2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing

Hard Data on Soft Errors: A Large-Scale Assessment of Real-World Error Rates in GPGPU

Imran S. Haque
Department of Computer Science
Stanford University
Stanford, CA, USA
ihaque@cs.stanford.edu

Vijay S. Pande
Department of Chemistry
Stanford University
Stanford, CA, USA
pande@stanford.edu

***Abstract*—Graphics processing units (GPUs) are gaining widespread use in high-performance computing because of their performance advantages relative to CPUs. However, the reliability of GPUs is largely unproven. In particular, current GPUs lack error checking and correcting (ECC) in their memory subsystems. The impact of this design has not been previously measured at a large enough scale to quantify soft error events.**

We present MemtestG80, our software for assessing memory error rates on NVIDIA graphics cards. Furthermore, we present a large-scale assessment of GPU error rate, conducted by running MemtestG80 on over 50,000 hosts on the Folding@home distributed computing network. Our control experiments on consumer-grade and dedicated-GPGPU hardware in a controlled environment found no errors. However, our survey on Folding@home finds that, in their installed environments, two-thirds of tested GPUs exhibit a detectable, pattern-sensitive rate of memory soft errors. We show that these errors persist after controlling for overclocking and environmental proxies for temperature, but depend strongly on board architecture.

I. Introduction

Commodity programmable graphics processing units (GPUs) have made TFLOP-scale floating-point performance available on the desktop. This extremely high performance makes GPUs attractive in a number of computationally-limited applications including molecular dynamics [5], [19], molecular comparison [7], and machine learning [3].

While GPGPU (general-purpose computation on GPUs) can be attractive, its origin is in the relatively-error-tolerant area of consumer graphics; GPU reliability in non-graphics applications is largely unproven. Previous work [15], [16] has questioned GPU logic reliability. A particular concern is the reliability of GPU memory, as its lack of ECC (error-checking-and-correcting) has been previously noted [16], [17].

No prior work has tested GPGPU on a large enough scale to quantify the rate of "soft" or stochastic memory errors. Our contribution in this work is a quantification of the reliability of GPU memory subsystems carried out on over 50,000 GPUs on the Folding@home distributed computing network. MemtestG80, a standalone version of our test code, is available under the LGPL license at `https://simtk.org/home/memtest`.

Our experiment comprises over 840 terabyte-hours of memory testing distributed over more than 50,000 individual GPUs worldwide. We also present control experiments carried out on individual nodes and a GPU cluster. Our results demonstrate a detectable, pattern-sensitive rate of memory faults in the installed base of commercial GPU hardware. Even after controlling for overclocked cards and time of day (as a proxy for ambient temperature) we find that two-thirds of GPUs exhibit sensitivity to memory faults in a pattern-dependent manner. This error rate depends strongly on board architecture, with devices based on the newer GT200 GPU exhibiting failure rates an order of magnitude lower than those based on the older G80/G92 design.

We begin with a primer on soft error generation and detection mechanisms, present the design and validation of MemtestG80, and explain the methodology of our large-scale experiment. We then present the results and analysis of the experiment and conclude with a discussion of the implications of our findings.[1]

II. Background and Previous Work

Errors in hardware systems can be classified as "soft" or "hard": hard errors are triggered by physical hardware defects, whereas soft errors are random, transient faults not due to physical defects. The term "soft error" has traditionally referred to radiation-induced upsets in electronic circuits [6], [21]. Transient errors can also be caused by non-radiation mechanisms, such as leakage from adjacent circuits, timing violations, and improper signal routing or power design [4], [10], [15]. In this work, we use the term "soft error" to refer to all transient faults, rather than just radiation-induced errors.

A very-large-scale approach is required in order to efficiently accumulate statistically-significant data regarding

[1] Additional analysis and data can be found in the online Supplementary Information at `https://simtk.org/home/memtest`.

978-0-7695-4039-9/10 $26.00 © 2010 IEEE
DOI 10.1109/CCGRID.2010.84

soft error rate (SER). Estimates for RAM SER have been drawn from a variety of sources [20], ranging from 5×10^{-14} to 4×10^{-7} errors per bit-hour. Recent data from Google suggest a further hard error rate in DRAM between $2.5 - 7 \times 10^{-11}$ errors per bit-hour [14]. Data from IBM indicate that in natural environments, even with hundreds of devices under test, more than 1,000 testing hours may be required to accumulate statistically meaningful test results [6]. Additionally, possible environmental (e.g., thermal or radiation) effects on SER dictate that a variety of conditions be tested. For example, cosmic ray flux varies by a factor of two depending on latitude [21], and approximately 13x between sea level and 10,200 ft in altitude [6].

High-level software and modeling techniques have been proposed for soft error detection and correction. Software hardening techniques can be used to detect logic and memory soft errors [12]. Sheaffer et al. take a simulation-based approach and characterize the architectural vulnerability of graphics algorithms on GPUs [15]. Their later work points out that this characterization is inappropriate for GPGPU, which requires tighter error guarantees than conventional graphics [16]. Shi et al. applied a similar active testing strategy to ours to 506 GPUs at major GPU clusters and found hard errors in 1.8% of tested boards, but no soft errors [17].

While prior work has laid out testing methods to detect soft errors, no previous studies have applied these techniques to a large installed base to assess the impact of soft errors on the emerging GPGPU platform. Our contributions in this work are twofold: first, a test code using proven methods to detect soft errors; second, a large body of data using this tester to assess SER on tens of thousands of GPUs worldwide.

III. Design and Validation of MemtestG80

In this section we describe the design and validation of MemtestG80, our CUDA-based [11] code to test for memory errors on NVIDIA GPUs based on the Tesla architecture [8]. MemtestG80 is a CUDA implementation of most of the memory tests in Memtest86 [2], a widely-used open-source memory tester for x86-based machines that implements many popular memory-test patterns, including randomized tests and tests for pattern-sensitive errors.

For conciseness, we refer the reader to the supplementary information for descriptions of the test patterns, details on the implementation of a logic test unique to MemtestG80 (based on a linear congruential generator (LCG)), and additional data not presented in the main paper.

In the following, one "iteration" of MemtestG80 refers to the execution of each test once. For tests that take place in multiple rounds, every round is executed (with one exception, explained in Section IV). Such an iteration over 64 MiB of memory typically takes between 1 to 5 seconds to complete, depending on GPU speed.

A. Validation

To validate MemtestG80, we carried out both negative and positive control experiments. Since the purpose of this study is to investigate the latent error rate in GPU hardware, a true negative control (one in which we are guaranteed no errors) is not possible. To minimize errors from overheating or power disturbances, the controls were run on machines in controlled-temperature environments with known-good power sources.

We ran negative controls on a GeForce 8800 GTX and on eight NVIDIA Tesla C870 boards. Over 93,000 iterations of MemtestG80 were run over 700MiB of GPU memory on the 8800GTX. An aggregate 1.48 million iterations were run over 320 MiB on each Tesla board. No errors were detected on these boards, demonstrating that errors detected by MemtestG80 are unlikely to be spurious or due to errors in the code.

To ensure that MemtestG80 detects errors under conditions known to generate memory errors, we also carried out a positive control experiment. Since overclocking is known to generate memory errors, we used overclocking as our positive control. MemtestG80 was run on a GeForce 9500GT (default memory clock rate = 400 MHz) with its memory clock rate set to 400, 420, 430, 440, 450, 475, 500, and 530 MHz. Each test comprised 20 iterations of MemtestG80 (10 at 530MHz). To keep thermal effects from affecting test results, between tests the board was reset to 400MHz and allowed to cool to a constant temperature. Finally, we ran 20 more iterations at 400 MHz to verify that the hardware had not been damaged. The results of the positive control experiment are presented in Table I and Supplementary Figure 1.

Two results deserve special attention. First, both variants of the moving-inversions test (which write constant 32-bit patterns to memory) are completely insensitive to overclocking-induced errors. This inspires our choice of the all-zero pattern as the logic test pattern (see Supplementary Information for details), as it seems to be insensitive to some classes of memory errors.

Second, the modulo-20 test is far more sensitive to overclocking-induced errors than are the other tests. The modulo-20 test proceeds in 20 rounds. In round i, a 32-bit pattern is written to each memory location whose offset from the start of tested memory is equal to i modulo 20; the bitwise complement of the pattern is then written twice to every other memory location. Subsequently, the offsets equal to i modulo 20 are read back and verified. The sensitivity of this test may stem from the fact that the test's stride of 20 is unlikely to align with any architectural stride in the memory chips, so tested cells are more likely to be physically scattered and influenced by neighboring cells.

The sensitivity of the modulo-20 test persists in real-world situations and is key to our sampling of error rates. The pattern sensitivity of the memory subsystem is in itself an

Test name	Frequency	WER
Moving inversions (ones/zeros)	—	0
Moving inversions (random)	—	0
Modulo-20	430	9.05×10^{-4}
Random blocks	500	3.72×10^{-10}
Memtest86 walking 8-bit	475	1.14×10^{-9}
True walking zeros (8-bit)	475	2.56×10^{-8}
True walking ones (8-bit)	475	2.75×10^{-9}
Walking ones (32-bit)	475	3.76×10^{-9}
Walking zeros (32-bit)	500	9.64×10^{-9}

TABLE I
LOWEST CLOCK RATE (MHZ) AT WHICH MEMTESTG80 TESTS FOUND ERRORS IN 20 ITERATIONS AND WORD ERROR RATES AT 500MHZ RAM CLOCK

interesting result, as it demonstrates that the likelihood of encountering a memory error is highly dependent on the access and data patterns.

IV. EXPERIMENTAL METHODOLOGY

To carry out the large-scale assessment presented in Section V, MemtestG80 was deployed on the Folding@home (FAH) distributed computing network [1], [18]. For each execution of MemtestG80, we collected device information (card name, memory size, and shader-domain clock speed). The majority (97.5%) of our data were collected with a memory test region size of 64 MiB and a logic test period of 512. Because of the high memory bandwidths required on GPUs, memory blocks are interleaved across physical memory chips to speed total throughput. We believe that the tested memory region sizes are sufficiently large that they are likely to be spread across all or a substantial fraction of chips on the tested devices. On Folding@home, only 2 rounds of the modulo-20 test were run per test iteration. Later rounds were performed in following iterations.

We ran a variable number of test iterations, collecting individual results for every test iteration. Rather than measuring the exact bit-error-rate we consider only whether *any* errors were detected during an iteration. It is possible that a test which detected errors may have its "error flag" toggled off by a memory or logic error, creating a false negative. If a test is successful, a GPU error may toggle an error flag bit on — but this is in itself a GPU error. Thus, this approach solves false positives: in the worst case, our results may underestimate the error rate, but will never overestimate it.

We identify GPUs by Folding@home client installation ID numbers. This may multiply-count GPUs, but we believe that this overestimation is not significant. The test was run at least once on over 50,000 distinct client IDs, with an aggregate of over 18.9 billion test iterations executed. Geographically, tested boards are distributed worldwide, with very good coverage of North America, Europe, and major population centers elsewhere. Based on logic clock frequencies, similar numbers of overclocked and "stock" (at or below reference frequencies) boards were sampled (Supplementary Figure 2). Finally, we achieve good coverage of GPUs across the NVIDIA product line, including both the G80 and GT200 GPU architectures (Supplementary Table 1).

V. RESULTS

Our underlying statistical model is that each card (in combination with its environment) has its own probability of failing a test iteration $P\left(fail\right)$, and that each card is drawn independently from some underlying distribution $P\left(P\left(fail\right)\right)$. This independence assumption is justified by the fact that our installed base is widely separated geographically. In all following plots and analyses, "$P\left(fail\right)$" refers to any given card's probability of failing a single MemtestG80 iteration. We classified each test iteration as having failed or not using the method from Section IV and calculated an empirical probability of failure (per test iteration) for each card tested as the ratio of failed tests to total tests, thereby estimating $P\left(P\left(fail\right)\right)$. To add statistical validity, we applied various cutoffs for the minimum number of test iterations a card must have completed to be used in constructing this empirical card-reliability probability distribution. For graphical clarity, instead of presenting $P\left(P\left(fail\right)\right)$, we present the cumulative distribution functions (CDFs), or integrated probability distributions, $F\left(P\left(fail\right)\right)$, where $F\left(P\left(x\right)\right) = P\left(P\left(fail\right) < x\right)$.

Fig. 1 displays the CDFs derived from this data for five values of the iteration threshold. Each trace represents the distribution calculated using a different cutoff for the number of iterations required to have been completed to consider a card for inclusion.

The most apparent trend in the data is the strongly bimodal distribution. All the CDFs start with a nonzero value at $P\left(fail\right) = 0$, representing the fraction of cards at each threshold which never failed a test. All CDFs further show a second population with a mean $P\left(fail\right)$ around 2×10^{-5}, which represents nearly all the remaining cards. Finally, there is a very small population of cards with failure rates higher than 1×10^{-4}. This bimodal trend is statistically relevant, as it continues to appear in the data even at the largest cutoff. At a threshold of 300,000 iterations, approximately one-third of cards tested never exhibited a memory error. Nearly all of the remainder had failure probabilities between 0 and 10^{-4}; only about 2% had failure probabilities higher than this. We suspect that these high-error-rate boards are ones with hard errors in GPU memory or logic, and we do not consider them further.

VI. ANALYSIS

In this section we analyze the results, broken down by properties of the GPUs. Our main statistical method is the information gain criterion for data partitioning attributes; informally, the information gain in an attribute measures the

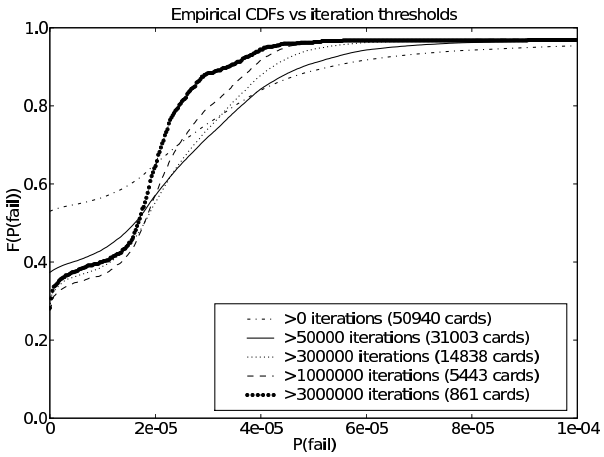

Fig. 1. Empirical cumulative distribution functions F(P(fail)) of card failure probability P(fail) at several test-iteration thresholds

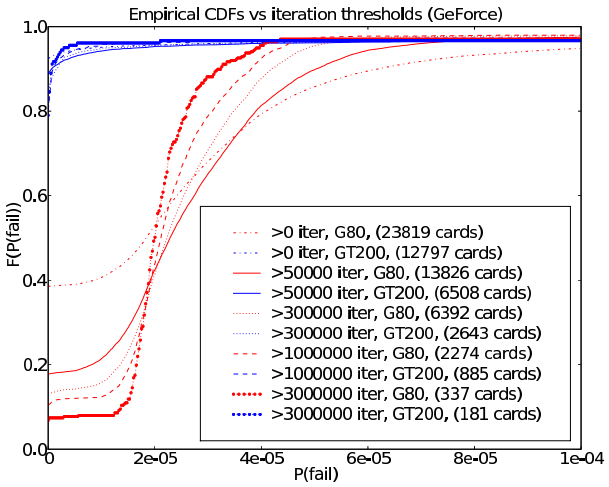

Fig. 2. Empirical CDFs of card failure probability at several test-iteration-count thresholds, by architecture (G80/G92 in red, GT200 in blue)

amount of variability in an underlying distribution explained by the attribute. Mathematical details are provided in the Supplementary Information.

A. Bimodality of P(fail)

The bimodal distribution of card failure probabilities illustrated in Fig. 1 raises an obvious question: is the existence of cards with nonzero failure probability easily explained through a simple structural or environmental variable, or is it inherent to the population of boards? To answer this we tested a set of hypotheses on our data set. As a reference, a perfect indicator variable on this dataset (separating the data into cards which never failed and those which did) has an information gain $I(D;V)$ of 0.9217 bits.

Splits of the dataset based on overclocking status and local time of day of test execution (a proxy for ambient temperature, to which we did not have access) both failed to show significant information gain on our data: 0.0565 and 0.0346 bits respectively. However, a split based on GPU architecture (G80 vs GT200) had an information gain of 0.521 bits and is significant. Fig. 2 shows that GT200-based boards were far less likely to fail MemtestG80 iterations than G80-based boards; in particular, GT200-based boards cluster around $P(fail) = 2.2 \times 10^{-6}$, an order of magnitude lower than the mode $P(fail)$ for the overall dataset.

Our data suggest that the bimodal structure of the failure probability distributions is caused by differing architectures in the boards tested. Specifically, the newer GT200 architecture has an apparent soft error rate nearly tenfold lower than that of G80. The most obvious user-visible enhancement on the GT200 memory controller relative to that on G80 is improved support for coalescing memory operations, or combining multiple memory reads or writes into single transactions. We simulated the memory access patterns for either GPU on the modulo-20 test, and found that both chips issue the same number of transactions, but G80 transmits 16.7% more bytes than GT200. By itself this does not explain a 10-fold reduction in error probability for GT200.

The large sample sizes for boards on both architectures make it unlikely that there is a consistent environmental difference between installations of either board type. While it is possible that the error rate is an age-induced effect (G80 is an older design than GT200, and it is possible that G80 boards in our sample are physically older than GT200 boards), our data suggest that GT200 is inherently more resistant to memory errors than G80.

A notable result is that, when broken down by architecture, the population of boards that never failed a test iteration decreases monotonically with respect to increasing iteration thresholds. This suggests that all the tested boards may have a nonzero error rate, and that some boards did not run the test long enough to detect one of these relatively-rare events.

B. Consumer vs professional hardware

Our dataset includes GT200-based cards from NVIDIA's Tesla line of high-end dedicated-GPGPU hardware. We here consider whether high-end (Tesla) hardware is more reliable than consumer (GeForce) hardware. Fig. 3 plots failure probability CDFs for GeForce and Tesla hardware based on GT200. While the Tesla plots are rougher due to the much smaller sample size, they appear to represent the same distribution as that underlying the GeForce traces. This suggests that the soft-error mechanism we observe is present in both GeForce and Tesla hardware at similar rates.

C. Impact of errors on molecular dynamics

To assess the impact of memory errors on scientific computing, we looked for mutual information between the probability that a given card generates memory errors and the probability that the same card triggers a simulation

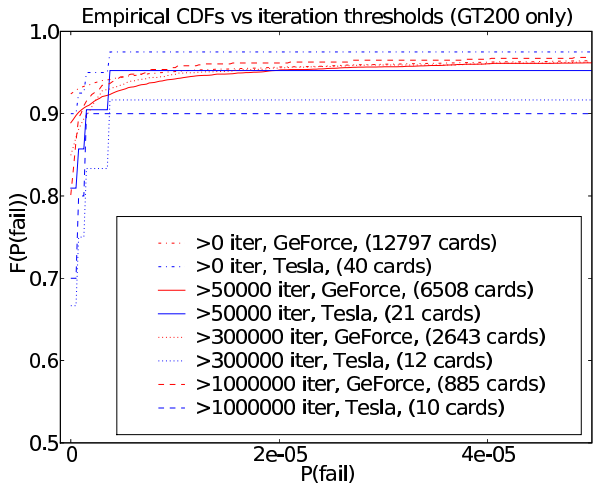

Fig. 3. Empirical CDFs of card failure probability at several test-iteration-count thresholds, by product line (GeForce in red, Tesla in blue)

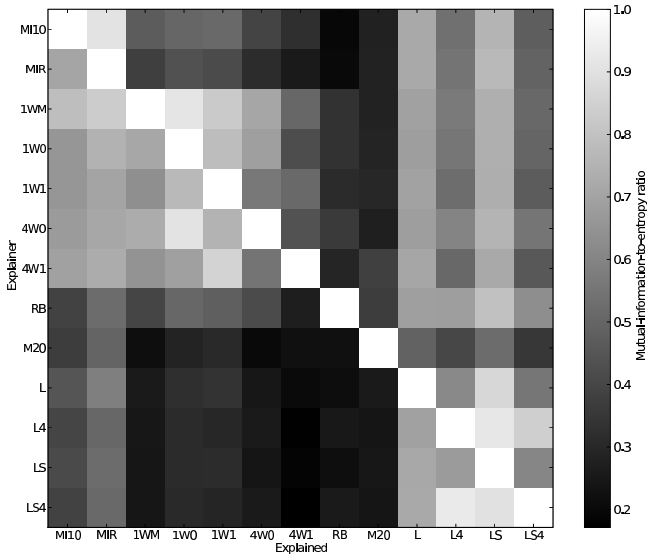

Fig. 4. Mutual information-to-entropy ratios for each test pair. Each entry is the fraction of the entropy of the test in that column explained by the test in that row. Brighter squares indicate that more of the variance of the explained test is explained by the explainer test. Test codes defined in Supplementary Information.

failure (SF) on its Folding@home work units. Counting only work units in which at least one MemtestG80 iteration was executed, the mutual information between MemtestG80 errors and Folding@home SFs was 0.131 bits, compared to overall entropies of 1.965 and 1.018 bits respectively for memory error and SF distributions. This indicates that MemtestG80 errors likely do not correlate well with SFs. However, we believe that this measure underreports the true impact. SFs have a variety of causes which may be unrelated to errors on the board; furthermore, because of the design of Folding@home, certain types of SFs were not reported to the servers and thus not logged. Hence, our results are inconclusive as to the impact of observed errors on scientific simulations.

D. Failure modes of tests

By examining the mutual information between the results of each individual test comprising a MemtestG80 iteration, it is possible to better understand the mechanisms triggering failures under various conditions. Fig. 4 shows the ratio of mutual information to total entropy for each pair of tests in MemtestG80. Informally, this describes how well the results of each test explain the results of each other test; it is a nonlinear measure of correlation. Mathematical details can be found in the Supplementary Information. Several interesting trends emerge from this data:

1) **The Modulo-20 test stands on its own**
   Both the M20 column and the M20 row in Fig. 4 have small values across their lengths, indicating the Modulo-20 test covaried strongly with no other test. This is likely due to the Modulo-20 test's high sensitivity and reinforces the notion that it probes a different failure mechanism than do other tests.
2) **The Random Blocks test is a good logic test**
   Although it was not intended as a logic test, the large values in the RB row for the columns corresponding to the LCG-based logic tests indicate that RB does a good job of capturing the errors measured by the LCG tests. Conversely, the small values in the RB column for the LCG tests demonstrate that RB is measuring a superset of errors relative to the LCG tests. This result is reasonable, as the RB test is very shader-logic intensive. We have designed it around a multithreaded, multi-core Park-Miller Minimal Standard pseudorandom number generator [13], which performs many more logic operations than any other MemtestG80 test.
3) **The logic tests measure a distinct failure mode from most memory tests**
   The four-iteration variants of the logic test (L4 and LS4) are poorly explained by most memory tests, and in particular, are less-well-explained by the memory tests than are their one-iteration counterparts (L and LS). This is to be expected, as the one-iteration variants are more influenced by memory errors. However, the bright block in the bottom-right of Fig. 4 shows that the logic tests covary strongly among themselves. Furthermore, memory tests have higher mutual information to the L4 test than the LS4 test, indicating that the use of shared memory in the latter is a significant variable. Together, these results show that the logic and memory tests detect distinct failure modes, and that errors can be observed in the on-GPU shared memory.

VII. Conclusions

We have presented the first large-scale study of error rates in GPGPU hardware, conducted over more than 50,000 GPUs on the Folding@home distributed computing network. Our control experiments on consumer-grade and dedicated-GPGPU hardware in a controlled environment found no errors. However, our large-scale experimental results show that approximately two-thirds of tested cards exhibited a pattern-sensitive susceptibility to soft errors in GPU memory or logic, confirming concerns about the reliability of the installed base of GPUs for GPGPU computation. We have further demonstrated that this nonzero error rate cannot be adequately explained by overclocking or time of day (a proxy for ambient temperature), but does correlate strongly with GPU architecture. While we cannot rule out user error, misconfiguration on the part of Folding@home donors, or environmental effects as causing nonzero error rates, our results strongly suggest that GPGPU is susceptible to soft errors under normal conditions on non-negligible timescales.

These data are particularly relevant both to GPU-based distributed computing applications and to vendors of consumer-targeted software that relies on GPU acceleration. We emphasize that although our data were collected only on NVIDIA GPUs, we have no reason to believe that the reliability picture significantly differs for GPUs from ATI, Intel, or other manufacturers, as the driving forces behind GPU development to date have not emphasized GPGPU-style reliability concerns.

What can be done?: A first step for software developers is to incorporate memory test functionality, like that found in MemtestG80, to proactively detect malfunctioning cards. On the hardware side, the addition of parity or ECC functionality to the memory subsystem would prevent memory-induced silent errors. The partial ECC in the GDDR5 specification is a first step, but cannot protect against errors in the memory controller or RAM itself.

NVIDIA's announcement that their next-generation GPU (Fermi) will feature ECC on its memory is encouraging. However, because of the economic pressures on the consumer GPU market, it is possible that ECC will not be featured at the consumer level, where the primary application remains graphics. Consequently, consumer GPGPU developers may need to implement redundant computation or software-based ECC [9].

Nevertheless, our data demonstrate that it is certainly possible to perform reliable GPGPU computing on consumer-grade hardware, but that doing so requires close attention to the characteristics of the hardware. With the great power of single-board TFLOP computation comes a great responsibility to ensure data integrity.

Acknowledgments

We foremost thank the Folding@home donors, without whom this study would not be possible. We further thank Dr. Paul Coteus and Ewen Cheslack-Postava for useful discussions, and Adam Beberg, Ewen Cheslack-Postava, Philip Guo, Peter Kasson, Alex Rasmussen, and Dustin Maghamfar for comments on the manuscript. We acknowledge support from an NSF graduate fellowship and from NIH (R01-GM062868, U54 GM072970) and NSF (CHE-0535616).

References

[1] A. L. Beberg et al. Folding@home: Lessons from eight years of volunteer distributed computing. In *8th IEEE Intl. Wkshp. High Perf. Comp. Biol. (HiCOMB 2009)*, 2009.

[2] C. Brady. Memtest86. *http://www.memtest86.com*.

[3] B. Catanzaro, N. Sundaram, and K. Keutzer. Fast support vector machine training and classification on graphics processors. In *Proc. 25th Intl. Conference on Machine Learning (ICML 2008)*, pages 104–111, 2008.

[4] B. F. Cockburn. Tutorial on semiconductor memory testing. *J. Electronic Testing*, 5(4):321–336, Nov 1994.

[5] M. S. Friedrichs et al. Accelerating molecular dynamic simulation on graphics processing units. *J. Computational Chemistry*, 30(6):864–72, 2009.

[6] M. S. Gordon et al. Single-event-upset and alpha-particle emission rate measurement techniques. *IBM J. Research and Development*, 52(3), May 2008.

[7] I. S. Haque and V. S. Pande. PAPER - accelerating parallel evaluations of ROCS. *J. Computational Chemistry*, 31(1):117–132, 2009.

[8] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. NVIDIA Tesla: A unified graphics and computing architecture. *IEEE Micro*, 28(2):39–55, 2008.

[9] N. Maruyama, A. Nukada, and S. Matsuoka. Software-based ECC for GPUs. In *2009 Symposium on Application Accelerators in High Performance Computing (SAAHPC'09)*, 2009.

[10] C. Metra, M. Favalli, and B. Riccò. Self-checking detection and diagnosis of transient, delay, and crosstalk faults affecting bus lines. *IEEE Trans. Comput.*, 49(6):560–574, 2000.

[11] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, 2008.

[12] B. Nicolescu, R. Velazco, and M. S. Reorda. Effectiveness and limitations of various software techniques for "soft error" detection: a comparative study. In *Proc. 7th Intl. On-Line Testing Wkshp., 2001.*, pages 172–177, 2001.

[13] S. K. Park and K. W. Miller. Random number generators: good ones are hard to find. *Commun. ACM*, 31(10):1192–1201, October 1988.

[14] B. Schroeder, E. Pinheiro, and W. D. Weber. DRAM errors in the wild: a large-scale field study. In *SIGMETRICS '09: Proc. 11th Intl. Conf. Meas. Model. Comp. Sys.*, pages 193–204, New York, NY, USA, 2009. ACM.

[15] J. W. Sheaffer, D. P. Luebke, and K. Skadron. The visual vulnerability spectrum: characterizing architectural vulnerability for graphics hardware. In *GH '06: Proc. 21st ACM SIGGRAPH/Eurographics Symposium on Graphics Hardware*, pages 9–16, 2006.

[16] J. W. Sheaffer, D. P. Luebke, and K. Skadron. A hardware redundancy and recovery mechanism for reliable scientific computation on graphics processors. In *GH '07: Proc. 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, pages 55–64, 2007.

[17] G. Shi, J. Enos, M. Showerman, and V. Kindratenko. On testing GPU memory for hard and soft errors. In *2009 Symposium on Application Accelerators in High Performance Computing (SAAHPC'09)*, 2009.

[18] M. Shirts and V. S. Pande. Screen savers of the world unite! *Science*, 290(5498):1903–1904, Dec 2000.

[19] J. E. Stone et al. Accelerating molecular modeling applications with graphics processors. *J. Computational Chemistry*, 28(16):2618–2640, September 2007.

[20] Tezzaron Semiconductor. Soft errors in electronic memory - a white paper. Technical report, January 2004.

[21] J. F. Ziegler. Terrestrial cosmic rays. *IBM J. Research and Development*, 40(1), January 1996.