Silent Data Corruption (SDC) Vulnerability of GPU on Various GPGPU Workloads

Manoj Vishwanathan[1], Ronak Shah[2], Kyung Ki Kim[3] and Minsu Choi[1]
[1]Dept of ECE, Missouri University of Science & Technology, Rolla, MO, USA, {mvn8c,choim}@mst.edu
[2]Analog Devices, San Jose, CA, USA, rvstg6@mst.edu
[3]Dept of Electronic Eng., Daegu University, Gyeongsan, Korea, kkkim@daegu.ac.kr

***Abstract*—GPU (Graphics Processing Unit) is emerging as a key 3D/2D graphics and parallel workload accelerator in various SoC applications. As semiconductor fabrication technology continues to scale, chips (especially those with extremely high transistor counts such as processors) are becoming increasingly vulnerable to faults that could produce unwanted errors in computing. The most severe problem is Silent Data Corruption (SDC) because this fault insidiously generates erroneous outputs without being detected. This paper discusses the characterization of SDC vulnerability of GPU on various GPGPU (General Purpose computing on GPU) workloads.**

I. Introduction

Traditionally, GPUs were not protected against soft error since their main task was to accelerate 3D graphics applications where errors manifest as visual artifacts and go almost unnoticed at 60Hz screen refresh rate. However, as they move into general-purpose computing market, this is no longer accepted. Modern GPUs, such as AMD GCN [1] and NVIDIA Kepler [2], are protected by on-chip and off-chip memory error protection (e.g., EDAC - Error Detection and Correction codes) and bus protection (e.g., CRC - Cyclic Redundancy Check). However, non-memory resources such as ALU, instruction decoding, control logic and other GPU-specific thread management hardware structures are still not protected against soft error. A soft error in a logic circuit may not be always captured in a memory circuit because it could be masked by one of three inherent masking phenomena reported in [3]–[6].

There is a significant shortcoming of EDAC (Error Detection And Correction) memories against soft errors caused by transient faults. EDAC memories can detect/correct a certain number of bit flips from the non-erroneous data word that is previously latched. That means error protection is provided if and only if a non-erroneous word and its check bits are already present. The problem is that they cannot detect/correct an erroneous incoming data word from a logic block, if a Single Event Transient (SET) fault strikes the logic block and generate a glitch. For instance, Fig. 1 in page 1 shows a NAND2 gate affected by a transient *flip-to-1* error caused by a particle strike. If this error bit is latched, it cannot be detected/corrected by EDAC. Whatever incoming data word regardless of its logical correctness is accepted **as is**. Then, it is processed to generate check bits, and both data word and check bits are latched. Now, the latched erroneous data word propagates through the rest of the system and may cause further SDC. This scenario cannot be detected/corrected by the other EDAC memories in the system. The main objective of this work is to characterize SDC vulnerability of GPU structures on various GPGPU workloads through fault injection simulation.

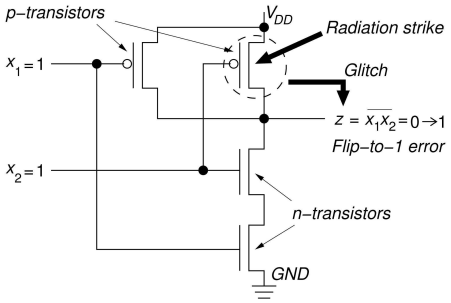

Fig. 1. NAND2 gate affected by a transient *flip-to-1* error [4]. If this erroneous bit is latched without being masked, it can't be detected/corrected by EDAC and may cause SDC.

II. The proposed fault injection framework

AMD Evergreen GPU has been selected as a baseline architecture in this work. A block diagram of the target GPU is shown in Fig. 2. For cycle-accurate ISA-level architectural simulations, Multi2Sim heterogeneous system simulation framework has been extensively leveraged to generate cycle-accurate simulation results. Multi2Sim is a simulation framework for heterogeneous CPU-GPU computing platform that integrates task-parallel CPUs and data-parallel GPUs. Currently it supports the simulation of superscalar, multi-threaded x86 CPU and AMD's Evergreen and Graphics Core Next (GCN) GPUs [7], [8]. An overview of simulation paradigm in Multi2Sim architectural simulator is shown in Fig. 3.

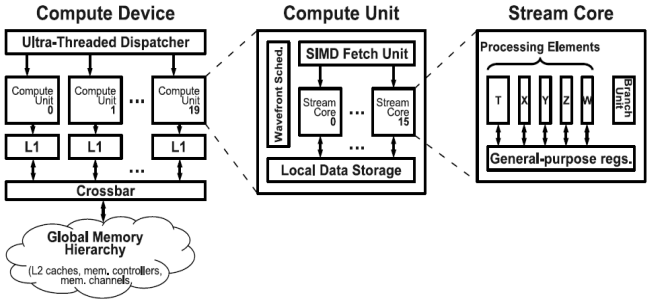

Fig. 2. Block diagram of the target GPU device [7].

A standard set of GPGPU benchmark workloads included in OpenCL SDK (Software Development Kit) [9] provided by GPU hardware vendor has been used to quantitatively evaluate SDC vulnerability. OpenCL (Open Computing Language) is an industry-standard, cross-platform, royalty-free parallel programming framework designed to program scalable heterogeneous computing platforms consisting of CPUs, GPUs and other coprocessors [10]–[12]. The benchmark programs selected for evaluation include DCT (Discrete Cosine Transform), prefix sum, fast Walsh transform,

matrix multiplication, scan large array, bitonic sort, and eigenvalue. They are selected to cover various hardware utilization and access patterns representing modern GPGPU applications.

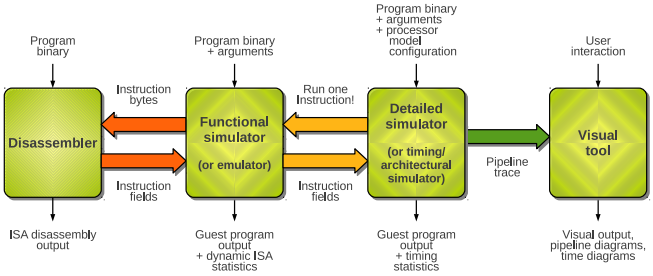

Fig. 3. Simulation paradigm in Multi2Sim architectural simulator [7].

Soft errors are spatially and locally random in nature [13]–[15]. Faults are randomly injected in any of the three GPU memory structures, register file (REG), local memory (MEM), and active mask stack (AMS), simulating unmasked error bits are originating from the logic HW structures connected to them. Then, simulate whether it causes an erroneous output, abnormal termination, or stall, where all three cases are considered as a failed execution with respect to the injected fault bit. We inject a faulty bit and compare the output of the OpenCL kernel with a reference fault-free output generated by the CPU. If the output generated by the GPU exactly matches, then, it is considered as a pass (i.e., benign fault), otherwise marked as a failure. Random fault injection is repeated until well-converged SDC data is obtained.

III. Results

Fault injection-based SDC vulnerability simulation results are summarized in Table I showing workload, # of failed execution (EXE_{fail}), # of faults actually utilized ($Fault_{hit}$), # of SDC (SDC), SDC rate (SDC_{rate}) and SDC vulnerability (SDC_{vul}), respectively. EXE_{fail} includes executions giving incorrect outputs, stalls or hangs. Out of them, only failed executions with erroneous outputs are SDC. Therefore, $SDC_{rate} = SDC/EXE_{fail}$. Also, $Fault_{hit}$ shows how many injected faults are actually utilized in target workload's execution. So, SDC vulnerability with respect to the target workload can be quantified by finding the ratio of SDC and $Fault_{hit}$ (i.e., $SDC_{vul} = SDC/Fault_{hit}$). From the last row showing the average values of SDC_{rate} and SDC_{vul}, one can conclude that: 1) each injected fault bit has 15.91% chance to cause SDC when actually utilized in execution, and 2) SDC is significantly more likely (i.e., 91.58%) than stall or hang when the correct execution is not successful due to the injected fault bit.

TABLE I
Workload-wise SDC measurement results

$Workload$	Exe_{fail}	$Fault_{hit}$	SDC	SDC_{rate}	SDC_{vul}
DCT	383	1900	350	91.38	18.42
Prefix Sum	99	456	93	93.93	20.39
Fast Walsh	30	705	30	100	4.25
Mat Mult	712	2576	691	97.05	26.82
Scan Large	15	432	10	66.67	2.31
Bitonic	13	115	13	100	11.30
Eigenvalue	50	165	46	92	27.89
Avg				91.58	15.91

As discussed in the previous section, three major GPU memory structures including AMS, REG and MEM are targeted for the proposed fault injection to simulate unmasked soft error bits originating from logic structures. In order to quantify each structure's relative SDC vulnerability, GPU structure-wise SDC simulation results are also shown in Table II. For example, 4, 346 and 0 SDC executions are resulted at AMS, REG and MEM out of the total of 350 SDC executions for DCT workload. From the results shown in this table, it can be observed that logic structures providing input words to REG are major sources of SDC and should be protected to mitigate SDC.

TABLE II
Structure-wise SDC measurement results

$Workload$	AMS	REG	MEM
DCT	4	346	0
Prefix Sum	0	73	20
Fast Walsh	0	30	0
Matrix Mult	1	680	9
Scan Large	0	9	1
Bitonic Sort	0	13	0
Eigenvalue	0	46	0

References

[1] AMD. (2012) White Paper: AMD GRAPHICS CORES NEXT (GCN) ARCHITECTURE. [Online]. Available: http://www.amd.com/us/Documents/GCN_Architecture_whitepaper.pdf

[2] NVIDIA. (2012) Whitepaper: NVIDIAs Next Generation CUDA Compute Architecture: Kepler TM GK110. [Online]. Available: http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[3] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on.* IEEE, 2002, pp. 389–398.

[4] I. Polian, J. Hayes, S. Reddy, and B. Becker, "Modeling and mitigating transient errors in logic circuits," *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, no. 4, pp. 537–547, 2011.

[5] K. Mohanram and N. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *International Test Conference*, 2003, pp. 893–901.

[6] N. Miskov-Zivanov and D. Marculescu, "A systematic approach to modeling and analysis of transient faults in logic circuits," in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design.* IEEE, 2009, pp. 408–413.

[7] Multi2Sim Developer Team. (2012) Multi2Sim: A CPU-GPU Simulator for Heterogeneous Computing. [Online]. Available: http://www.multi2sim.org

[8] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2Sim: a simulation framework for CPU-GPU computing," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*. ACM, 2012, pp. 335–344.

[9] " AMD Accelerated Parallel Processing (APP)Software Development Kit (SDK) ." [Online]. Available: http://developer.amd.com/sdks/amdappsdk/

[10] J. Kim, S. Seo, J. Lee, J. Nah, G. Jo, and J. Lee, "OpenCL as a unified programming model for heterogeneous CPU/GPU clusters," in *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*. ACM, 2012, pp. 299–300.

[11] J. Lee, J. Kim, S. Seo, S. Kim, J. Park, H. Kim, T. Dao, Y. Cho, S. Seo, S. Lee *et al.*, "An OpenCL framework for heterogeneous multicores with local memory," in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*. ACM, 2010, pp. 193–204.

[12] J. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 3, p. 66, 2010.

[13] J. Tan and X. Fu, "RISE: improving the streaming processors reliability against soft errors in gpgpus," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*. ACM, 2012, pp. 191–200.

[14] F. Wang and Y. Xie, "Soft error rate analysis for combinational logic using an accurate electrical masking model," *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, pp. 137–146, 2011.

[15] J. Tan, N. Goswami, T. Li, and X. Fu, "Analyzing soft-error vulnerability on GPGPU microarchitecture," in *Workload Characterization (IISWC), 2011 IEEE International Symposium on.* IEEE, 2011, pp. 226–235.