Silent Data Corruption and Embedded Processing with NASA's SpaceCube

Kenneth M. Zick, Chien-Chih Yu, John Paul Walters, and Matthew French

***Abstract*—Dramatic increases in embedded data processing performance are becoming possible using platforms such as the NASA SpaceCube. With a flexible architecture and commercial devices, selected computations can be tuned for the highest performance while giving up perfect data reliability. More needs to be known about the nature of silent data corruption in this paradigm. When it occurs, how pervasive is it? To what extent can it be mitigated while near-optimal performance is maintained? This paper provides new insights into these questions, via a fault emulation-based study of two disparate applications running on a hard-core embedded processor. Two very low-cost methods of data error detection reduce the worst type of SDC by 89–97% with a performance overhead of < 1%.**

***Index Terms*—Silent data corruption, onboard processing, radiation hardening-by-software, radiation-induced upsets.**

I. Introduction

HIGH-PEFORMANCE embedded computing is enabling paradigm shifts in space systems and many other domains. The ability to tune non-critical data processing for the highest performance can improve responsiveness and increase the amount of science data returned. However, the computations are often subject to significant noise and there remains a need for understanding and optimizing the threat of silent data corruption (SDC) in the returned data.

We have been tasked by NASA's Earth Science Technology Office to research Autonomous Onboard Processing for Sensor Systems (AOPSS). The platform of focus is NASA's SpaceCube, which is pushing the boundaries of high-performance by including commercial off-the-shelf devices. SpaceCube 1.0 [1] contains four embedded PowerPC 405 cores residing on two commercial Xilinx Virtex-4FX FPGAs. These hard cores provide several times higher performance than typical soft cores due to higher clock frequencies, more efficient caches, multiple instruction issue, etc. Previously, methods were developed for detecting and recovering from processor crashes and hangs [2]; a prototype design has been uploaded to the SpaceCube 1.0 currently flying aboard the International Space Station as a part of the extended MISSE-7 experiment [3]. Complementing those availability-related efforts, this paper focuses on the threat of silent corruption of computed results.

A goal of the research program is to achieve the highest levels of throughput, while keeping the overhead for data reliability features under 1%. Clearly out of the question are some common methods such as dual- or triple-modular redundancy, and full temporal redundancy. Furthermore, when using hard core embedded processors, we cannot leverage the many methods which involve changes to the processor hardware design.

This paper makes the following contributions:

- experimental results showing SDC probabilities on applications representing two extremes;
- novel data on the pervasiveness of corruption in individual SDC errors;
- an evaluation of the efficacy of two very low-cost data error detection methods.

II. Related Work

Estimating the threat of silent data corruption in microprocessor results has been addressed with fault simulation, fault emulation, radiation testing, and analytical approaches [4]. The associated task of data error detection and correction has been studied extensively; most approaches require hardware design changes [5] and are not applicable when using pre-designed hard processor cores. The works most relevant here use a software-based approach. Duplication of instructions was used to detect errors in an FFT application on the ARGOS mission [6]. A hybrid approach involving changes to both software and to field-programmable gate array (FPGA) logic is recommended in [7]; as with our study, the authors consider an FPGA-embedded PowerPC 405. However, the overhead for data error detection is high since variables are replicated and compared using redundant memory writes. Moreover, the rate of false positives is very high. A recent paper describes a software-based method for graphics processors and demonstrates 87% detection of data corruption for 15% overhead [8].

III. Fault and Error Emulation

In this study, the fault model is a single-bit, single event upset that occurs in a register or in one of the caches on the PowerPC 405 processor. Such upsets can be caused by ionizing radiation. Note that cache upsets are important with

Manuscript received January 4, 2012.

K. Zick, J. P. Walters, and M. French are with the University of Southern California Information Sciences Institute, Arlington, VA 22203 USA (phone: +1 703-812-3711; fax: +1 703-812-3712; e-mail: kzick@isi.edu).

C.-C. Yu is with the University of Michigan, Ann Arbor, MI 48109 USA (e-mail: ccyu@umich.edu).

the SpaceCube 1.0 since the parity circuitry for the Virtex-4 PowerPC 405 cores is not operational [9]. Given the fault model, the corresponding error model is silent data corruption in which a system completes a computation but returns an undetected incorrect result. Crashes and hangs are handled with other means [2] and are not studied here.

We perform reliability studies by running an application-under-test on a PowerPC 405 processor embedded in an FPGA and simultaneously injecting faults with an interrupt-based methodology (extension of [10]). We use an ML410 FPGA board which contains a Virtex-4FX60 just as with the SpaceCube 1.0. A second processor on the FPGA checks the computed results and logs the error behavior for post-processing. A diagram of the experimental setup is shown in Fig. 1.

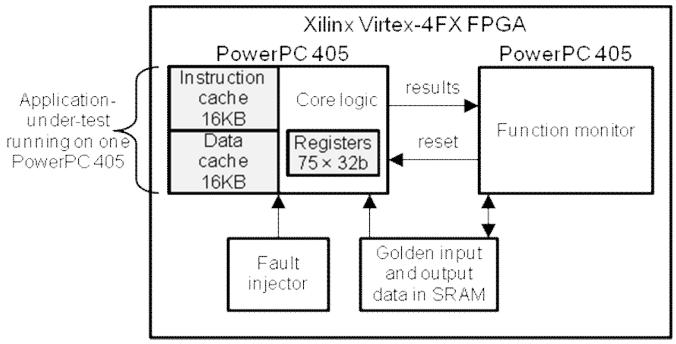

Fig. 1. Diagram of experimental setup. Faults are injected into software-writable elements (shaded) of a PowerPC 405 embedded processor.

One metric of interest is the probability that a fault will lead to an SDC error (*SDC probability per fault*). Such a metric can be combined with fault rates to determine an overall probability or rate of SDC. A further dimension of analysis that is rarely studied but that we recommend is the *pervasiveness* of corruption associated with individual errors; we will measure this in terms of the number of corrupted words in a computed result. Corruption that is very pervasive is often the most problematic for embedded processing applications since it can affect multiple computations; the special case of corruption exceeding a threshold number of data words will be called *unbounded* corruption.

IV. Low-Cost Data Error Detection

We take a new look at two straightforward software-based methods of data error detection that together satisfy the restriction of 1% runtime overhead. The first is *range checking* of processed data. In many algorithms, the data at some step must adhere to a valid range of values. Application programmers may be aware of some of these ranges, but typically do not create soft error-aware checks, nor can they be expected to. Applications can be profiled to determine the appropriate ranges and, in concert with fault emulation, range checking code can be inserted. The constants representing the ranges (e.g., a min and max value) can be replicated in the code to mitigate errors in the checking itself.

Embedded data processing applications are often iterative; certain instructions, constants, and variables are re-used as the processor loops over large data structures. Local copies of these re-used state elements often reside in a register or cache, and if the local copy incurs an error, corruption of output data can occur on an ongoing basis. To detect these errors before the offending value escapes from the register or cache, applications can perform an extra iteration through a loop but with a known set of input and output values. We refer to this as *application built-in self-test* (application-BIST). A failing self-test indicates either ongoing corruption or an error in the self-test computation itself (false positive). The method is most appropriate for loops with many iterations such that the self-test causes negligible memory and execution time overhead.

The data error detectors can in principle be used to trigger an error recovery mechanism such as a reset or possibly a rollback to a previous checkpoint. Error information can also be provided in telemetry for use in data post-processing.

V. Experimental Results

We conducted data reliability studies using two representative applications very different in character. The first is a fast Fourier transform (FFT) which is common in onboard processing and is a part of our MISSE-7 experiment on the International Space Station. Errors occurring during an FFT computation often spread to multiple words of the output signal. Our FFT is one-dimensional, complex-numbered, and has 128 points. The inputs and outputs of each computation consist of 128 pairs of single-precision floating-point values. The input is an arbitrary signal as on our MISSE-7 experiment. The second application involves hyperspectral image processing, a powerful technique used on EO-1 mission with the Hyperion instrument [11] and on the upcoming HyspIRI mission. Specifically, we use the transform function from an application that corrects for the effects of Earth's atmosphere [12]. This atmospheric correction computation will be called AtCorr. The inputs and outputs of each computation consist of 256 pixels where each pixel value is a single-precision floating-point number. As opposed to the FFT, the 256 portions of the computation are largely independent from each other, though they do depend on some shared state. As input we use a portion of image data from Hyperion. Each application is implemented in C and has been compiled to the PowerPC 405. Floating-point operations are emulated in software since the embedded hard-core processor under study does not have a built-in floating-point unit.

Fault injection campaigns were conducted for each application, with roughly 40,000 trials per scenario. During each trial, the computations were repeated over a period long enough for "heartbeats" [2] to occur, while a fault was injected at a random time and location. This period was roughly 10 seconds (600 computations). The golden input and output data were kept in a separate read-only memory protected from corruption. The computed data was checked by an

independent processor after each computation; after the entire period, the application was restarted and the process repeated.

The ratio of observed SDC errors to the number of faults injected was used as an estimate of the SDC probability. Vulnerability to register faults and cache faults were analyzed separately. The number of injectable register cells was 75 registers × 32 bits = 2400, while the number of injectable SRAM cells was 284,672 (512 cache lines times 279 I-cache cells and 277 D-cache cells).

A. *SDC Probability per Fault*

We tallied the SDC probability for four cases: the baseline with error detectors turned off, with each detector alone, and with both detectors. For register upsets, the range checking method detected 31% (FFT) and 28% (AtCorr) of data corruption. Application-BIST provided little value in this case since the errors tended to be bounded (see section IV.B) and thus left no traces that could be easily caught by BIST. The results for register upsets are shown in Figures 2 and 3.

For cache upsets, application-BIST detected 58% (FFT) and 34% (AtCorr) of data corruption errors. Unlike register data, the lifetime of data in the caches is often long enough to be amenable to periodic self-test. Using both methods of error detection, the SDC probability is reduced by 65% (FFT) and 58% (AtCorr). Results are illustrated in Figures 4 and 5.

B. *SDC Pervasiveness*

SDC errors were classified into three bins according to the pervasiveness of corruption: single word errors, multi-word errors affecting one 256-word result, and unbounded corruption affecting multiple computations.

For register upsets in the FFT, the vast majority of errors affected more than one word, due to the highly-correlated nature of the FFT algorithm. AtCorr, on the other hand, involves independent pixel calculations and indeed single-word errors were common. Results are shown in Tables I and II. With both applications, error detection was most effective for unbounded errors; though these errors were small in number, they are also the most problematic, so mitigation is important.

SDC errors associated with cache upsets had a different character. In the baseline cases, single-word errors were the smallest class, and unbounded errors were a significant portion of all errors – 60% (FFT) and 37% (AtCorr). This is largely due to instruction cache upsets, which tend to persist and lead to unbounded corruption. As before, the error detection was most effective for unbounded errors, and since these were common, the overall decrease in SDC reached 65% (FFT) and 58% (AtCorr).

C. *Overhead*

The execution time overhead of data error detection was measured to be < 0.25% for each application. The range checking code is minimal, and the application-BIST can be run infrequently (in our case, one extra loop iteration after every 600 iterations). A second type of overhead can occur when an error detector itself is corrupted by a soft error and falsely reports a data error. The measured frequency of these false positives during the fault emulation campaigns is shown in the Table V.

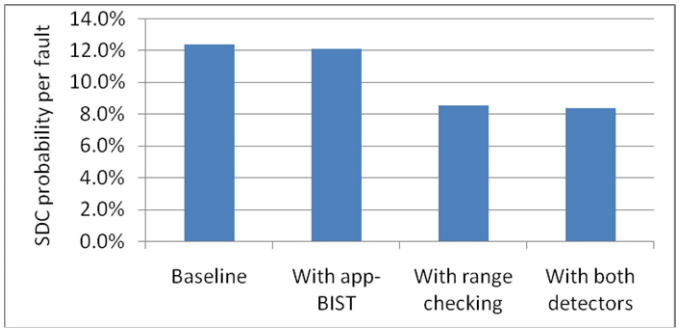

Fig. 2. SDC probability for register upsets in FFT application.

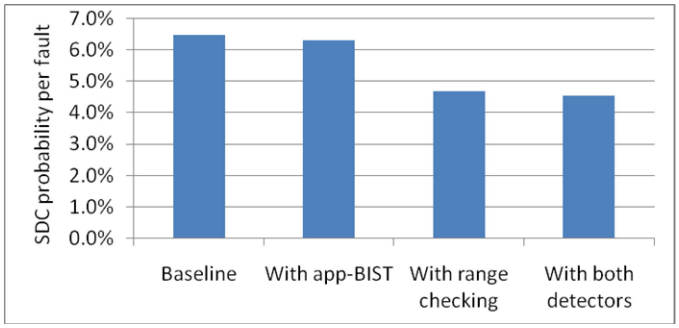

Fig. 3. SDC probability for register upsets in AtCorr kernel.

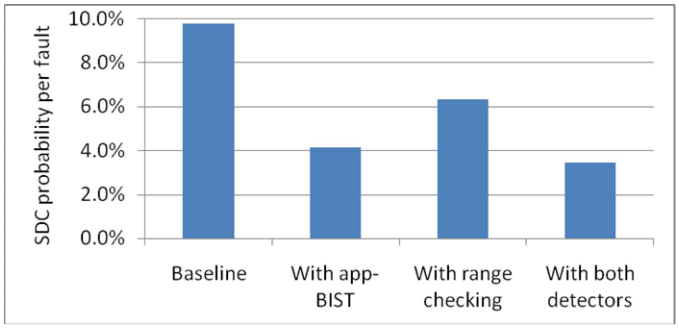

Fig. 4. SDC probability for cache upsets in FFT application.

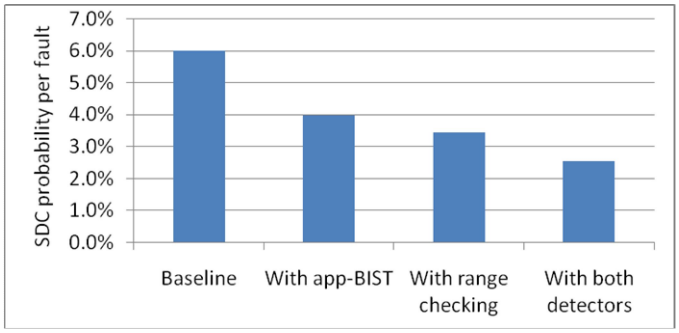

Fig. 5. SDC probability for cache upsets in AtCorr kernel.

TABLE I
PERVASIVENESS OF SDC ERRORS – FFT APPLICATION, REGISTER UPSETS

Number of corrupt words	Baseline	With error detectors	Relative reduction
1	0.47%	0.40	14%
2 – 256	11.4%	7.9%	31%
Unbounded (> 256)	0.53%	0.056%	**89%**
Total	12.3%	8.3%	33%

TABLE II
PERVASIVENESS OF SDC ERRORS – ATCORR KERNEL, REGISTER UPSETS

Number of corrupt words	Baseline	With proposed error detectors	Relative reduction
1	6.3%	4.5%	28%
2 – 256	0.013%	0.011$	20%
Unbounded (> 256)	0.19%	0.008%	**96%**
Total	6.4%	4.5%	30%

TABLE III
PERVASIVENESS OF SDC ERRORS – FFT APPLICATION, CACHE UPSETS

Number of corrupt words	Baseline	With error detectors	Relative reduction
1	0.193%	0.186%	4%
2 – 256	3.73%	3.05%	18%
Unbounded (> 256)	5.85%	0.193%	**97%**
Total	9.77%	3.43%	65%

TABLE IV
PERVASIVENESS OF SDC ERRORS – ATCORR KERNEL, CACHE UPSETS

Number of corrupt words	Baseline	With proposed error detectors	Relative reduction
1	1.0%	0.78%	22%
2 – 256	2.76%	1.53%	45%
Unbounded (> 256)	2.25%	0.23%	**90%**
Total	6.0%	2.54%	58%

TABLE V
FALSE POSITIVES FROM DATA ERROR DETECTORS PER UPSET

Scenario	False positives per upset		
	Range check	App-BIST	Both detectors
Register upsets, FFT	0.05%	0.25%	0.26%
Register upsets, AtCorr	0.01%	0.34%	0.36%
Cache upsets, FFT	0.13%	0.35%	0.48%
Cache upsets, AtCorr	0.25%	0.21%	0.44%

VI. DISCUSSION AND FUTURE WORK

The results illustrate the extent to which pervasiveness of corruption can depend on the application; in some scenarios 95% of SDC errors impacted a single data word, while in others less than 5% did. With both applications, over 90% of unbounded data corruption errors were eliminated using application self-test.

While the results show that very low-cost methods can be useful, this study has limitations. The evaluated detectors are an option for noncritical signal processing and scientific data processing; more robust methods are needed for mission-critical computations such as flight control. The applications were run stand-alone without an operating system; separate studies would be needed regarding SDC errors originating in the memory management unit. Another limitation is the fault model, which does not account for single event transients, multi-bit upsets, arbitrary timing of upsets within a clock cycle, and upsets in software-invisible state; these more sophisticated fault mechanisms will be emulated during upcoming pulsed-laser testing at the Naval Research Laboratory.

The study suggests possibilities for further research. While we considered onboard error detection, a complementary area of study is data error detection through post-processing on the ground. The findings also have relevance for the next-generation SpaceCube 2.0 which employs PowerPC 440 embedded processors.

VII. CONCLUSION

This study provides new insights into the SDC threat in high-performance embedded processing. Applications representing two extremes are studied. While non-pervasive SDC errors are only partially mitigated (4–45%), the most problematic errors (unbounded) are reduced by 89–97% with extremely low overhead.

REFERENCES

[1] D. Petrick, "SpaceCube: current missions and ongoing platform advancements," *MAPLD Conf.*, 2009.

[2] M. Bucciero, J.P. Walters, and M.C. French, "Software fault tolerance methodology and testing for the embedded PowerPC," *IEEE Aerospace Conf.*, 2011.

[3] Materials International Space Station Experiment – 7, http://www.nasa.gov/mission_pages/station/research/experiments/MISSE-7.html.

[4] S. Mukherjee, *Architecture Design for Soft Errors*. Burlington, MA: Morgan Kauffman, 2008.

[5] K. Pattabiraman, G.P. Saggese, D. Chen, Z.T. Kalbarczyk, and R.K. Iyer, "Automated derivation of application-specific error detectors using dynamic analysis," *IEEE Trans. Dependable and Secure Computing*, pp. 640–655, 2011.

[6] M.N. Lovellette *et al.*, "Strategies for fault-tolerant, space-based computing: Lessons learned from the ARGOS testbed," *IEEE Aerospace Conf.*, 2002.

[7] P. Bernardi, L. Sterpone, M. Violante, and M. Portela-Garcia, "Hybrid fault detection technique: a case study on Virtex-II Pro's PowerPC 405," *IEEE Trans. Nuclear Science*, vol. 53, no. 6, pp. 3550–3557, December 2006.

[8] K.S. Yim, C. Pham, M. Saleheen, Z. Kalbarczyk, and R. Iyer, "Hauberk: lightweight silent data corruption error detector for GPGPU," *IEEE Int'l Parallel & Distributed Processing Symposium*, pp. 287–300, May 2011.

[9] Xilinx, Inc., "Virtex-4 PowerPC 405 - Errata for all Virtex-4 FX Devices," http://www.xilinx.com/support/answers/20658.htm.

[10] M. Bucciero, J.P. Walters, R. Moussalli, S. Gao, and M.C. French, "The PowerPC 405 Memory Sentinel and Injection System," *Int'l Symp. Field-Programmable Custom Computing Machines*, pp. 154-161, 2011.

[11] Earth Observing-1, NASA Goddard Spaceflight Center, http://eo1.gsfc.nasa.gov/.

[12] E.F. Vermote, D. Tanre, J.L. Deuze, M. Herman, and J.-J. Morcette, "Second simulation of the satellite signal in the solar spectrum, 6S: an overview," *IEEE Trans. Geoscience and Remote Sensing*, vol. 35, no. 3, pp.675–686, May 1997.