Identifying the Most Reliable Collaborative Workload Distribution in Heterogeneous Devices

Gabriel Piscoya Dávila, Daniel Oliveira, Philippe Navaux, Paolo Rech
Institute of Informatics, UFRGS, Porto Alegre, Brazil
Email: {gpdavila,dagoliveira,navaux,prech}@inf.ufrgs.br

***Abstract*—The constant need of higher performances and reduced power consumption has lead vendors to design heterogeneous devices that embed traditional CPU and an accelerator, like a GPU or FPGA. When the CPU and the accelerator are used collaboratively the device computational performances reach their peak. However, the higher amount of resources employed for computation has, potentially, the side effect of increasing soft error rate. In this paper we evaluate the reliability behavior of AMD Kaveri Accelerated Processing Units executing a set of heterogeneous applications. We distribute the workload between the CPU and GPU and evaluate which configuration provides the lowest error rate or allows the computation of the highest amount of data before experiencing a failure. We show that, in most cases, the most reliable workload distribution is the one that delivers the highest performances. As experimentally proven, by choosing the correct workload distribution the device reliability can increase of up to 9x.**

I. Introduction

Lately, heterogeneous architectures that integrate computing cores of different nature in a single device appeared in the market. Most of these devices include a traditional CPU and an accelerator, such as a Graphics Processing Units (GPU) or Field Programmable Gate Array (FPGA). Heterogeneous systems typically allow the computing cores to share a common memory, significantly improving performances and reducing power consumption by removing costly device-to-device data transfers. Thanks to their improved efficiency, heterogeneous devices are widely used in portable devices such as smartphones, tablets, and laptops. Additionally, autonomous vehicles, which require a considerable computing power to detect objects in real-time but dispose of a limited power budget would significantly benefit from heterogeneous devices improved performances.

Heterogeneous devices became then extremely attractive for the markets that are investing in autonomous vehicles, such as the automotive, military, and aerospace ones. Self-driven cars are expected to be soon produced in large-scale, Unmanned Aerial Vehicle (UAV) is essential to guarantee security, and space agencies, such as NASA and ESA, are working on an autonomous vehicle for deep space exploration. Moreover, the computing efficiency of heterogeneous devices and their reduced memory transfer time are also an attractive solutions in the High Performance Computing (HPC) domain to reach exaFLOPS scale (10^{16} floating point operations per second).

As heterogeneous devices move into the automotive, military, aerospace, and HPC markets, questions about their reliability, a secondary aspect for user applications, start to arise. We cannot trade-off performances for reliability in safety-critical applications. Reliability has also been listed as one of the 10 top challenges to reach exaFLOPS scale [1]. Transient errors lead to lower scientific productivity and significant monetary loss [2], [3]. While the reliability of standalone CPUs and FPGAs has been extensively studied [4], [5] and some preliminary papers have also addressed the reliability of GPUs for HPC and automotive applications [6], [7], there are few works that target heterogeneous devices reliability.

In this paper, we increase the knowledge on heterogeneous devices reliability targeting the impact of collaborative workload distribution in the device error rate. Our goal is to identify the workload distribution that brings higher reliability to the heterogeneous device. We investigate the reliability behavior of AMD Kaveri Accelerated Processing Units (APUs), that integrate CPU and GPU in a single chip. We evaluate, through both accelerated neutron beam experiments and software fault-injection, the error rate and the amount of data the device correctly produces before experiencing a failure. We consider four algorithms from different computation classes and progressively distribute the workload from the CPU to the GPU, with the two cores working collaboratively, and evaluate how the workload distribution affects the device reliability.

The main contributions of this paper are: (1) A realistic error rate analysis of heterogeneous devices, (2) the accuracy of software fault-injections in measuring the sensitivity of such devices, and finally (3) the workload distribution that provides the highest reliability regarding error rate and execution time.

The remainder of the paper is organized as follows. Section II presents background information on heterogeneous devices reliability. Section III describes the benchmarks we select and the way the workload is distributed. Section IV details our evaluation methodology. Section V presents the results of the fault injection campaign and neutron beam experiments. Finally, Section VI concludes the paper.

II. Background

Heterogeneous devices typically consist of a CPU combined with an accelerator embedded in the same chip. The most common devices used as accelerators are GPUs and FPGAs.

Previous works demonstrate that an efficient workload distribution between the available cores that consider their

978-3-9819263-2-3/DATE19/©2019 EDAA

different computing nature can significantly improve performances [8]. Heterogeneous devices performances can also be improved by implementing a collaborative programming approach where the CPU is not idle while waiting for the accelerator to complete the job. Thus, the CPU executes relevant parts of the problem and work collaboratively with the accelerator to complete the job in the fastest possible way, reducing the execution time and energy consumption of the device [9], [10].

The authors in [11] also brought to discussion the reliability of APU hardware structures calculating their Architectural Vulnerability Factor (AVF) when executing a set of heterogeneous benchmarks. These preliminary results show that the CPU core has a significantly higher AVF than the GPU, about $3\times$. However, the study does not take advantage of a collaborative approach as CPU and GPU workloads were executed sequentially [12], [13]. In this paper, we evaluate the impact of the collaborative execution of algorithms on both devices versus an execution on the CPU or GPU alone.

A. Radiation effects in heterogeneous devices

A radiation-induced transient error leads to: (1) no effect on the program output (i.e., the fault is masked, or the corrupted data is not used), (2) a Silent Data Corruption (SDC) (i.e., an incorrect program output), or (3) a Detected Unrecoverable Error (DUE) (i.e.,a program crash or device reboot). The raw radiation sensitivity strongly depends on the device physical implementation [14] while the probability for faults in low-level resources to propagate at visible states depends on both the device architecture [15] and on the executed code [16]. However, some reliability behaviors have been shown to be shared among different devices and algorithms [17].

The most common ways to evaluate a device or software reliability are accelerated neutron beam tests and software fault injections. Beam tests emulate the natural radiation effects in all the available computing resources providing the realistic error rate of a device expressed in **Failure in Time** (FIT, errors per 10^9 hours of operation) [14]. However, as errors are observed only at the output, beam tests offer limited faults visibility. With fault injections it is possible to calculate the **Program Vulnerability Factor** (PVF), which is the probability for a fault program state to affect the output correctness [16].

The workload distribution impacts the number of resources required for computation (and, then, the FIT) as well as the probability for a fault to propagate (so the PVF). However, the workload distribution also impacts performances. Neither the FIT or PVF are directly related to the code execution time. A configuration may have a higher error rate but also a shorter execution time, which means that a higher number of executions may be completed between two errors. To also consider execution time in our reliability analysis we use the **Mean Executions Between Failures** (MEBF) [18], [19]. The higher the MEBF the higher the code reliability.

III. Benchmarks and Collaborative Workload Distributions

There are two main collaboration patterns of Workload Distribution (WD): data and task partitioning [9]. In the former, the available cores execute the same algorithm, each on the assigned portion of input data. In the latter, each core executes a task that helps the device to reach the solution. In this paper, we focus on data partitioning for two main reasons: (1) the architectural characteristics of the APU, specifically the shared memory between CPU and GPUand (2) in the tasks distribution the CPU and GPU would execute significantly different codes, making it harder to understand if the observed differences in FIT rates are related to the core, the code, or the combination of them.

Data partitioning can be obtained by partitioning the input or the output data. We test both input and output data partitioning. When output data is partitioned, a merge operation is required when both cores complete the execution, which is not required for input data partitioning [9].

To evaluate the impact of collaborative WD on heterogeneous devices reliability we selected a subset of codes from the Collaborative Heterogeneous Applications for Integrated Architectures Benchmarks (CHAI) [9]. As detailed in the following, each one represents a different class of application and is likely to stimulate specific resources on the tested device. Hence, we believe that the results could be generalized to applications with similar computing characteristics.

Bezier Surface (BS) is a compute-bound algorithm widely used in engineering. The algorithm is a generalization of Bezier curves where a surface is stretched according to control points performing tensor-product operations. The collaborative approach on BS performs *output data partitioning* dividing it into tiles and assigning them to the CPU or the GPU. In this approach, both compute units work concurrently to achieve the result. A merge operation is required to connect the edges of the output tiles when the workload is distributed. Thus, the program needs to execute more operations when both compute units are used, and the FIT may increase according to the reliability of the merge operation.

Stream Compaction (SC) is a memory-bound algorithm commonly used in databases and image processing applications. SC is composed of a data manipulation primitive that removes elements from an array, keeping only those satisfying an input predicate. The *input data partitioning* collaborative approach on SC divides the input array into tiles and distribute them among the CPU and GPU. The memory-bound characteristic of this benchmark makes it more suited to the CPU with a more advanced memory hierarchy. The GPU will have to idly wait for data which may increase fault masking resulting in a lower FIT at the cost of execution time.

Canny Edge Detection (CED) is an image processing algorithm and, as such, it intrinsically performs better on a GPU. CED is a technique to extract useful structural information from images. The *input data partitioning* collaborative approach on CED divides the input frames between the CPU

and GPU. The CPU and GPU work concurrently to achieve the result and no communication is needed between them. Since all compute unit are executing intensive operations, consequently increasing the resources usage, the FIT may increase when both compute units are working. The input frames are a subset of Urban Dataset used for training classification neural networks [20].

Breadth First Search (BFS) is a search in graphs algorithms that performs non-uniform memory access widely used in GPS Navigation Systems. We use a hierarchical queue-based version of graph traversal algorithm. As graphs are irregular, the number of nodes and edges to be processed at a given time may vary. It is known that small frontiers are more efficiently processed on the CPU while large ones are better suited for the GPU [21]. Therefore, the *input data partitioning* collaborative approach is implemented by selecting a threshold, that depends on the nodes that have to be processed on the next frontier. The BFS is the only benchmark where kernel functions are invoked continuously when CPU and GPU are distributing the workload. Thus, this increase in synchronization points between CPU and GPU may lead to a higher DUE FIT rate as well when both compute units are working together. The input graph we select for our evaluation represents the highways, streets, and bridges of Great Lakes area in the US [22]

IV. Methodology

In this study, we consider the AMD A10 7890K Kaveri APU. This device includes 4 Steamroller CPU cores and an AMD Radeon R7 Series GPU containing 512 cores with 866MHZ each. The APU has two sets of 96KB shared L1 instruction caches and four sets of 16KB L1 data caches. The APU also has two sets of 2MB shared L2 caches.

Radiation experiments were performed at the ChipIR facility, Rutherford Appleton Laboratory, Didcot, UK in June 2018. ChipIR flux is suitable to mimic the terrestrial neutron effects on electronic devices. This flux means that error rates measured at ChipIR scaled down to the natural flux provide the predicted error rates on a realistic application expressed in Failure In Time (FIT). The experimental flux is about 6 to 8 orders of magnitude higher than the atmospheric neutron flux at sea level (which is $13n/(cm^2/h)$ [23]). Tests were conducted for more than 500 hours of beam time. When scaled to the natural environment, our results cover at least 5×10^8 hours of normal operations, which are 57,000 years.

In the real environment, having more than one fault at a time is rare. To maintain this behavior, the experiments were tuned to guarantee observed output error rates lower than 10^{-4} errors/execution, ensuring that the probability of more than one neutron generating a failure in a single benchmark execution remains negligible.

For the software fault-injection campaign we use CAROL-FI [24]. As demonstrated in [25], CAROL-FI can obtain useful information about the error propagation paths and the authors also performed radiation experiments alongside fault injection campaigns in the same device. CAROL-FI inject faults at the source code level, measuring the PVF. It is worth noting that, as we do not have access to the AMD proprietary architectural-level fault injector, we can inject only at source code level, accessing the PVF and not the AVF. However, the PVF still can predict the vulnerability of a code executed in a specific architecture [16].

TABLE I
Execution time, FIT, and MEBF for the tested codes and configurations

Code	WD CPU-GPU	Exec. Time [s]	SDC FIT [a.u.]	DUE FIT [a.u.]	MEBF [a.u.]
BS	100%-0%	2.5	19.3	1.35	74.5
	70%-30%	3.4	21.1	1.36	50.3
	50%-50%	2.4	20.1	1.88	74.3
	40%-60%	1.9	26.4	2.74	70.7
	10%-90%	0.9	26.5	3	149.7
	0%-100%	1.0	14.6	2.9	242.3
SC	100%-0%	0.17	83	3.77	261.1
	70%-30%	0.11	50.4	2.47	620.3
	50%-50%	0.19	38.1	2.19	507.1
	40%-60%	0.22	34.3	2.14	485
	0%-100%	0.37	37.7	1.01	263.3
CED	100%-0%	5.8	3.7	3.46	165.8
	70%-30%	4.6	4.9	3.44	159.1
	40%-60%	2.7	4.3	3.57	312.4
	10%-90%	0.8	1.3	3	3549.4
	0%-100%	0.1	1.7	3.68	14749.5
BFS	100%-0%	0.94	1.2	0.53	3160.8
	50%-50%	0.35	2.5	4	4136.2
	0%-100%	0.30	1	5.1	11920.5

TABLE II
PVF for the tested codes

Code		CPU PVF	GPU PVF
BS	SDC	3.05%	0.40%
	DUE	9.38%	28.50%
SC	SDC	0.35%	0.93%
	DUE	39.7%	48.8%
CED	SDC	2.5%	1.03%
	DUE	17.70%	14.07%
BFS	SDC	1.4%	2.3%
	DUE	13.2%	28.1%

V. Results

In this section, we report and discuss collaborative WD performances and error rates measured by radiation experiments and fault injections. Additionally, using the MEBF metric we can identify the most reliable workload distribution.

Table I depicts the radiation experiments result for the four benchmarks tested. The second column shows the WD between CPU and GPU. The third column presents the execution time in seconds. The fourth and fifth columns present the SDC FIT and DUE FIT rate. Finally, the last column shows the MEBF. The FIT rates and MEBF data have been normalized to the lowest SDC FIT rate we have measured. This normalization allows a direct comparison of the FIT rates of the different codes and different configurations without revealing business-sensitive data.

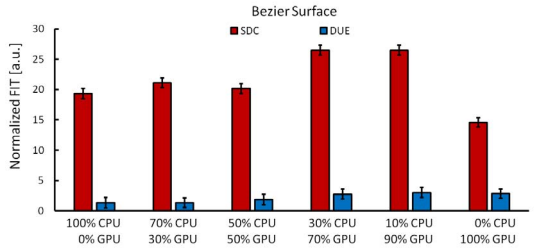

Fig. 1. FIT rates for Bezier Surface

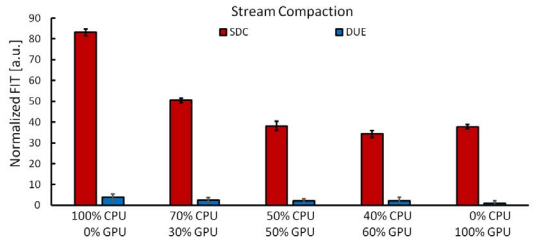

Fig. 2. FIT rates for Stream Compaction

A. Error Rate

We have performed fault-injection experiments using the CAROL-FI as described in section IV. Table II shows the SDC and DUE PVF rates for all codes with the workload assigned entirely to one of the cores (100% CPU or 100% GPU).

Figures 1, 2, 3, and 4 show the SDC and DUE FIT rates for the experimental results of each tested code with different collaborative WDs, from the whole workload assigned to the CPU (100% CPU, 0% GPU in the Figures) to the whole workload assigned to the GPU (0% CPU, 100% GPU in the Figures). Not surprisingly both the SDC and DUE rates change from code to code and from configuration to configuration. However, the DUE rate is less dependent on the executed code or configuration than the SDC rate. The average DUE FIT rate variation among the tested codes is about 35% while the average SDC rate varies of more than 100%. The DUE rate has a component that is independent of the executed code and derives from the underlying hardware characteristics such as control logic, synchronizations, and interfaces [17]. DUEs can also be generated by faults affecting the instruction cache or the code execution but, as shown in [17], DUE is mostly independent on the executed code. The DUE PVF shown in Table II could not replicate the behaviour observed during radiation experiments, confirming that one must access the hardware components to replicate the reliability behaviours.

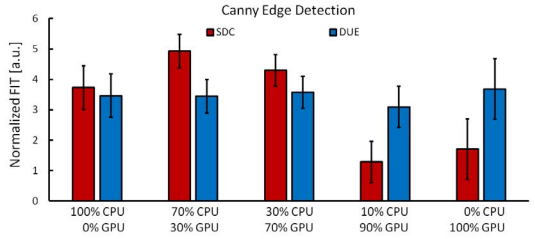

Fig. 3. FIT rates for Canny Edge Detection

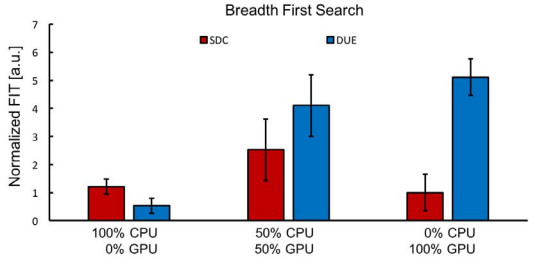

Fig. 4. FIT rates for Breadth First Search

BFS DUE FIT rate shows an increasing trend when we increase the workload computed by the GPU (refer to Fig. 4). The BFS code executes the GPU kernel only when the number of nodes to be processed surpasses a predefined threshold, resulting in several GPU kernel launches and stressing the synchronization between the CPU and GPUs as explained in section III. Thus, as the GPU workload increases, the number of kernel launches increases as well leading to more DUEs. On the other hand, the remaining codes DUE rate have no statistical difference since we have only one kernel launch independent of WD.

SDC FIT rate, in contrast to DUE FIT rate, strongly depends on the algorithm that is being executed [17], [26], [16] and on how the workload is being distributed. First, if the workload is assigned entirely to one of the cores (100% CPU or 100% GPU) the other core is in idle, meaning that data errors are not expected to impact the code output. In other words, using collaboratively the CPU and GPU is likely to increase the number of exposed resources and, thus, the SDC rate. Second, even if the executed code is the same, CPU and GPU have remarkably different AVFs [11]. This different AVF means that a fault in the GPU resources is less likely to impact the output than a fault in the CPU.

Figures 1 and 3 shows the BS and CED SDC FIT rate respectively, we can see that GPU is more reliable than CPU when there is no WD which agrees with the PVF results in Table II and the AVF results in [11]. When the workload is distributed between CPU and GPU, the SDC rate increases since the code executes concurrently and more resources are now used increasing the probability of a fault to cause SDCs. The only exception is for CED with the WD of 10% CPU and 90% GPU which presents a lower SDC FIT rate than CPU or GPU. However, as the error bars demonstrate, due to time limits for radiation experiments we could not gather enough data to have a statistical sound analysis for this configuration.

SC SDC FIT rate can be seen in Figure 2. GPU also prove to be more reliable than CPU, but sharing the workload between CPU and GPU does not increase the overall SDC FIT rate, but we can see a downward trend towards the GPU SDC rate. The SC code is extremely simple, and memory operations dominate most of the time. Thus, when we use more the GPU, we only increase the amount of time spent with memory

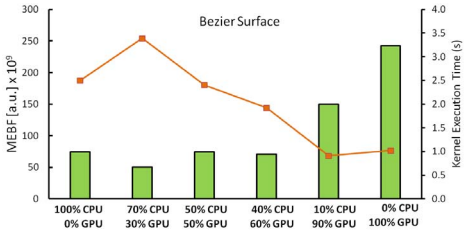

Fig. 5. MEBF for Bezier Surface

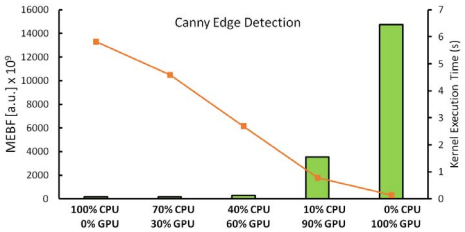

Fig. 7. MEBF for Canny Edge Detection

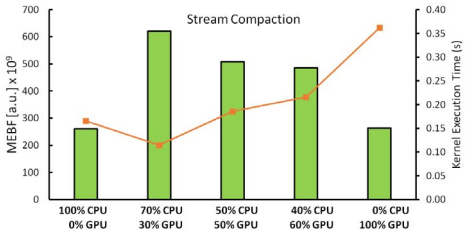

Fig. 6. MEBF for Stream Compaction

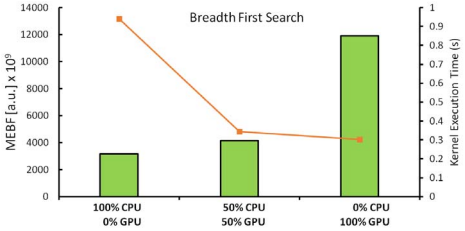

Fig. 8. MEBF for Breadth First Search

operations while decreasing the time with other operations that could cause SDCs, then, faults in unused resources during idle time will likely be masked. Additionally, the SC SDC PVF in Table II shows an opposite trend where the SDC PVF is lower for CPU. However, since SC depend heavily on memory operations which is not accessible at source code level, the PVF analysis cannot observe architectural trends that compose the most significant portion of this benchmark.

Figure 4 present the SDC FIT rate for the BFS benchmark. BFS shows a similar trend to BS and CED when the workload is shared between CPU and GPU. Thus, using more compute units concurrently we are increasing the area leading to a higher SDC FIT rate. We can also see in Figure 4 the GPU as the most reliable compute unit, but the PVF results in Table II shows the opposite trend. The BFS, similar to SC, relies extensively on architectural characteristics since BFS invokes the GPU kernel for each step of the computation, even if the workload is performed 100% in CPU. Thus, the reliability of this synchronization at each step cannot be captured by PVF analysis since the resources are not accessible at source code level.

B. Mean Execution Between Failure

FIT is the universally used metric to compare the reliability of devices and algorithms [23]. However, FIT depends only on the sensitivity of the used resources and on the probability of faults to affect the output without directly considering other important factors as execution time. To have an operative evaluation of the workload distribution that guarantees highest reliability we consider the Mean Executions Between Failures (MEBF) metric. MEBF is the number of correct executions completed between two failures and is measured dividing the MTBF (inversely proportional to FIT) with the code execution time (reported in Table I). The higher the MEBF the higher the amount of useful data produced between failures and, thus, the higher the reliability.

As detailed in the previous sections, the WD affects both the execution time and the FIT. Through the MEBF shown in Figures 5, 6, 7, and 8 we can identify the configuration that delivers higher performances and lower error rate. A proper workload distribution can significantly increase the amount of data correctly produced by the device. For BS, CED, and BFS (Figures 5, 7, and 8 respectively), choosing to use GPU-alone (100% GPU) can improve the reliability of 4.2x, 9.1x, and 2.8x, respectively. For SC shown in Figure 6, the most reliable configuration is achieved when 70% of the workload is assigned to the CPU with a 2.4x higher MEBF concerning CPU-alone (100% CPU). A promising result we can gather from our analysis is that the WD that guarantees shorter execution time is also the configuration with higher MEBF and, thus, the most reliable. This statement holds for all the tested code but BS, for which best performances are delivered with the 90% GPU configuration, but the highest MEBF is achieved using the GPU-alone. BS is faster on GPUs than CPUs (refer to Table I), but CPU can still complete 10% of the workload faster than the GPU completes 90% resulting in better execution time when compared to GPU-alone. Nevertheless, the difference in the execution time between 90% and 100% GPU is less than 10% (please refer to Table I). The significant FIT increase due to the additional number of resources required for computation when both GPU and CPU are used is not compensated by the slightly reduced execution time. For all the other codes the MEBF is higher when the execution time is lower, regardless of the FIT rate.

For SC the WD that delivers higher MEBF is not the one with the lowest FIT rate, which means that the benefit regarding reduced execution time brought by the additional resources used for computation is higher than the drawback of having a higher FIT rate. The WD impacts positively, with

gains up to 60%. The FIT rate decreases towards the GPU as explained in the previous section and it would expect the MEBF follows the same course. Due to the inefficiency of the GPU core to perform memory manipulations, the configuration that presents the best MEBF is the one that beat others in performance.

VI. Conclusions

This paper presents a reliability evaluation of WD on heterogeneous devices through radiation experiments and software fault-injection. Regarding silent errors, we show that software fault-injection can correctly identify the most reliable cores for compute-bound codes, but radiation experiments are needed for memory-bound codes where memory operations or kernel synchronization dominate execution time using hardware resources inaccessible to fault injection. We also identify the WD that achieves the highest reliability and show that in most cases it corresponds with the distribution that delivers the highest performances. While the higher amount of resources used to speed up computation increases the device faults sensitivity, the benefits regarding higher performances are higher than the drawback of having a higher error rate. Finally, DUE FIT rates depends on the collaboration pattern of WD used, showing that several kernel invocations and synchronization among cores have an adverse impact.

VII. Acknowledgment

This study was partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. Neutron beam time was provided by ChipIR (DOI: 10.5286/ISIS.E.87856107) thanks to C. Frost and C. Cazzaniga.

References

[1] R. Lucas, "Top ten exascale research challenges," in *DOE ASCAC Subcommittee Report*, 2014.

[2] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson *et al.*, "Addressing failures in exascale computing," *International Journal of High Performance Computing Applications*, p. 1094342014522573, 2014.

[3] A. S. C. Research, "Scientific Discovery through Advanced Computing - The Challenges of Exascale," https://science.energy.gov/ascr/research/scidac/exascale-challenges/, 2016, [Online; accessed 5-March-2016].

[4] G. Wang, S. Liu, and J. Sun, "A dynamic partial reconfigurable system with combined task allocation method to improve the reliability of fpga," *Microelectronics reliability*, vol. 83, no. 1, pp. 14–24, 2018.

[5] A. Vallero, A. Carelli, and S. D. Carlo, "Trading-off reliability and performance in fpga-based reconfigurable heterogeneous systems," in *2018 13th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)*, April 2018, pp. 1–6.

[6] F. G. Previlon, C. Kalra, D. R. Kaeli, and P. Rech, "Evaluating the impact of execution parameters on program vulnerability in gpu applications," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 809–814.

[7] A. Vallero, S. Tselonis, D. Gizopoulos, and S. Di Carlo, "Multi-faceted microarchitecture level reliability characterization for nvidia and amd gpus," in *VLSI Test Symposium (VTS), 2018 IEEE 36th*. IEEE, 2018, pp. 1–6.

[9] J. Gómez-Luna, I. El Hajj, V. Chang, Li-Wen Garcia-Flores, S. Garcia de Gonzalo, T. Jablin, A. J. Pena, and W.-m. Hwu, "Chai: Collaborative heterogeneous applications for integrated-architectures," in *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*. IEEE, 2017.

[8] L.-W. Chang, J. Gómez-Luna, I. El Hajj, S. Huang, D. Chen, and W.-m. Hwu, "Collaborative computing for heterogeneous integrated systems," in *International Conference on Performance Engineering (ICPE), 8th ACM/SPEC*. ACM/SPEC, 2017.

[10] T. Baruah, Y. Sun, S. Dong, D. R. Kaeli, and N. Rubin, "Airavat: Improving energy efficiency of heterogeneous applications," *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 731–736, 2018.

[11] H. Jeon, M. Wilkening, V. Sridharan, S. Gurumurthi, and G. H. Loh, "Architectural vulnerability modeling and analysis of integrated graphics processors," in *IEEE 10th Workshop on Silicon Errors in Logic - System Effects (SELSE)*, 2013.

[12] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (shoc) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. ACM, 2010, pp. 63–74.

[13] Y. Sun, X. Gong, A. K. Ziabari, L. Yu, X. Li, S. Mukherjee, C. Mccardwell, A. Villegas, and D. Kaeli, "Hetero-mark, a benchmark suite for cpu-gpu collaborative computing," in *2016 IEEE International Symposium on Workload Characterization (IISWC)*, Sept 2016, pp. 1–10.

[14] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 305–316, Sept 2005.

[15] S. S. Mukherjee *et al.*, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2003.

[16] V. Sridharan and D. R. Kaeli, "Eliminating microarchitectural dependency from architectural vulnerability," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, Feb 2009, pp. 117–128.

[17] V. Fratin, D. Oliveira, C. Lunardi, F. Santos, G. Rodrigues, and P. Rech, "Code-dependent and architecture-dependent reliability behaviors," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018, pp. 13–26.

[18] G. Reis, J. Chang, N. Vachharajani, and S. Mukherjee, "Design and evaluation of hybrid fault-detection systems," in *Proceedings of the 2005 International Symposium on Computer Architecture, ISCA'05*. IEEE Press, 2005, pp. 148–159.

[19] P. Rech, L. L. Pilla, P. O. A. Navaux, and L. Carro, "Impact of GPUs Parallelism Management on Safety-Critical and HPC Applications Reliability," in *IEEE International Conference on Dependable Systems and Networks (DSN 2014)*, Atlanta, USA, 2014.

[20] K. Fragkiadaki, W. Zhang, G. Zhang, and J. Shi, "Two-granularity tracking: Mediating trajectory and detection graphs for tracking under occlusions," in *European Conference on Computer Vision*. Springer, 2012, pp. 552–565.

[21] L. Luo, M. Wong, and W.-m. Hwu, "An effective gpu implementation of breadth-first search," in *Proceedings of the 47th design automation conference*. ACM, 2010, pp. 52–55.

[22] "9th dimacs." www.dis.uniroma1.it/challenge9/download.shtml, 2006.

[23] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices," JEDEC Standard, Tech. Rep. JESD89A, 2006.

[24] D. Oliveira, V. Frattin, P. Navaux, I. Koren, and P. Rech, "Carol-fi:an efficient fault-injection tool for vulnerability evaluation of modern hpc parallel accelerators," in *Proceedings of the Computing Frontiers Conference*, ser. CF'17. New York,NY,USA: ACM, 2017.

[25] D. Oliveira, L. Pilla, N. DeBardeleben, S. Blanchard, H. Quinn, I. Koren, P. Navaux, and P. Rech, "Experimental and analytical study of xeon phi reliability," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: ACM, 2017, pp. 28:1–28:12. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126960

[26] 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," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 36. Washington, DC, USA: IEEE Computer Society, 2003, pp. 29–. [Online]. Available: http://dl.acm.org/citation.cfm?id=956417.956570