Intel Pentium III and the AMD K7 microprocessors. The data indicates that there is a high tolerance to total dose and there is no susceptibility to latchup from protons. Single event upsets and functional interrupts are present. However, for the Pentium III, if running with the caches disabled is an option and with mitigation in place, these events may be controllable to allow for operation in the space environment. The thermal issues and the power requirements of these processors will most likely be the limiting factors in their usage in space applications.