JPL Publication 93-12

Test Report for Single Event Effects of the 80386DX Microprocessor

R. Kevin Watson
Harvey R. Schwartz
Donald K. Nichols

February 15, 1993

NASA

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Funding from this support came from Code QR, Reliability, Maintainability, and Quality Assurance, for the Microelectronics Space Radiation Effects 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.

Table of Contents

List of Figures

List of Tables

ABSTRACT

The Section 514 Single Event Effects (SEE) Testing and Analysis Group has performed a series of SEE tests of certain strategic registers of Intel's 80386DX CHMOS IV microprocessor. Following a summary of the test techniques and hardware used to gather the data, we present the SEE heavy ion and proton test results. We also describe the registers tested, along with a system impact analysis should these registers experience a single event upset.

INTRODUCTION

Single event latchup (SEL) and single event upset (SEU) may be induced in devices operating in space by cosmic rays and/or solar flare particles. SEL occurs when a cosmic ray turns on a parasitic bipolar PNPN structure. Device malfunction with excessive current and local heating are the result, ending in possible catastrophic failure. In contrast, SEU is a soft error that occurs when a cosmic ray causes the data stored in a circuit element to flip to the opposite state.

Obtaining SEE data for the Intel 80386DX microprocessor is of importance to the NASA Microelectronics Space Radiation Effects Program (MSREP), because several NASA projects are considering using this part. Most notable to date, Space Station Freedom will use 80386DX-based computers for its data management system, and the Small Explorer Data System (SEDS), a subsystem of the Solar Anomalous Magnetospheric Particle Explorer (SAMPEX) mission, uses an 80386DX as its main processor.

From an SEU standpoint, 80386DX registers can be placed in one of three classifications: application level, system level or hardware level. Application level registers perform functions which are not generally crucial to the integrity of a system. Two examples are the extended source index (ESI) register, used to move blocks of data in memory, and the extended BX (EBX) register, used for temporary data storage. System level registers, on the other hand, can have a direct impact on the integrity of a system should they be upset. Examples include the global descriptor table register (GDTR), involved in system memory management, and the task register (TR), involved in task management. The hardware level register classification covers everything that cannot be directly addressed on the software level. These types of registers run the gamut in terms of their ability to affect system integrity. Even though Intel has refused to release details on these types of registers, a few can be tested. An example of a hardware level register that can be tested is the group of 34 address latches behind the 80386DX address pins. Appendix A summarizes all known 80386DX registers.

Previous testing[1] concentrated mostly on the less important application level registers. For this reason JPL has undertaken an effort to gather data on the crucial system level registers.

[1] Tom Scott, Space Station Program, Data Management System, *Systems Engineering and Integration Trade Studies; Single Event Test Method and Test Result for Intel's 80386*, IBM Report No. 891BMX0215, IBM, Manassas, VA (April 17, 1989).

80386DX OVERVIEW AND BACKGROUND

The Intel 80386DX is a very high performance Complex Instruction Set Computer (CISC) microprocessor that at 33 MHz can execute instructions at a peak rate of 8 to 9 million instructions per second. It has separate 32-bit data and address paths that can provide a peak throughput of 66 megabytes per second (again, at 33 MHz) while directly addressing 4 gigabytes of physical memory. In addition, a sophisticated six-way pipelined architecture allows the 80386DX to perform instruction fetching, decoding, execution, and memory management functions in parallel, with little or no time penalty.

The 80386DX is available in three speeds: 20, 25 and 33 MHz. Older 20 MHz and possibly 25 MHz parts were fabricated using the Intel Complementary High-Performance MOS CHMOS III process. All current versions are implemented in Intel's 1.0 micron double-metal CHMOS IV process. Intel's literature puts the transistor count at 275,000, but this may be misleading. *Microprocessor Report's* Michael Slater[2] indicated that there are about 156,000 actual transistors used in the design. The discrepancy can be attributed to Intel having counted all possible transistor sites on the die as opposed to the actual number of transistors used. Intel states in its data sheets that neither version of the 80386DX can latch up because the CHMOS III parts are on an epitaxial substrate while the CHMOS IV parts use twin-wells.

Intel manufactures several variants of the 80386DX, which include the 80386SX with a 16-bit external data bus, the 80376 microcontroller, and the two-chip 80386SL, which provides a high level of integration for portable computer manufacturers. The 80386DX die is mounted in a ceramic 132-pin pin-grid-array package with the lid on the same side as the pins, presenting an awkward situation for SEE tests at oblique beam angles.

Although Intel does not second-source the 80386DX[3], the market demand and multi-billion dollar software base have prompted several companies to reverse-engineer and copy the 80386DX. These companies include: Chips and Technologies, NexGen and AMD. On March 25, 1991 AMD was the first to formally announce their copy: the Am386DX which, among other attributes, consumes less than half the power of the Intel part, is fully static, and can be clocked up to 40 MHz. The Am386DX is fabricated using AMD's CS21S process and has a minimum feature size of 0.8 microns with 0.9 microns as an average. To achieve a fully static design, AMD needed 4000 more transistors than Intel used, for a total of 160,000. To provide comparison information, the AMD AM386DX device was also tested. Its latchup (SEL) threshold occurred at LET=24 MeV/(mg/cm^2). This result led us to discontinue the test.

[2] Michael Slater, "AMD Formally Announces Am386DX", *Microprocessor Report*, Vol. 5 No. 6, April 3, 1991, p. 6.

[3] IBM is licensed by Intel to manufacture the 80386DX and its variants, however IBM uses them strictly for their own systems and does not sell them as a vendor. IBM, therefore, is not a second source in the strict sense.

MODES OF OPERATION

The 80386DX has three distinct modes of operation: 1) real address, 2) protected virtual address, and 3) virtual 8086. Following is a description of these modes.

1) Real Address Mode

In real address mode, the processor provides fast execution of 8086 instructions in an unprotected environment; i.e., the program that is executing "owns" the processor and can do anything it likes, including crashing the system. This mode of operation also duplicates the 8086 segmented addressing model where logical addresses are combined with segment information to form linear addresses which map directly to the physical address space. Real address mode allows use of some of the 80386 enhancements such as a few new addressing modes, access to some 32-bit registers and full 32-bit address space with the appropriate software instruction prefixes. The original Intel 8086 only has 16-bit registers and a 20-bit address space.

2) Protected Virtual Address Mode

This operational mode adds a new type of address translation, paging, and increases the maximum segment size to 32 bits vs. the real address mode's 16-bit segments. Paging and segmentation allow an operating system to translate addresses. On the 80386DX this means that any logical address supplied by a programmer goes through a level of indirection (segmentation), becoming a linear address. This linear address is then subject to yet another level of indirection (paging), becoming a physical address which is used to drive the address bus. This address translation mechanism is a very important concept, because it allows an operating system to mark certain areas of memory as unavailable to a task, or even invisibly mask out portions of memory.

The 80386DX can also implement demand paging to create a virtual environment for programs. In this case, neither the program code nor the application programmer needs to know how much RAM is available, or where it is located. If a program makes reference to a nonexistent page of memory, the 80386DX will trap the reference and call a *page fault handler*, which will then retrieve the desired data from a mass storage device, such as a hard disk, and place it in memory. The previous memory contents are effectively swapped with the data on the hard disk. The 80386DX can provide as much as 64 *terabytes* of this virtual memory.

Multitasking is one of the most powerful features of the 80386DX. At the hardware level it is designed to keep track of an almost limitless number of tasks. A task can be one of many programs that appear to be running simultaneously, but in actuality the processor allows each task to run for a certain amount of time before a *task switch* takes place and the processor gives its resources to another task. Task switching is very fast, as only one assembly instruction is needed to perform the switch, with the rest of the operation being handled by on-chip hardware.

Privilege Protection is implemented in hardware on the 80386DX and is a way of making sure that unfriendly tasks do not affect the operation of other tasks (most importantly, the operating system itself) that may be running concurrently. With this method, every piece of code and data is assigned one of four privilege levels called *rings*, where ring 0 has the highest priority and ring 3 the lowest. The 80386DX automatically performs this privilege validation on every memory cycle. This privilege validation takes form in five ways: type check, limit check, restriction of addressable domain, restriction of procedure entry points and restriction of instruction set. If the task is privileged enough, its memory access will be granted. If not, the processor will deny access and generate a privilege exception; the operating system will then take over and decide what to do with the offending task.

3) Virtual 8086 Mode

Virtual 8086 mode, a sub-mode of the protected mode, establishes an 8086 execution environment within the protected, multitasking environment of the 80386DX, thus, allowing an operating system to execute multiple 8086 and 80386DX programs concurrently.

TEST PHILOSOPHY

In this SEE test, as with all microprocessor tests performed by JPL, the test philosophy is to isolate individual registers, load them with known data, and expose them to the heavy ion source, while monitoring register contents. When an SEU occurs, the error is counted, the device is reset to a known state (usually the post-reset state), and then set up for the next possible SEU. This approach yields absolutely traceable data in terms of which elements and how many bits are upset in an extremely complex device. Cross-section and threshold calculations may be calculated with great accuracy and confidence for a given set of device elements. This allows the system engineers to build an accurate cross-section for a device from individual register cross-sections, based on the actual device application. In the 80386, for example, the Translation Lookaside Buffer (TLB), which is used in paging mode, contains more than 1600 bits which have a major impact on the total device cross-section. If the TLB is not used in an application (i.e. paging is turned off), however, its cross-section may be disregarded by the systems engineer. This approach also helps the vendor identify those areas of a chip that can profit by hardness redesign.

Testing with an instruction mix, on the other hand, reveals very little information about the radiation hardness of a complex device other than the fact that something happened to it when it got "hit". The cross-section obtained with an instruction mix tells nothing about individual elements of interest to the systems engineer. This is true for a number of reasons: so many operations are occurring at many levels in an instruction mix that the cause of an error, and a possible correction procedure, may never be determined; the experimenter has no idea what the state of the device should be when an error occurs; the device may be flagging errors when none are present after the first one, simply because that first one sent the device into an unknown state; or the instruction mix may not represent the normal operating state of the device in a real-world application.

"AOK CIRCUIT" FOR REGISTER TESTS

The "AOK circuit", as it is informally known, is a circuit located on the DUT card which examines the state of the DUT at the time the test system would expect valid register data on the data bus. If the processor state is valid, a comparison is made between the data bus and a constant stored in a series of four 2K by 8-bit EPROMS. This comparison circuitry can compare byte, word, and double-word values that are byte aligned on the data bus. If the comparison determines that the data is invalid, a signal is sent off-card to a counter, indicating that a valid SEU has occurred. If the processor state is invalid, the test card does not "look" at the data bus, but instead sends out a signal to a second counter, indicating that an unknown portion of the DUT had been subjected to an SEU and was not functioning correctly.

DUT CARD AND TEST SYSTEM OVERVIEW

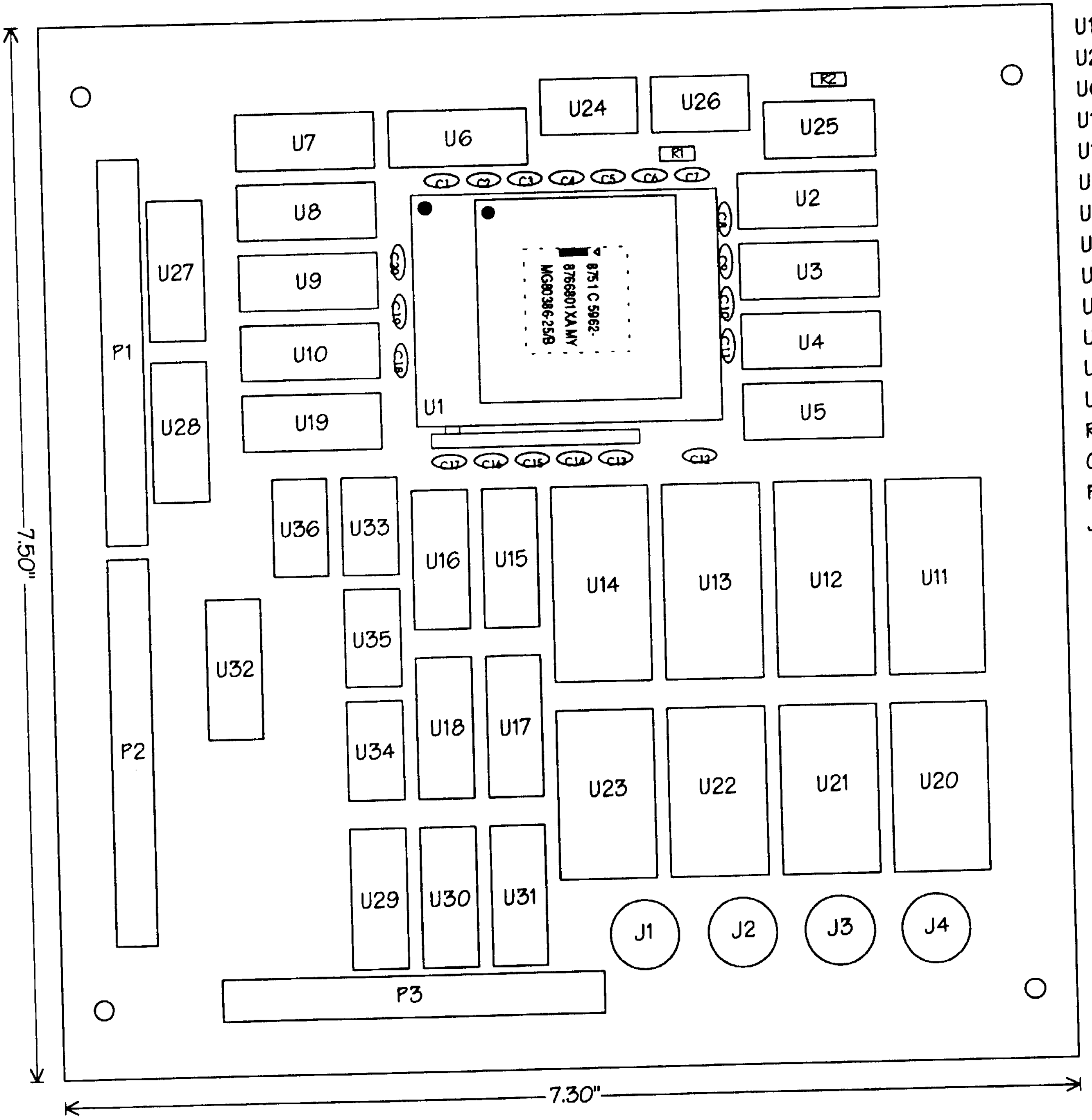
Due to the high level of complexity of the 80386DX, and difficulties inherent in testing such a complex device, a highly integrated approach to device testing has evolved. The DUT card actually contains a complete custom onboard microcomputer and monitoring circuitry, and as such becomes a part of the test system.

The test card is a 7.50" x 7.30" copper-clad vectorboard onto which the DUT and peripheral circuitry are soldered and/or wire-wrapped. The copper cladding provides a ground plane, which helps reduce "ground bounce" by minimizing ground inductance. The DUT and peripheral logic are powered and monitored separately by Hewlett-Packard power supplies using 4-wire techniques. The test card is controlled from outside the test chamber by a Hewlett-Packard 16520A 50-megabit/second pattern generator. A Hewlett-Packard 16515A one gigahertz timing analyzer can be used to observe the behavior of the DUT in the beam. The test card, in return, provides signals that indicate the status as well as the types of errors that have occurred. These signals are sent to a set of digital display counters for external monitoring. This test setup can operate synchronously at clock rates (at the CLK2 input) of up to 25 MHz while sorting and categorizing errors at a rate of over 12,000 per second.

For reference, Figure 1 is a one-to-one scale drawing of the test card, and Figure 2 is a test setup cabling diagram, showing all stimulus, feedback, and power connections needed to perform a SEE test. Additionally, a complete test card schematic is included in Figure 3.

TEST METHODOLOGY

During the test, the microprocessor will execute the instructions necessary to exercise the appropriate register. These instructions are located in four onboard 256K by 8-bit EPROMS that operate with no wait states. As few instructions are needed to test a



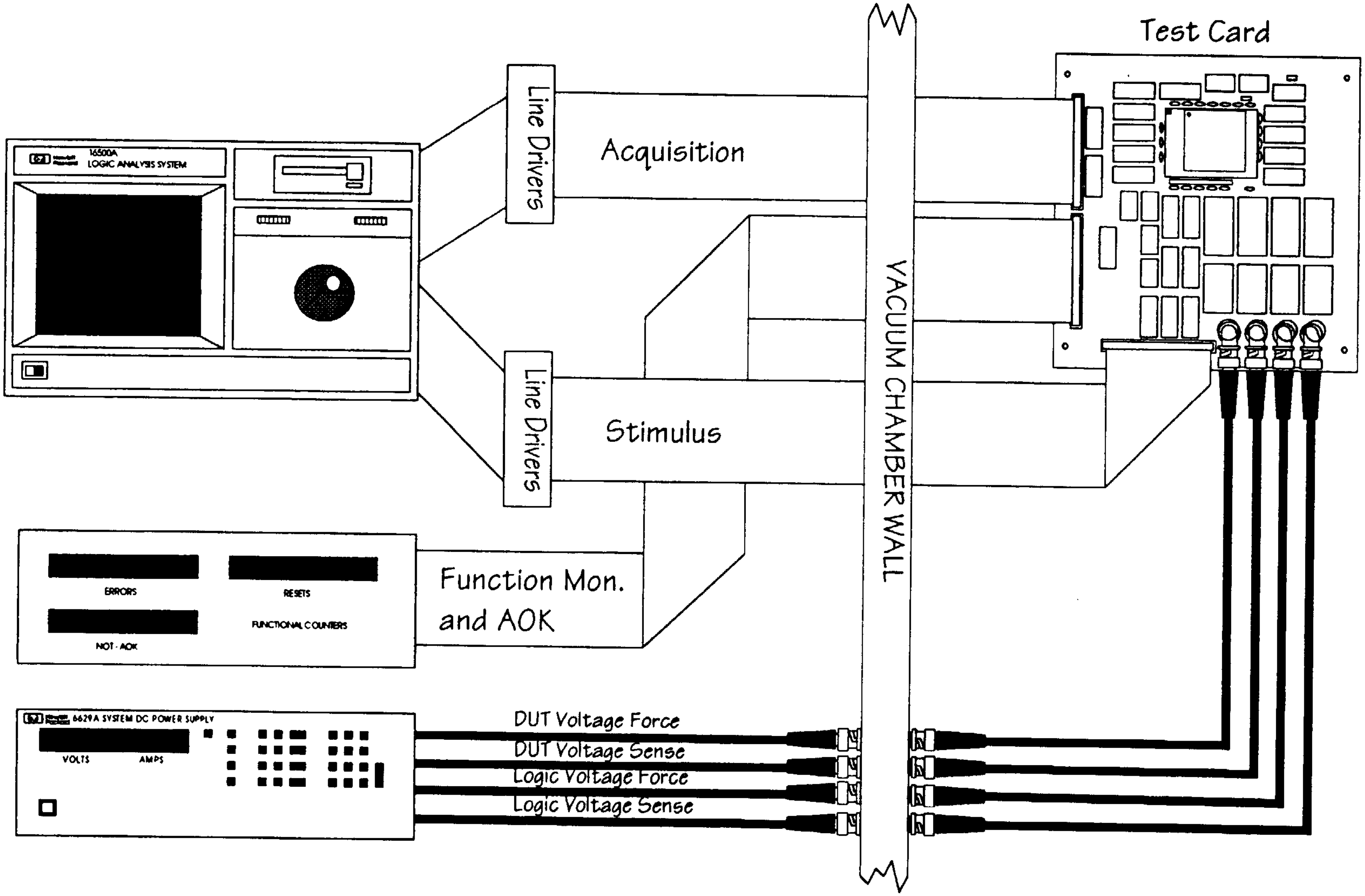
U1	80386DX (DUT)
U2,3,4,5	74LS245
U6,7,8,9,10	74LS373
U11,12,13,14	27C256
U15,16,17,18,19	74LS688
U20,21,22,23	2716
U24,36	74LS04
U25	74LS109
U26	74LS11
U27,28,29,30,31,32	74LS244
U33	74LS08
U34	74LS00
U35	74LS20
R1,2	4.7 KOhm ¼Watt
C1-20	.01 µF Ceramic,RF
P1,2,3	40-Pin Ansley
J1,2,3,4	Male BNC

Intel 80386DX
SEE Test Card
Component Layout
Scale 1:1

Jet Propulsion Laboratory
California Institute of Technology
Radiation Effects and Testing Group
Design and Layout: R. Kevin Watson
Construction: Michael O'Connor

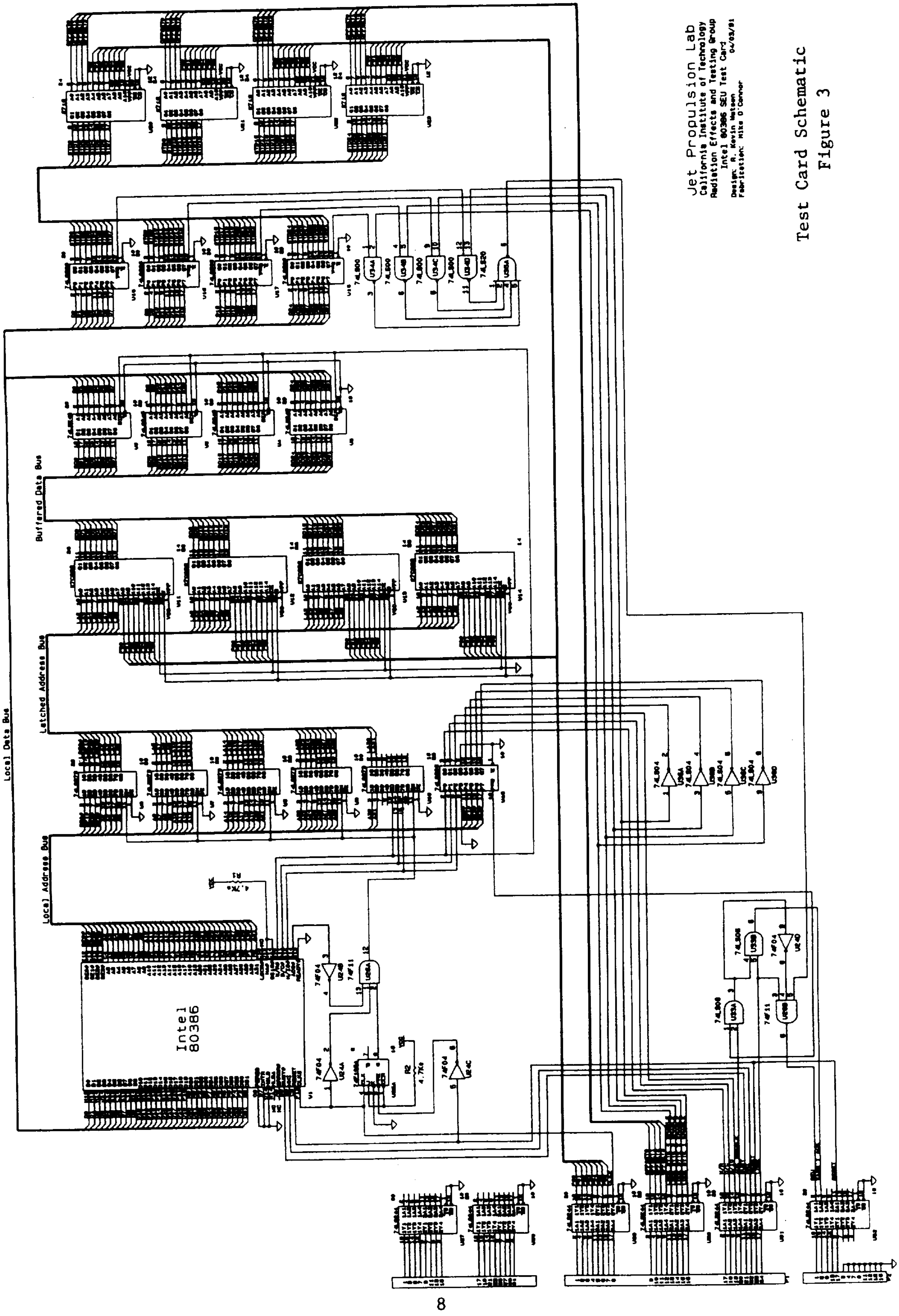
Figure 1

RKW 6/1/91



80386 Test Setup Diagram

Figure 2


Jet Propulsion Lab
California Institute of Technology
Radiation Effects and Testing Group
Intel 80386 SEU Test Card
Design: R. Kevin Watson 04/03/91
Fabrication: Mike O'Connor
Intel 80386
Local Data Bus
Local Address Bus
Latched Address Bus
Buffered Data Bus
R1
4.7K
R2
4.7K
74F04
74F11
74LS00
74LS04
74LS08
74LS20

Test Card Schematic
Figure 3

register, and there are many possible tests (i.e., register and pattern combinations) it is advantageous to page physical memory into the address space of the microprocessor. The type of test to be performed is then selected remotely by the pattern generator which selects the appropriate memory page. Program memory on the test card is organized as 128 pages, each containing 256 double-words. This memory is not decoded, meaning that each kilobyte page wraps around many times, occupying the entire address space. This allows the reset code to be executed at FFFFFFF0h, while non-maskable interrupts can still be serviced at 00000008h, all in one page.

This paging scheme also gives the test system the advantage of being able to control the type of instructions that the DUT can execute at any given time. The test system, as an example, can dynamically change the page to one containing all HLT (halt) instructions during the shutdown period of a test cycle, when the DUT is integrating errors. Should the DUT come out of shutdown mode, perhaps due to an extraneous interrupt, and start to execute code, the processor would be placed back into shutdown mode because its entire address space is filled with halt instructions.

Also on-board the test card is the AOK circuitry described earlier and necessary to help ensure the correct classification of errors.

As an example of a typical test run, the test system performs the following steps to test the CR3:

1) The DUT is reset, synchronizing it with the test system and placing it into a known state. The test card also sends a signal to an external counter, signaling that another test sequence has started.
2) The DUT initializes itself then starts execution at FFFFFFF0h, which contains a sequence of instructions that loads the CR3 with the desired pattern. Specifically, a MOV EAX,[32-bit pattern], then a MOV CR3,EAX.

3) The DUT then executes a HLT instruction, placing it into shutdown mode where it will lie dormant.

4) After the DUT acknowledges entry into shutdown mode, the test system changes the memory page to one that contains all HLT instructions (i.e., one that is filled with F4h).

5) The test system continues to clock the DUT for a period equal to about 80 percent of the total test cycle time. It is during this period that the actual errors are integrated in the register under test.

6) The test system returns the original memory page to the 80386DX address space.

7) The DUT is brought out of shutdown mode with a non-maskable interrupt (NMI). To acknowledge the NMI, the 80386DX must first get the NMI handler address from location 00000008h in physical memory, which contains 00000100h.

8) The processor starts to execute code at 00000100h and is given the instructions to move the contents of CR3 out onto the data bus. Specifically, a MOV EAX, CR3 then a MOV DX, 5550h and finally an OUT DX, EAX are executed to achieve this result.

9) At the point when the 80386DX would normally transfer the data onto the data bus, the test system examines the processor state to verify that a properly aligned I/O write is taking place. If the processor is not performing normally, the test card sends a signal to a counter indicating that the processor was found to be in an invalid state, and takes no further action. If the processor state is valid, the data bus will be checked for errors. If an error is noted the test card increments the external "valid SEU" counter.

10) The test sequence starts all over again at #1, when the pattern generator loops back and resets the DUT.

SPACE STATION FREEDOM'S OPERATING SYSTEM

The operating system chosen by IBM for Space Station Freedom (SSF) is LynxOS from Lynx Real-Time Systems Inc. in Campbell, California. LynxOS is a multitasking/multiuser operating system that is compatible with the two leading versions of UNIX: AT&T System V and Berkeley 4.3 BSD. LynxOS also conforms to several important standards: POSIX (portable operating system interface for computer environments, IEEE 1003.1), real-time extensions to POSIX (IEEE 1003.4), and Ada (the language of choice on SSF) bindings for POSIX (IEEE 1003.5).

During telephone discussions with George Kohli (Lynx Software Engineer), Mike Bunnell (another Lynx Software Engineer) and Paul Panepinto (Lynx Sales Manager) it was verified that LynxOS would make use of the 80386DX's protected virtual address mode of operation, with paging enabled. It was also ascertained that LynxOS was bought "off the shelf" for use on SSF and would not be modified before flight. This last fact makes system level register data even more important because LynxOS was designed for real-time control applications on Earth where SEE is not a problem. Thus, LynxOS is not "SEU aware" and would not protect the system from SEUs. Although, to be fair, it should be pointed out that no operating system could possibly prevent all such damage, but steps could certainly be taken to prevent some SEU related damage.

REGISTER DESCRIPTION AND SYSTEM IMPACT ANALYSIS

Below is a description of the registers tested by JPL to date. Also included is a general analysis of the system impact should that register become corrupted via an SEU. This analysis applies to any operating system that uses the full capabilities of the 80386DX's protected virtual address mode, including LynxOS. In the context of this analysis "crashing" generally means that the 80386DX itself enters "shutdown mode", which can happen for several reasons. As an example, if another exception occurs while the 80386DX is processing a double-fault exception it would signal that shutdown was imminent and immediately do so. To protect the overall system, an operating system can also initiate shutdown mode by executing a HLT instruction from ring 0. A 80386DX in shutdown mode would require a non-maskable interrupt or reset before it would begin to execute instructions again.

CONTROL REGISTER 3 (CR3)

The CR3 is a system level register containing the 32-bit, page aligned, physical base address of the *page directory*. For this reason, it also is sometimes called the Page Directory Base Register (PDBR), which is a data structure in memory that contains an array of 32-bit linear addresses. These addresses point to yet another data structure in memory called a *page table*, containing an array of 32-bit physical addresses. The operating system has the option of having one global page directory for all tasks, one page directory for each task or somewhere in between the two.

To translate a linear address to a physical address the 80386DX's paging hardware must perform many steps. The first step is to receive a 32-bit linear address from the segmentation hardware and split it into three components: a 10-bit page directory offset, a 10-bit page table offset, and a 12-bit page frame offset. Using the contents of the CR3 as a base address and the page directory offset as a pointer, the paging hardware looks up a page table base address in the page directory. The paging hardware then uses the page table offset to look up an entry in the selected page table. This entire process is outlined in Figure 4.

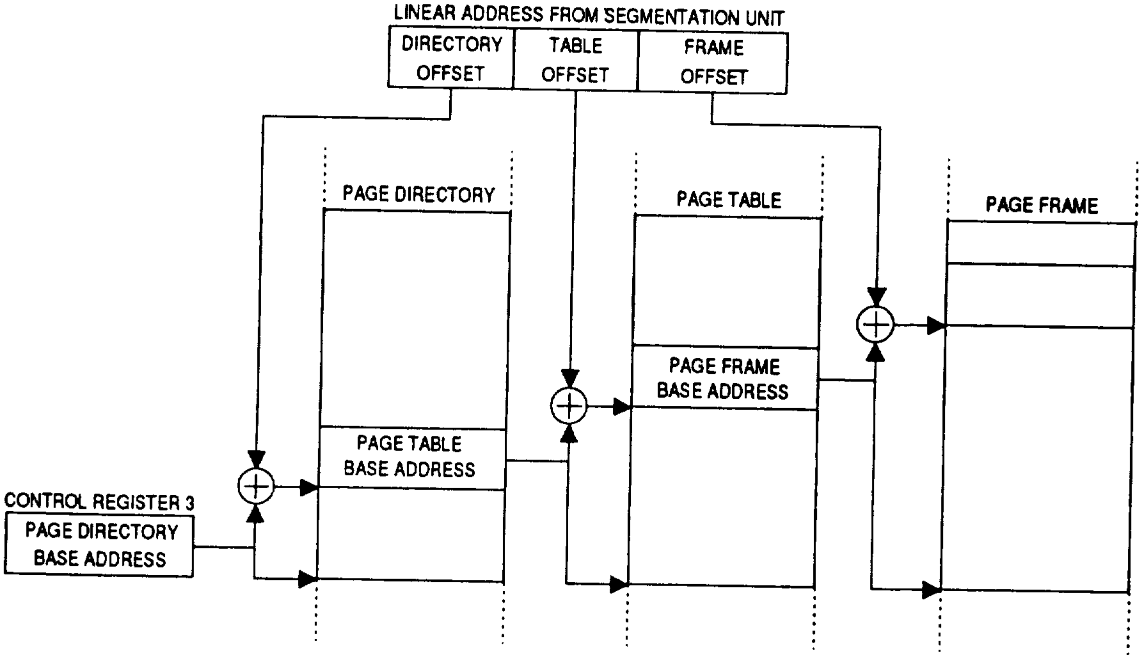


Figure 4. Page Translation

Should the CR3 experience a single event upset, the current task would lose track of all page tables and do anything from causing a protection violation to crashing the entire operating system. The amount of damage caused to the system would depend largely on the privilege level of the task that experiences the error and the amount of "intelligence" built into the operating system to detect and, if possible, correct such occurrences.

As an example, the operating system, which operates at the highest privilege level (i.e., ring 0), could easily crash if it lost track of some, or all, of its data and code. On the other hand, if an application running at the lowest privilege (i.e., ring 3) lost its data and code, that application would eventually cause an exception that would have to be handled by one of the operating system's exception handlers. The operating system could be unaffected if the protection violation initiated a task switch which caused the CR3 to be reloaded with a valid address from the fault handler's Task State Segment (TSS - a type of segment saving processor state information needed to restore a task). If the protection fault does not cause the CR3 to be reloaded, which would occur for example, if the fault handler were called through a call gate, the operating system itself would become subject to a corrupted CR3.

Because all page lookups are cached (see TLB, below), the possibility arises that all page information for a given task would reside in the TLB. This would happen if the sum of all pages were no greater than 128 kbytes (32 pages x 4 kbytes per page). In this case the system would be unaffected because CR3 would eventually be refreshed from that task's TSS upon successive task switches.

All 32 bits of CR3 were tested with either an all ones, all zeros, checkerboard (AAAAAAAAh), or inverse checkerboard (55555555h) pattern. While testing, the pattern was present in CR3 eighty percent of the time. The other twenty percent of the time was devoted to initializing the processor after reset and executing the actual code necessary to get information into and out of CR3. It should be noted that because the page directory must be page aligned, the 12 least significant bits of CR3 are assumed to be zero by the 80386DX and are ignored.

TRANSLATION LOOKASIDE BUFFER (TLB)

The TLB is a four-way set-associative 32-entry cache that buffers page table accesses. Without this buffering, the 80386's paging hardware would have to access two levels of tables for each memory access, an unacceptable performance penalty. Each entry in the TLB holds the upper twenty bits of a linear address, the upper twenty bits of the associated physical address and some tag bits used for data protection and cache implementation. The linear address field and tag bits are implemented in content-addressable memory (CAM) while the physical address field is implemented using random access memory (RAM).

One cache entry from each of the four sets was tested. All forty bits of the linear and physical address fields from this entry were tested using either an all ones, all zeros, checkerboard (AAAAAAAAh), or inverse checkerboard (55555555h) pattern. During testing, the pattern was present in the cache entry eighty percent of the time. The other twenty percent of the time was devoted to initializing the processor after reset and executing the actual code necessary to get information into and out of the entry. Of the approximately 1600 bits, 160 bits were tested for SEU.

GLOBAL DESCRIPTOR TABLE REGISTER (GDTR)

The GDTR is a system level register that holds the 32-bit linear base address and 16-bit limit of the Global Descriptor Table (GDT). The GDT is a data structure in memory that holds all non-interrupt related *segment descriptors* that are global to the system. A segment descriptor is an eight-byte data structure which provides the 80386DX with information about the type, size and location of a memory segment, as well as necessary segment attributes. The individual descriptors are selected by an address offset stored in a segment selector register. Segment selectors can only point to certain types of descriptors to which they are matched. As an example, the Task Register (TR) must always point to a TSS. In any 80386DX protected-mode operating system there must be at least three descriptors defined in the GDT: null, code, and data descriptors. LynxOS, like UNIX, does not support a segmented memory architecture and therefore would not likely have a need to define a great number of descriptors in the GDT.

Should the 16-bit limit portion of the GDTR experience an SEU, the result would depend on whether this 16-bit value was increased or decreased. Should the limit be increased, the GDT would possibly alias, or overlap, another segment(s) in memory. This aliasing would have little or no effect on the system because neither the portion of the operating system that maintains the GDT or the task(s) that owned the aliased segment(s) would know about this new limit. The task that owned the aliased segment would just continue using it as it had before the SEU. The operating system generally would not write new GDT entries in this aliased area because the maximum GDT limit would be determined once, at boot time, and never exceeded. Even if the operating system had a provision to dynamically allocate GDT space, it would have to first check that it was not going to try to allocate memory that belonged to another segment. At this time the operating system would determine that it would indeed overlap another segment and either move the "offending" segment out of the way or relocate the GDT to a larger portion of memory.

If the GDT limit were decreased and then a task attempted to use a selector value beyond the new limit, one of two possible types of exceptions would be generated: a General Protection Exception or an Invalid TSS Exception. The latter exception would occur only during a task switch when the 80386DX would normally try to validate the new TSS. If the operating system were SEU-aware it might incorporate some code in these two exception handlers that would compare the GDTR limit with the correct value that would be kept in memory. The operating system in this case would be able to recover gracefully, with no system impact. If the operating system were not SEU-aware, the operating system would lose all segments associated with the "lost" descriptors. Although it is easy to imagine the system crashing under such circumstances, the actual amount of damage to the system would depend directly on the number and criticality of the lost segments.

The other portion of the GDTR, the 32-bit base address, is extremely important to an operating system because it defines the GDT location in memory. If the GDT were to become misaligned, or lost altogether, the system would be put into a state of chaos and eventually crash.

All 32 bits of the GDTR's base address portion were tested with either an all ones, all zeros, checkerboard (AAAAAAAAh), or inverse checkerboard (55555555h) pattern. While testing, the pattern was present in the GDTR eighty percent of the time. The other twenty percent of the time was devoted to initializing the processor after the reset and to executing the actual code necessary to get information into and out of the GDTR.

INTERRUPT DESCRIPTOR TABLE REGISTER (IDTR)

The IDTR is a system level register that holds the 32-bit linear base address and 16-bit limit of the Interrupt Descriptor Table (IDT). This IDT, like the GDT, is a data structure in memory that holds segment descriptors that are global to a system. Unlike the GDT, the IDT can only contain interrupt or exception related segment descriptors.

The IDT associates each exception or interrupt vector with a segment descriptor for the task or procedure which services the associated event. To look up the corresponding descriptor, the processor scales the interrupt or exception vector by eight, the number of bytes per descriptor, this value is then compared against the IDT limit. If this value does not exceed the limit, it is added to the base address to form the 32-bit linear address of the desired descriptor. If this value does exceed the limit, the processor will enter shutdown mode immediately.

Should the 16-bit limit portion of the IDTR experience an SEU, the result would depend on whether the 16-bit value increased or decreased. Should the limit increase, the IDTR might alias to one or more different segments in memory. Aliasing would have little or no effect on the system because neither the portion of the operating system that maintains the IDTR or the task(s) that owned the aliased segment(s) would know about this new limit. The task that owned the aliased segment would continue using it, just as if the SEU had not occurred. The operating system generally would not write new IDT entries in this aliased area because the maximum IDT limit would be determined once, at boot time, and never exceeded. Even if the operating system had a provision to dynamically allocate IDT space, it would have to first check that it was not going to try to allocate memory belonging to another segment. At this time, the operating system would determine that it would indeed overlap another segment and either move the "offending" segment out of the way or relocate the IDT to a larger portion of memory. On the other hand, if the IDT limit were decreased and the system was called upon to service an interrupt whose segment descriptor lay outside the limit, the 80386 would immediately enter shutdown mode, hanging the system.

If the 32-bit base address portion of the IDTR were subject to an error, the IDT would become misaligned or lost altogether, the operating system would not be able to handle interrupts or faults and would eventually crash.

All 32 bits of the IDTR's base address portion were tested with all ones, all zeros, checkerboard (AAAAAAAAh), or inverse checkerboard (55555555h) patterns. During testing, the pattern was present in the IDTR eighty percent of the time. The other twenty percent of the time was devoted to initializing the processor after reset, and executing the actual code necessary to move information into and out of the IDTR.

EXTENDED AX REGISTER (EAX)

The EAX register is one of eight general purpose 32-bit registers on the 80386DX. These registers are used to hold operands for logical, arithmetic and address calculations. The EAX is one of the most widely used registers in the 80386, and as such may contain anything from a system-critical initialization procedure pointer, a seldom-used non-critical flag or no useful information whatsoever during the occurrence of an SEU. The operational result of an SEU occurring in the EAX is, therefore, extremely unpredictable, but has a high probability of not being catastrophic. Because the CPU mostly executes tasks that are at high protection levels (i.e., higher ring levels), any serious error caused by an SEU will be trapped by the operating system and cleaned up.

All 32 bits of the EAX register were tested with all ones, all zeros, checkerboard (AAAAAAAAh), or inverse checkerboard (55555555h) patterns. During testing, the pattern was present in the EAX register eighty percent of the time. The remaining twenty percent was devoted to initializing the processor after reset, and executing the actual code necessary to move information into and out of the EAX register.

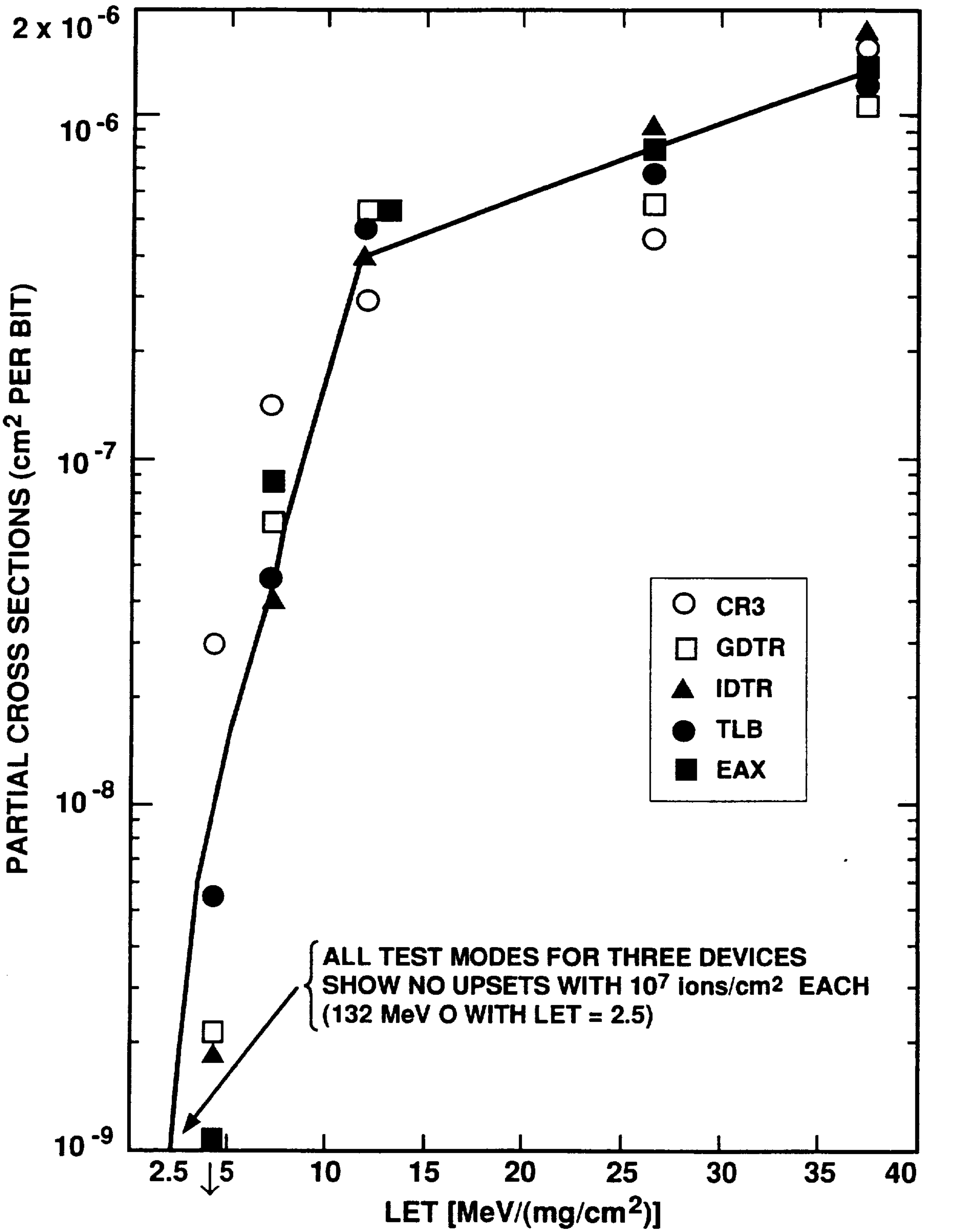
TEST RESULTS

1) Heavy Ions

The CHMOS IV Intel 80386 was tested at the Brookhaven National Laboratory (BNL) in May and July 1991. It received a partial characterization at room temperature at 25 MHz (CLK2 input) of five function blocks (see Table 1) for ions spanning a LET range of 2.5 to 60 MeV/(mg/cm^2). All data were taken with ions at normal beam incidence (theta=0), because the package configuration does not permit testing at oblique angles. The data, averaged over two or more of the test parts (S/N 1259, 1260, 1261 and 1262) and normalized to cross section per bit for each function block, are given in Figure 5 for most of the seven test ions shown in Table 2. It was found during the test that all four test sets of the TLB exhibited an identical soft error response independent of "all 1's" and "all 0's" test modes, so the plot condenses that result. When a flux comparison (10^4 vs. 10^5 ions/cm^2) showed no difference in SEU or SEL results (as expected), most of the remaining data were taken in the checkerboard (CB) mode at the higher flux (10^5 ions/cm^2 per second for a total of 10^6 ions/cm^2 in ~10 second runs).

JPL INTEL 80386 SEE TEST DATA

FUNCTIONAL BLOCKS (CB) MAY AND JULY 1991



ALL TEST MODES FOR THREE DEVICES SHOW NO UPSETS WITH 10^7 ions/cm2 EACH (132 MeV O WITH LET = 2.5)

Figure 5

The data of Figure 5 show no significant difference in either partial cross section or soft error threshold among the five function blocks tested. The data for LET=2.5 showed no soft errors for any of the five function blocks for each of two devices (S/N 1261 and 1262) for 10^7 132 MeV O ions/cm^2 per run. Data for LET=60 showed only latchups with large currents in the 300 mA to 942 mA range, which obscured any soft errors. (NOTE: the latchup cross section was ~$6x10^{-5}$ cm^2 per device for 320 MeV I.) The data at LET=40 showed soft errors as well as an occasional latchup, so it is very near the latchup threshold. One device was tested at 5.5 V and 4.5 V bias, with no significant difference in response to the two bias levels. Note that the observed high current latchups were of the classic, potentially catastrophic type caused by a single particle, completely independent of flux, rendering the device non-functional. It should be noted that S/N 1259 was destroyed during testing due to latchup. Classic latchups are not to be confused with small-current, "micro-latchup" phenomena observed by others.

The data may be summarized as: latchup threshold LET=40 MeV/(mg/cm^2) with 5.5 V bias and soft error threshold LET=3.5 ± 1 MeV/(mg/cm^2) with 4.5 V bias, all at ambient temperature.

A Comparison -- Tom Scott[4] was on-site for the May test. He has compared his IBM data to JPL's for the EAX and TLB registers, and reports that the results from the two organizations are nearly identical. He concurs that JPL's CR3, IDTR, and GDTR tests are an important addition to the microprocessor characterization -- in part, because of the critical nature of upsets that occur for those functions.

Table 1. Intel 80386 Function Blocks Tested

Function Block	Bits Tested out of Total
EAX - Extended AX, general purpose register	32 out of 32
*GDTR - Global Descriptor Table register	32 out of 48
*IDTR - Interrupt Descriptor Table register	16 out of 48
*CR3 - Control Register 3, Page Directory	32 out of 32
TLB - Translation Lookaside Buffer:	
Set 0	40 out of 1632
Set 1	40 out of 1632
Set 2	40 out of 1632
Set 3	40 out of 1632

*Heretofore untested function blocks.

NOTE: An "all 1's", "all 0's", checkerboard (CB) and inverse CB are available for all function blocks.

[4] Tom Scott, Private Communication, 1992.

Table 2. Intel 80386 -- BNL Test Ions (May and July 1991)

LET [MeV/(mg/cm^2)]	Ion and Energy	Range (microns)
2.5	132 MeV O	150
4.5	51 MeV O	38
7.6	194 MeV Si	81
13	153 MeV Cl	44
24	255 MeV Fe	44
40	285 MeV Br	37
60	320 MeV I	33

2) Protons

Proton data were taken at the Harvard Cyclotron in June 1991. Proton energies were 160 MeV (an undegraded beam used for only a few early measurements) and a set of 148, 90 and 50 MeV energy levels obtained by passing the beam through an appropriate degrader. The lower energy beams yield a larger more uniform beam cross section at the expense of a ten- to twenty-fold reduction in beam flux, and are therefore desirable only when test conditions can accept the lower flux (fluence) level.

The test results for five parts are summarized in Table 3. In the first run with part S/N 1263, 1.4x10^{11} 160 MeV p/cm^2 were delivered to the "all-ones" mode of CR3, causing two "not-AOK" soft errors. The "not-AOK" error designation refers to upsets of some susceptible bits located outside the tested function block. A subsequent run of this part for the same function configuration induced device failure (evidenced by tester runaway) after 5.7x10^{10} p/cm^2 for a total of 2.0x10^{11} p/cm^2 --- or 15 Krads. This dose is somewhat lower than the range of the total ionizing dose (TID) tolerance of 20 to 24 Krads determined from IBM's Gamma environment[5]. For subsequent runs, the proton fluence was limited to 10 to 15 Krads, and there were no further TID device failures. Because of these constraints on total device fluence (~10^{11} p/cm^2), there were also very few errors, and the statistics do not determine whether they are occurring randomly among the five tested function configurations shown in Table 1.

[5] Ibid, footnote 1.

Table 3. Summary of Proton Soft Errors for Intel 80386

S/N	Total Fluence (10^{11} p/cm^2)	Energy (MeV)	Not AOK	SEUs	Pattern	Test Mode	Failure Site
1263	2.0	160	2	0	Ones	CR3 only	Unknown
1259	1.4	160	1	1	Ones	All 5	GDTR
1260	1.13	148	0	0	CB	All 5	---
1260	1.13	148	0	1	CB	CR3 only	CR3
1261	1.36	148	0	0	CB	All 5	---
1262	1.7	148	1	0	CB	All 5	Unknown

Conclusion - The approximate overall cross section is calculated by adding up all the errors and fluence for all devices given in Table 3. Thus:

6 errors / {(average ~50 bits) x 8.72×10^{11} p/cm^2}

or cross section = 1.4×10^{-13} cm^2 per bit (148 MeV protons).

SUMMARY

Testing of the 80386DX microprocessor for single event effects (SEE) has been discussed, along with the importance of this device to NASA's MSREP program and individual flight projects, such as Space Station Freedom. The capabilities of this device, as well as its widespread installed user base and available development tools make it highly desirable as one of the core elements in sophisticated flight computers, and for complex embedded controller applications.

JPL's SEE testing shows that the 80386DX SEL threshold with heavy ions occurs at LET=40 MeV/(mg/cm^2). The SEU threshold LET=3.5 ±1 MeV/(mg/cm^2), and the soft error cross sections exceed 10^{-6} cm^2 (100 square microns) per bit at LET=40.

A test with 148 MeV protons, representative of trapped proton and flare proton energies, gives an SEU cross-section = 1.4×10^{-13}cm^2 per bit.

All tests were conducted at ambient temperature and worst case bias conditions.

APPENDIX A

KNOWN 80386DX REGISTERS

SYSTEM LEVEL REGISTERS

REGISTER NAME	# ACTIVE BITS / REGISTER WIDTH	COMMENTS
Global Descriptor Table Register (GDTR)	48/48	Address and limit of the GDT
Interrupt Descriptor Table Register (IDTR)	48/48	Address and limit of the IDT
Local Descriptor Table Register (LDTR)	16/16	Points to LDT descriptor entry in the GDT
Local Descriptor Table Segment Descriptor	48/48	Cached LDT descriptor entry in the GDT
Task State Segment Register (TR)	16/16	Points to TSS descriptor entry in the GDT
Task State Segment Descriptor	48/48	Cached TSS descriptor entry in the GDT
CS Segment Descriptor Cache	73/74	Cached CS descriptor entry in the GDT
DS Segment Descriptor Cache	72/74	Cached DS descriptor entry in the GDT
ES Segment Descriptor Cache	72/74	Cached ES descriptor entry in the GDT
FS Segment Descriptor Cache	72/74	Cached FS descriptor entry in the GDT
GS Segment Descriptor Cache	72/74	Cached GS descriptor entry in the GDT
SS Segment Descriptor Cache	73/74	Cached SS descriptor entry in the GDT
Control Register 0 (CR0)	6/32	System level control / status register
Control Register 1 (CR1)	0/32	Not used - Intel reserved
Control Register 2 (CR2)	32/32	Page fault linear address
Control Register 3 (CR3)	20/32	Page directory base address
Debug Register 0 (DR0)	32/32	Linear breakpoint address 0
Debug Register 1 (DR1)	32/32	Linear breakpoint address 1
Debug Register 2 (DR2)	32/32	Linear breakpoint address 2
Debug Register 3 (DR3)	32/32	Linear breakpoint address 3
Debug Register 4 (DR4)	0/32	Not used - Intel reserved
Debug Register 5 (DR5)	0/32	Not used - Intel reserved
Debug Register 6 (DR6)	7/32	Breakpoint status
Debug Register 7 (DR7)	27/32	Breakpoint control
Test Control Register (TR6)	28/32	TLB test control
Test Status Register (TR7)	23/32	TLB test status
Translation Lookaside Buffer (TLB)	1632/1632	Paging cache

APPLICATION LEVEL REGISTERS

REGISTER NAME	# ACTIVE BITS / REGISTER WIDTH	COMMENTS
Extended AX Register (EAX)	32/32	
Extended BX Register (EBX)	32/32	
Extended CX Register (ECX)	32/32	
Extended DX Register (EDX)	32/32	
Extended Source Index Register (ESI)	32/32	
Extended Destination Index Register (EDI)	32/32	
Extended Base Pointer Register (EBP)	32/32	
Extended Stack Pointer Register (ESP)	32/32	
Extended Instruction Pointer Register (EIP)	32/32	Points to next code segment instruction
Extended Flags Register (EFLAGS)	32/32	Application level control / status register
Code Segment Register (CS)	16/16	Code segment descriptor offset in the GDT
Data Segment Register (DS)	16/16	Data segment descriptor offset in the GDT
E-Space Segment Register (ES)	16/16	Data segment descriptor offset in the GDT
F-Space Segment Register (FS)	16/16	Data segment descriptor offset in the GDT
G-Space Segment Register (GS)	16/16	Data segment descriptor offset in the GDT
Stack Segment Register (SS)	16/16	Stack segment descriptor offset in the GDT

HARDWARE LEVEL REGISTERS

REGISTER NAME	# ACTIVE BITS / REGISTER WIDTH
Math Coprocessor Instruction Pointer	48/48
Math Coprocessor Data Pointer	48/48
Math Coprocessor Opcode	11/11
Linear Feedback Shift Register (LFSR)	207/207
Barrel Shifter	64/64
Pre-fetch queue	128/128
Instruction queue	96/96
Address bus latch	34/34
Data bus latch, output	32/32
Data bus latch, input	32/32

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 93-12	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle Test Report for Single Event Effects of the 80386DX Microprocessor	5. Report Date February 15, 1993 6. Performing Organization Code
7. Author(s) R. Kevin Watson, Harvey R. Schwartz, Donald K. Nichols	8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109	10. Work Unit No. 11. Contract or Grant No. NAS7-918 13. Type of Report and Period Covered JPL Publication
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546	14. Sponsoring Agency Code RE156 BK-506-59-11-01-00

15. Supplementary Notes

16. Abstract

The Jet Propulsion Laboratory Section 514 Single Event Effects (SEE) Testing and Analysis Group has performed a series of SEE tests of certain strategic registers of Intel's 80386DX CHMOS IV microprocessor. Following a summary of the test techniques and hardware used to gather the data, we present the SEE heavy ion and proton test results. We also describe the registers tested, along with a system impact analysis should these registers experience a single event upset.

17. Key Words (Selected by Author(s))	18. Distribution Statement Unclassified; unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 26	22. Price