A Field Failure Analysis of Microprocessors used in Information Systems

Syed Z. Shazli, Mohammed Abdul-Aziz, Mehdi B. Tahoori, David R. Kaeli
Department of Electrical and Computer Engineering,
Northeastern University
Emails:{sshazli,mabdul,mtahoori,kaeli}@ece.neu.edu

Abstract

Soft errors due to cosmic particles are a growing reliability threat to information systems. In this work, a methodology is developed to analyze the effects of single event upsets (SEU) and obtain Failure In Time (FIT) rates for commercial server microprocessors in live information systems. Our methodology is based on data collected from error logs and error traces of the microprocessors collected from systems in the field. Soft errors are further localized within the microprocessor resources with the assistance of the machine check architecture.

1. Introduction

Information systems are employed in an increasing number of application areas, motivating research activity focused on reducing the time–to–market, improving performance, reducing power consumption and increasing reliability. This latter aspect has assumed an increasingly important role, especially when dealing with critical applications. As device geometries continue to shrink, microprocessors will experience an increasing number of soft errors. These bit flips can temporarily corrupt the data being processed. As a consequence, future designs will need to be able to detect and recover from transient errors caused by soft errors [5][10].

Soft errors are intermittent malfunctions of the hardware that are not reproducible. Single Event Upsets (SEUs) that cause soft errors are generated by cosmic particles, energetic neutrons and alpha particles hitting the surface of silicon devices [1]. As feature sizes shrink, the amount of charge per device decreases, and so a particle strike is much more likely to cause an error. Particles of lower energy, which are more abundant than high energy particles, will generate sufficient charge to cause a soft error. With the exponential increase in transistor counts (as predicted by Moore's Law), we expect to see a similar increase in error rates for unprotected systems [4]. As a result, it is expected that maintaining microprocessor error rates at acceptable levels will require specific design changes [2].

A key requirement of present day information systems is availability. Since most transient errors take the system to an invalid state, they can take several tens of minutes or hours on large systems to recover. Hence, these errors can considerably impact system availability [3].

In this paper, we focus on *soft error rate* (SER) estimation of microprocessors used in information systems by analyzing actual field data. This work is done based on our collaboration with a major industrial information computing system manufacturer. We present a case study of failure rate data for commonly used 32-bit server processors in different information systems. We have also analyzed the exact locations of SEUs within the microprocessor and compared it against our previous analytical models. This study is based on field data collected from thousands of commercial information systems over a three year period in various national and international locations.

The remainder of this paper is organized as follows. In Section 2, an overview of the microprocessor, along with its usage in the information system, is presented. Section 3 details the methodology used in this study to estimate the soft error rate for the microprocessor used in the information system. Some case studies highlighting the methodology used are presented in Section 4. Section 5 covers the analysis of possible SEUs occurring in the system along with computation of FIT rates. Finally, Section 6 concludes the paper.

2. Overview of the information systems

The class of information system under consideration in this study is a high-availability computing system equipped with multiple microprocessors. These information systems have hundreds of terabytes of drive capacity and can serve hundred connected hosts.

The internal bussing architecture provides for a high degree of redundancy so that a failure in any component on a link does not disconnect other components from the system. However, simultaneous failures in multiple components can lead to events which impact reliability and potentially reduce availability. We call these *Reliability Impact Events*

(RIEs). Such events may cause sudden interruptions in service and potentially result in data unavailability. Our analysis is based on studying such events in the error logs of live systems.

One of the components that interconnect to the buses is the *Logical Unit Module* (LUM). Each LUM contains one or two single-core state-of-the-art IA-32 microprocessors (the so-called "server processors"), depending on the particular system. The microprocessors used in these LUMs act as the main control center for the system. They are responsible for processing I/O requests from the attached host(s).

On powering up the system, each microprocessor boots up an operating system. The operating system manages all functions of the information system and supports some basic configuration operations. A load balancing utility running on the host schedules jobs between the two microprocessors. If a RIE appears in one of the microprocessors, the utility transfers all the jobs to the other LUM. The LUM is responsible for all data buffering and caching, though performs no processing on the stored data.

Since data is heavily protected with ECC, soft errors in LUMs cannot affect the data items and cause *silent data corruption* (SDC). Therefore, the occurrence of a RIEs (availability degradation) is the main impact of SEUs in these systems.

The microprocessor under study supports a *Machine Check Architecture* (MCA) as described in [3] and [11]. Issues related to the design of MCAs have been studied in earlier reliability work targeting server systems [8]. The MCA allows the operating system to detect, signal, and record information about selected machine fault conditions. Some of those faults are correctable, while others are uncorrectable (i.e., only detectable). The machine check mechanism is intended to enable system providers and system software developers to diagnose, isolate, and understand microprocessor failures. It is also intended to enable system recovery mechanisms to be employed. In the following, we give details of the two types of systems under study.

System Type A - In these systems, there are two server microprocessors on each LUM and two LUMs per system, for a total of four microprocessors per system. Two levels of on-chip caching are available on these microprocessors. Cache contains both copies of data in main memory and *tag*, to map to actual data in main memory. The level 1 (L1) data cache is parity protected (detection), while the Level 2 (L2) cache has ECC protection (detection and correction) on data and parity protection (only detection) on tags.

System Type B - In these information systems, there are dual microprocessors on each LUM, and two LUMs in each system (i.e., four microprocessors per system). As System B is a newer generation system than System A, it uses a more recent version of the server microprocessor. The server microprocessors in System B are fabricated using a newer technology node compared to those in System A. The drive capacity of System B is almost twice as that in System A. In the microprocessors used in System B, the size of L1 and L2 cache is twice the size of corresponding caches in microprocessors of System A. Similar to microprocessors used in System A, the L1 data cache is parity protected, while the L2 cache has ECC protection on data and parity protection on tags. The population (system-hour) of System B in the field is an order of magnitude smaller than System A.

3. Methodology

3.1 SEUs in the field

Field data analysis is traditionally used in industry to evaluate system failure rates, monitor system reliability, and conduct long-term trend analysis to identify the need for future design changes [6][9]. The value of any field study is highly dependent upon the quality and completeness of the event logs. If the information is missing, incomplete, or difficult to interpret, our ability to analyze the data is diminished.

As a figure of merit we consider *FIT*, which is defined as the number of failures per billion operating hours. The aim of this work is to calculate the FIT rate due to soft errors for the microprocessors used in the LUMs, based only on the SEUs observed in the field. Failure logs were obtained from the repository of field data from the manufacturer.

In the information systems studied in this work, we limit our focus on microprocessor RIEs that did not result in LUMs being replaced in the system, corresponding to one-time RIEs. We therefore investigated all of the possible causes of RIEs. Initially, all unscheduled RIEs occurring in the microprocessors contained in the LUMs were analyzed. A large number of unscheduled RIEs were identified. These can occur for one of several possible reasons:

a) Component failure
b) Software-based errors
c) Power issues
d) Radiation-induced errors.

The next step was to isolate those cases which could have occurred because of a soft error. We call these *Possible* soft errors.

Errors which were confirmed to have occurred inside the microprocessor using error logs and traces were labeled as *Probable* soft errors and are denoted by SEU_{CPU}. Most of these cases resulted in Machine Check exceptions. In some cases, we were able to identify the register in which a bit flip had occurred, with the help of error traces.

In some cases invalid memory addresses were generated by the microprocessor and resulted in a microprocessor RIE. If a RIE could not be linked to faulty software, and the traces did not point to a specific register in which the error occurred, we refer to these cases as $SEU_{INV\text{-}MEM}$. Since invalid addresses may still be generated because of a software bug, we do not include them in the Probable soft error bucket. Instead, we construct a *Potential* soft error bucket and include such cases in this category.

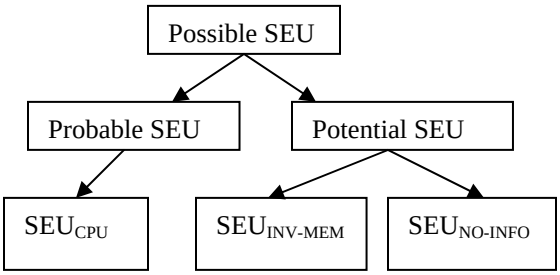


Figure 1 Classification of SEU events

The *Potential* soft error bucket also contains those cases of isolated RIEs in which the logs did not point to a specific fault in hardware or software. The error could not be confirmed to have occurred in the microprocessor or be attributed to an invalid memory address. Most of these errors were classified as *Potential* because some of the logs that were studied were incomplete. However, since these were isolated cases out of the total number of RIEs, they were a potential candidate for SEUs. We include these cases in a category called $SEU_{no\text{-}info}$. Our classification taxonomy is shown in Fig 1.

The following approach was taken to identify these cases:

i) Note the time and date of the RIE.
ii) Search the events in the microprocessor logs, for that incident.
iii) If the events preceding the incident can be classified as reasons component, software, or power related failures, do not include them in the *possible* SEU bucket.

Using the above criteria, we obtained a bucket containing *Possible* soft errors which had a significant number of incidents. Note that these are individual instances of RIEs which could not be linked to a hardware component, power supply issue, or corrupted software.

3.2 Localizing SEUs within the microprocessor

The Probable SEUs occurring inside the microprocessor were further investigated with the help of the Machine Check Architecture (MCA) information available to help to isolate just exactly where in the microprocessor they occurred. The machine check architecture is an internal architecture subsystem which detects and captures errors occurring within the microprocessor logic [3]. It handles 5 main subsystems.

- External Bus Logic
- Back-side Bus Logic
- Cache Unit
- Translation Look aside Buffer
- Instruction Fetch Unit

The machine check exception handler returns a 64-bit error code. Bits 0 to 15 contain the machine check error code, and bits 16 to 31 contain the model-specific error code. There are some reserved and informational bits and also some flags. The format of the 16-bit error code is as follows: `0000 0001 RRRR TTLL`. The description of the fields is as follows.

- The 2-bit TT sub-field indicates the type of transaction: data (00), instruction (01), or generic (10). A generic type is reported if the microprocessor is unable to determine the type of the transaction.
- The 2-bit LL sub-field indicates the level in the memory hierarchy where the error occurred: level-0 (00), level-1 (01), level-2(10), or generic (11). Again, the generic type is reported when the microprocessor cannot determine the hierarchy level.
- The 4-bit RRRR sub-field indicates the type of the action associated with the error. Actions include read and write operations, prefetches, cache evictions, and snoops.

4. Case studies

A significant number of RIEs were received due to invalid addresses being generated by the processor. In general, invalid memory accesses are typically caused by software faults. However, if the logs do not contain any further memory access violations while using the same software, we have included these incidents in the *Potential* SEU bucket. Similarly, some RIEs may have occurred because of power spikes. However, if the

logs did not show any power related problems on the system before or after the event, we included the incident in the *Potential* SEU bucket. In this section, we will present four examples to show how an isolated RIE can point to a soft error.

4.1 A probable SEU in the Instruction Pointer

We first examine a RIE that occurred in the microprocessor because of an invalid memory address. With the help of error traces, we were able to identify the events leading up to the incident. The following is the sequence of events prior to the incident.

- Function `ABC` was called just before the RIE occurred.
- Before calling `ABC`, the return address was pushed on the stack. The calling function was in the range `0xa5be8xxx – 0xa5beexxx`.
- The value of the Instruction Pointer (IP) at the time of RIE was `0xa7be8a8e`. This was an invalid address which resulted in the incident being signaled.
- Changing the IP from `0xa7be8a8e` to `0xa5be8a8e`, which lies within the range of the calling function, we can see that an invalid value was popped off the stack on the return from the function `ABC`. The true value differed from the actual value by one bit (the 7th bit from the left), and this was not identified to be a software bug. The incidence was included in the *probable* SEU bucket because the error was confirmed to have occurred inside the processor.

4.2 A bit flip in the address of a function

In another incident, an attempt was made to access an invalid memory address which resulted in a RIE. Stack traces were uploaded and the logs pointed to the following events prior to the incident:

- The last call before the incident references the address `0xf9a96652`. This is an actual function pointer, so this seems a valid value. Looking at the lines that should have been executed we find:

```
FunctionABC:
0xf9a96652  55    push ebp
```

The value at this address translates to `push ebp`. However, the value on the stack is not `ebp`.

- The address where the system incurred this error was `0xf9a96258`, which is not in the function.
- If we look at the instructions preceding the instruction where the system halted, we find:

```
0xf9a96252  53    push ebx
0xf9a96253  183b  sbb [ebx],bh
0xf9a96255  55    push ebp
0xf9a96256  d476  aam …
0xf9a96258  0884c90f851304
      or[ecx+ecx*8+0x413850f],al
```

The values in `ebx` and `ebp` match exactly the values on the stack.

- The last instruction at `0xf9a96258` referenced an invalid pointer that resulted in a RIE.
- We can see here that instead of going to `0xf9a96652`, the microprocessor went to the address `0xf9a96252`, which is a difference of 1 bit (the 11th bit from the right). Since the error is confirmed to have occurred inside the processor, we include this incident in the set of Probable SEUs.

4.3 A bit flip in the instruction cache

For a particular RIE, a machine check exception was generated and the error code returned was `0xBE00000020010152`. When we inspect the last 16 bits (`0000 0001 0101 0010`), we see that the 2 rightmost bits (10) denote that the error occurred in the L2 cache. The next 2 bits (00) indicate that the error was in the instruction cache. Code 0101 translates to a read error. Hence, we can conclude that a parity error occurred in the L2 instruction cache.

4.4 A potential SEU due to an invalid memory address

In one instance of a RIE, an invalid memory address was generated which led to a memory access violation. A software bug is generally the most probable cause of invalid memory accesses. However, in this RIE incident, there was no subsequent memory violation, even though the software running on the machine was not changed or upgraded. Since, the error was not reproducible and did not occur because of a specific software or hardware issue, we included it in the *Potential* SEU bucket in the $SEU_{INV\text{-}MEM}$ category.

5. Analysis of possible SEUs

Our study is based on field data gathered from all functioning systems of types A and B spanning installations in different locations, nationally and internationally. Data from several thousand systems operating for a total of more than half a billion hours was collected and analyzed

We initially looked at all unscheduled RIEs occurring in the microprocessors used in the LUMs. By looking at error logs and error traces, we tried to determine the root cause of each incident. Those incidents which resulted from a Machine Check exception were further investigated to identify the

location in the microprocessor where the error occurred. The exception handler returned a 64-bit error code. This error code was decoded to identify the failing unit inside the microprocessor. We present our findings in the following sections.

5.1 SEU distribution in the systems

There were a significant number of unscheduled RIEs that occurred in these systems. Over a hundred of these were classified as *Possible* SEUs. Figure 2 shows the statistics of *Possible* SEU events occurring in the two systems.

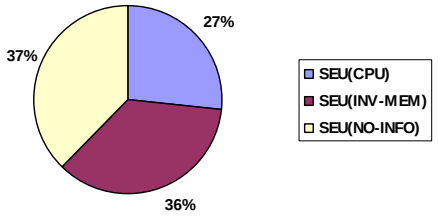
It was observed that some of the RIEs were due to events occurring inside the CPU (SEU_{CPU}). Other events were due to invalid memory addresses generated by the microprocessor ($SEU_{INV-MEM}$). Since these invalid addresses resulted in a single instance of a RIE and were not linked to faulty software, they were included in the *Potential* soft error bucket. A similar number of indeterminate cases were found where there was an isolated instance of a RIE, though it could not be linked to faulty hardware or software. In some cases, there was insufficient information in the logs, but since these were single instances of a RIE, and they were not linked to any software or hardware failure, they were included in the *Potential* SEU bucket. We group them as $SEU_{NO-INFO}$ in Figure 2.

5.2 SEU localization within the microprocessor

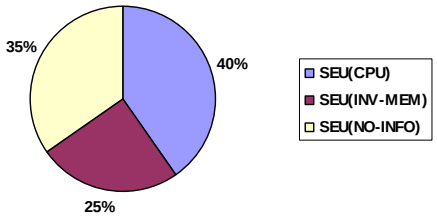
When an SEU is confirmed to have occurred inside the microprocessor, we classify it as a *Probable* SEU. For systems of type A we analyzed a statistically significant number of probable SEUs and identified the exact location of SEUs occurring within the microprocessor using the Machine Check Architecture information [11]. As mentioned in Section 2, the L1 cache in the microprocessor under study has parity protection. The L2 cache is an 8-way set associative cache with ECC protection on the data array and parity protection on tag array.

We performed an analytical study of vulnerabilities in cache memories in [7]. As mentioned in [7], ECC protection (detection and correction) reduces the effective vulnerability of the L2 data array to zero and the primary source of L2 vulnerability is the tag array (only detection). It was observed that the L2 tag vulnerability is greater than the vulnerability of L1 data and Instruction cache for most of the SPEC benchmarks when a similar processor and memory organization was considered.

To further investigate the vulnerability of the L2 tag addresses, the L2 tag vulnerability was profiled in terms of pseudo-hit vulnerability, replacement vulnerability, multi-hit vulnerability, and status vulnerability in [7]. The results showed that replacement vulnerability makes up almost 85% of the total tag vulnerability. This is due to the fact that tag-addresses become more susceptible once the first write occurs in the block and remain susceptible to SEUs until the block is replaced or flushed to lower levels of memory.



(a) System A



(b) System B

Figure 2 Breakdown of *Possible* SEUs in the two systems

The field analysis carried out in this work agrees with the (analytical) findings presented in [7]. With the help of the MCA given in [11], we further localized the sources of *probable* SEUs within the microprocessor, using the methodology presented in Sec. 3.2. It was observed that only 12% of the errors were found to have occurred in the L1 cache. This is primarily due to its small size. Almost 80% of the *probable* SEUs occurred on tag bits of the L2 cache. Additionally, there were some parity errors in the interconnect buses (which are only parity protected). The results are presented in Table 1.

Table 1 Breakdown of probable SEUs in System A

Parity on L2 Tag	80%
Data Parity on L1 Cache	12%
Interconnects	8%

5.3 Calculating FIT rates due to SEUs

The FIT rates were calculated separately for *Probable* SEUs and all *Possible* SEUs in the two types of systems. They were obtained by dividing the number of SEUs by the total run time of all systems. Run times were computed using the following formula:

$$running time = \sum_i system_i \times time\_in\_field_i$$

for $system_i$ in the field for $time\_in\_field_i$.

Table 2 Normalized FIT rates

	Normalized FIT (Probable SEU)	Normalized FIT (Possible SEU)
System A	1	3.68
System B	3.24	8.03

In this study, more than half a billion system-hours of information systems in the field were analyzed. System A has significantly higher systems-hours than system B. Table 2 shows the normalized FIT rates (to the *Probable* SEU FIT rate of system A) for these systems based on *Probable* and *Possible* SEUs (i.e., the lower and upper bounds). Due to extreme sensitivity of such data, we have only reported the normalized values instead of the actual FIT rates.

5.4 Analysis of Results

The FIT rate analysis of the microprocessor used in these systems, as presented in Table 2, suggests that the FIT rates of the microprocessors in System B are significantly (2-4 times) higher than System A.

Studying the architectures of Systems A and B, as presented in Section 2, shows that the on-chip cache of System B is twice the size of the cache in System A. Although the computed FIT rates are limited to particular RIE events and analysis of error logs, a comparative analysis can factor out common sources of error in the analysis. This data suggests that the system-level FIT rate of server microprocessors in this study has a direct relationship to the size of on-chip cache memory. In conclusion, the soft error susceptibility of the server microprocessors under study increases substantially when smaller feature sizes of transistors and more on-chip cache memories are used if soft error detection and protection capabilities remain the same.

6. Conclusions

In this study, we focused on the SEUs occurring in state-of-the-art microprocessors used in high performance information systems. Only system-level effects of soft errors were considered. These incidents, termed as Reliability Impact Events (RIE), can take several tens of minutes or more to service on large systems and can affect availability considerably. It is important to note that the issue under consideration is data un-availability rather than data corruption.

We looked into systems equipped with different numbers microprocessors per system and different on-chip cache sizes. We further localized the SEUs to those occurring in the microprocessor and those observed in memory. We found that the error trends found in particular memory structures inside the microprocessor agree with results obtained in earlier work using analytical models. Also, the size of on-chip cache memory has a direct impact on the FIT rate.

References

[1] S. Kim and A.K. Somani, "Soft Error Sensitivity Characterization for Microprocessor Dependability Enhancement Strategy," *Proc. of the Intl. Conference on Dependable Systems and Networks (DSN)*, pp. 416-425, June 2002

[2] S.S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," *Proc. of the Intl. Symp. on Micro-architecture (MICRO-36)*, pp. 29-40, 2003.

[3] A. Messer et. al., "Susceptibility of Commodity Systems and Software to Memory Soft Errors", *IEEE Transaction on Computers*, Vol. 53, No. 12, pp 1557, Dec 2004.

[4] R.C. Baumann, "Radiation induced Soft Errors in Advanced Semiconductor Technologies", *IEEE Transactions on Device and Materials Reliability*, Vol. 5, No. 3, pp 305-316, Sep 2005.

[5] C. Bolchini et. al. "A model of soft error effects in generic IP processors", *Proc. 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2005.

[6] M.F. Buckley et. al. "VAX/VMS event monitoring and analysis" *Proc. 25th International Symposium on Fault Tolerant Computing*, pp 414-423, June 1995.

[7] H. Asadi et. al. "Vulnerability Analysis of L2 Cache Elements to Single Event Upsets", *Proc. Design and Test Conference in Europe (DATE 2006)*, pp. 1276-1281, March 2006.

[8] K. Johnson, J. Demain, S. Prakashan, R. Welsch, A. Samarov, "The Use of Machine Check Architecture (MCA) to Perform Reliability Studies on Servers", *Proc. 2001 Quality and Productivity Research Conference*, Austin, Texas, May 2001.

[9] R. Velazco, R. Ecoffet, F. Faure, "How to Characterize the Problem of SEU in processors & Representative Errors Observed on Flight", *Proc. 11th IEEE International On-Line Testing Symposium (IOLTS'05)*, pp. 303-308, 2005.

[10] C. Weaver, J. Emer, S. Mukherjee, S. K. Reinhardt, "Techniques to reduce the soft error rate of a high-performance microprocessor", *Proc. 31st Annual International Symposium on Computer Architecture*, pp. 264-275, 2004.

[11] AMD64 Architecture Programmer's Manual Volume 2: System Programming, Advanced Micro Devices, www.amd.com, pp. 305-320, Sep 2003.