2022 22nd European Conference on Radiation and Its Effects on Components and Systems (RADECS) | 979-8-3503-7123-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/RADECS55911.2022.10412577

Constructing Application-level GPU Error Rate Model with Neutron Irradiation Experiment

Kojiro Ito, Hiroaki Itsuji, *Member, IEEE*, Takumi Uezono, *Member, IEEE*, Tadanobu Toba, Masatoshi Itoh, Masanori Hashimoto, *Senior Member, IEEE*

***Abstract*—This work explores GPU application factors that dominantly characterize SDC and DUE cross sections obtained by neutron irradiation experiments. We evaluate correlation coefficients between the cross sections and application metrics obtained from profiler outputs and fault injection results. Experimental results with six applications show that "warps per block" is a primary factor for both SDC and DUE. The DUE cross section modeling is improved by the secondary factor of L2 hit rate.**

I. Introduction

Soft errors have attracted attention as one of the causes of malfunctions in electronic equipment. GPUs having many cores are the most advanced systems fabricated in state-of-the-art technology, and they are known to be particularly susceptible to soft errors [1]. Soft errors in HPC and safety-critical systems can lead to silent data corruption (SDC) and detectable uncorrected error (DUE). Therefore, GPU soft error resiliency assessment is required [2]. On the other hand, application-level GPU soft error tolerance evaluation is currently insufficient. One of the reasons is that a lot of internal information is kept secret in commercial GPUs. For simulation-based error tolerance evaluation of a GPU application, information such as soft error rate, error pattern, and error propagation rate of each circuit element is required. However, the information available on commercial GPUs is limited to memory size and rough program operation principles. For example, information on the scheduler for allocating parallel threads, instruction cache, and pipeline registers is unavailable. As a result, fault injection is only performed for limited resources such as registers. Fault injection using an open-source GPU design has also been done. However, it is based on old NVIDIA G80, and the simulation accuracy is problematic because it is a model that assumes operation based on public information [3]–[5].

A hardware irradiation experiment is a more direct method of evaluating soft-error tolerance than simulation. This method irradiates a device with high-density radiation using an accelerator to observe soft errors with high frequency. In the irradiation experiments, the soft error rates of each memory visible to programmers have been measured [6]. The soft error tolerance when running applications with different parallelisms and instruction types have been measured [7]. Irradiation experiments have the advantage of reproducing the errors that actually occur. However, the cost is much higher, and consequently, it is more difficult to obtain a significant number of data than simulation.

Therefore, we construct a model that can estimate various GPU applications in this study. To build the model, we conduct irradiation experiments and obtain application information. The irradiation experiments are conducted on multiple applications with different numbers of parallelism and instruction types. The application information is obtained from the profiler and fault injection results. The model variables are selected from the application information by examining the correlation between the results of the irradiation experiments and the application information. The coefficients in the model are determined by regression analysis. The model accuracy is evaluated by examining the error between the irradiation experiment and the model. Since this estimation model takes only the application information as input, it is possible to estimate the error rate without conducting irradiation experiments for each application as long as the GPU hardware is the same.

The rest of this paper is organized as follows. Section II reviews related work regarding neutron irradiation and fault injection to GPUs. Section III describes the setup and result of the neutron irradiation experiment. Section IV constructs application-level error rate estimation models and discusses their accuracy. Finally, Section V concludes the discussion.

II. Related work

A. Neutron irradiation experiments

1) Memory: D. A. G. G. de Oliveira *et al.* measured the cross sections of the L2 cache, shared memory, and registers [6]. The experimental results show that the cross section of shared memory is larger than that of L2 cache and registers. This is related to the fact that shared memory is closer to the core and then has a larger cell area to pursue speed.

Also, they explain whether the bit flips that occur can be corrected by ECC [6]. Single error correction, double error detection (SECDED) ECC is available for high-end class GPUs. SECDED ECC corrects SBUs and MCUs, but MBUs cannot be corrected. They show that the probability of MBU

This work was supported in part by the Japan Science and Technology Agency, Program on Open Innovation Platform with Enterprises, Research Institute and Academia (JST-OPERA) Program under Grant JPMJOP1721. The neutron irradiation test was performed with the Grant-in-Aid for Scientific Research (S) from Japan Society for the Promotion of Science (JSPS) under Grant JP19H05664.

Kojiro Ito was with the Department of Information Systems Engineering, Osaka University.

Hiroaki Itsuji, Takumi Uezono, and Tadanobu Toba are with Center for Sustainability - Production Engineering and MONOZUKURI, R&D Group, Hitachi, Ltd.

Masatoshi Itoh is with Cyclotron and Radioisotope Center, Tohoku University.

Masanori Hashimoto is with the Department of Communications and Computer Engineering, Kyoto University.

occurrence is less than 10% with SECDED ECC, indicating that most of the bit flips that occur in memory elements can be corrected if ECC applies to the GPU of interest.

2) Scheduler: P. Rech *et al.* investigate the error tolerance of block schedulers and thread schedulers that allocate blocks and threads by changing the number of blocks and threads in a program [8]. They report that the soft error rate increases as the number of warps increases, possibly because of the increase in usage of the warp scheduler, registers, internal registers, logic gates, and so on. On the other hand, increasing the number of blocks also increases the soft error rate, but the increase is almost negligible in the case of large block sizes. Since the number of SMs is limited in the existing architecture, the large block size makes the internal resources reused, and the resource usage does not increase much.

3) GPU application: C. Lunardi *et al.* evaluate soft error tolerance for multiple applications and multiple GPU types [7]. The error rates of SDC and DUE vary depending on applications and GPU types, respectively. Therefore, it is helpful to measure the error propagation rate of the application by fault injection to estimate the SDC rate. On the other hand, the DUE rate is highly dependent on the type of GPU, which could be difficult to estimate by fault injection.

The impact of ECC on SDC and DUE is also reported [7], [9]. ECC is very beneficial in SDC. On the other hand, for DUE, an increased error rate is observed when ECC is applied. DUE detection due to uncorrectable errors is more common than the DUE decrease, thanks to error correction.

Our previous work [10] reports that SDCs originating from visible memory elements are comparable to SDCs originating from other invisible components. [6] suggests the impact of instruction cache error on DUE since larger programs tend to have the larger DUE cross section.

B. Fault injection

Fault injection is performed using SASSIFI [11] and NVBit [9], [12], [13]. SASSIFI and NVBit inject bit flips in general-purpose registers to measure architectural vulnerability factors (AVFs) [14], where AVFs represent the probability that a bit flip occurring in a register will result in an SDC or DUE. Since there are many registers in GPUs, registers are considered to have a significant relationship with SDC and DUE rates. Therefore, AVF has been evaluated in many papers. However, error injection tools are incapable of fault injection except for some resources such as memory elements and instructions.

FlexGrip and GPGPU-sim exist for open-source GPUs [3], [4]. The RTL is available and hence fault injection to the scheduler and pipeline registers is performed [15], [16]. On the other hand, the evaluation accuracy is uncertain because the model is based on a very old GPU, the NVIDIA G80, and other public information [5].

III. Neutron Irradiation Experiment

Fig. 1 shows the experiment environment. Six NVIDIA P2000 GPU cards are fixed at a distance of about 1 to 1.5 m from the host PCs using PCI-Express extension cables, and the GPUs are irradiated with neutron beams. The power supply unit to drive the host PC is outside the irradiation room with a 5-m extension cable because of its low neutron tolerance. Similarly, the storage unit for the host Linux OS is also located outside of the irradiation room with a USB extension cable.

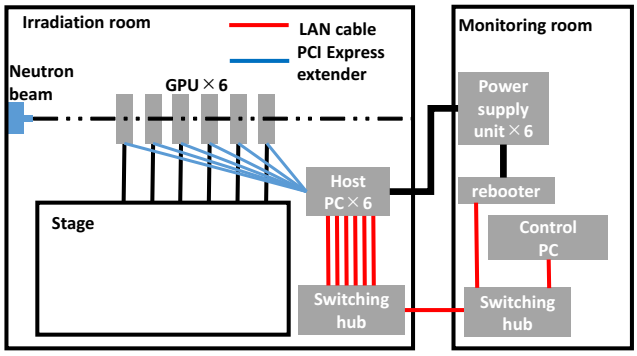

Fig. 1. Irradiation setup.

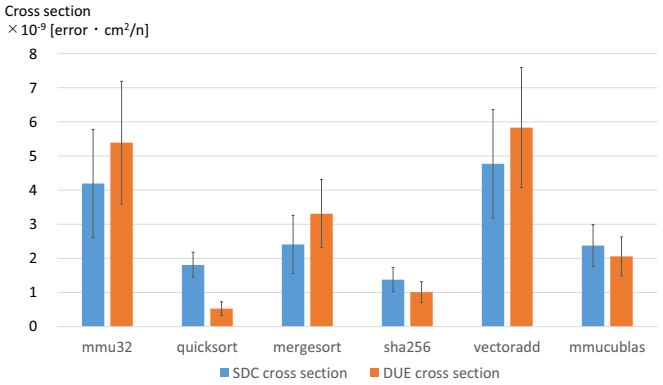

Fig. 2. Measured cross sections of GPU applications. Error bars represent one standard deviation.

We performed a quasi-monoenergetic neutron irradiation experiment at the Cyclotron and Radioisotope Center (CYRIC) at Tohoku University [17] similarly to [10]. A 70-MeV proton source produces the neutron beam, and the neutron beam has a flux peak at the energy near 70 MeV.

We used the six programs below for the experiment.

- mmu32: matrix multiplication with shared memory. One block handles a 32 × 32 matrix.
- quicksort: recursive quick sort. Each block uses one thread, and then parallelism efficiency is very low.
- mergesort: merge sort. Parallelism efficiency is higher than quicksort.
- sha256: hash function consisting of integer computations.
- vectoradd: parallel addition. Each thread performs one addition only.
- mmucublas: matrix multiplication with CUDA library. GPU utilization efficiency is very high.

mmu32, quicksort, mergesort, vectoradd, and mmucublas are sample codes provided by NVIDIA. sha256 is taken from [18].

Fig. 2 shows the SDC and DUE rates for each application obtained from the irradiation experiments. The SDC and DUE cross sections of mmu32 and vectoradd are high, while quicksort and sha256 have small cross sections. Thus, SDC

and DUE cross sections vary concerning applications.

IV. Model construction and evaluation

A. Model construction strategy

This work considers the following three models aiming to find a model with the minimum-maximum error. In the model construction, we use leave-one-out cross-validation for regression analysis since the number of samples, i.e., the number of applications, is limited in this work.

Model 1:

We identify a primary factor characterizing the application error rate from the available application information. Then, Model 1 is expressed as

$$y = a_1 \times PrimaryFactor + b,$$

where a_1 and b are model parameters. This work regards the factor with the highest correlation to the measured error rate as the primary factor.

Model 2:

Model 2 takes two factors of application information, the first factor being the same as Model 1.

$$y = a_1 \times PrimaryFactor + a_2 \times SecondFactor + b.$$

The second factor is selected such that the correlation between the estimate of Model 2 and the measured error rate improves significantly. We may test multiple second factors as different models of Model 2.

Model 3:

Model 3 consists of three factors.

$$y = a_1 \times PrimaryFactor + a_2 \times SecondFactor_1 + a_3 \times SecondFactor_2 + b,$$

where $SecondFactor_1$ and $SecondFactor_2$ are the factors selected in Model 2 construction. We test all the combinations of the second factors.

B. Application information

This work evaluated application information listed in Table VII. The prefix "AVF:" shows the architectural vulnerability factor obtained by injecting bit flips into registers with NVBit. The other factors are obtained by NVIDIA profiler [19]. The prefix "INST:" shows the proportion of instruction type. The prefix "stall_" means the proportion of stall reason.

1) Memory size and efficiency: Table I lists the number of registers and the size of shared memory used in the GPU applications. The size of the used L2 cache is the same for all the programs in this study because the input data of each program coming from the DRAM is larger than the L2 cache size. "register per thread" represents the number of blocks per thread, "register per block" represents the number of registers in a block, and "dispatched registers" represents the number of registers allocated to the GPU simultaneously. Since the number of registers used is determined per thread, a program with a large number of threads has a large number of registers. The shared memory is used only in mmu32, mergesort, and mmucublas.

TABLE I
USED MEMORY SIZE.

program	register per thread	register per block	dispatched registers	shared memory [KB]
mmu32	30	30,720	491,520	128
quicksort	30	30	3,840	0
mergesort	16	4,096	262,144	128
sha256	56	224	224	0
vectoradd	8	8,192	131,072	0
mmucublas	120	15,360	460,800	384

TABLE II
MEMORY EFFICIENCY.

program	gld_ efficiency	gst_ efficiency	shared_ efficiency	L2 hit
mmu32	100.0%	100.0%	70.0%	88.0%
quicksort	12.5%	5.0%	0.0%	99.9%
mergesort	80.5%	82.2%	27.1%	32.2%
sha256	3.1%	3.1%	0.0%	99.3%
vectoradd	100.0%	100.0%	0.0%	100.0%
mmucublas	100.0%	100.0%	47.9%	73.5%

Table II shows memory efficiency. "gld_efficiency" and "gst_efficiency" represent the throughput efficiency of load and store between the DRAM and L2 cache, and "shared_efficiency" represents the throughput efficiency of load and store for the shared memory. When the memory throughput is low, the memory remains in the standby state without being used, which increases the possibility of bit upsets during the standby state and may elevate the error rate. "L2 hit" indicates the hit ratio of the L2 cache. A higher hit rate means that the value in the L2 cache stays for a longer time, and it may degrade the error immunity.

2) Scheduler related metrics: Table III shows the number of blocks, the number of warps, and their execution efficiency. "blocks" denotes the number of blocks in the application. Blocks are allocated to SMs. When the number of blocks in the application is larger than the number of available blocks, the unallocated blocks are kept in the wait state. Then, when the other blocks finish their operations, a block in the waiting state is newly allocated.

"warps per block" is the number of warps in a block, and "dispatched warps" is the total number of warps in the allocated blocks, excluding the waiting blocks. There is a warp scheduler in SM, and threads in warps are allocated to cores by the warp scheduler. Therefore, the number of warps in a block affects the operation of the warp scheduler.

"sm_efficiency" is the percentage of time when at least one warp is running in an SM. If the number of blocks in an application is less than the number of available SMs, sm_efficiency decreases due to idol SMs. Also, programs that need to wait for synchronization with other blocks or warps are considered to degrade sm_efficiency.

"warp_execution_efficiency" represents the average number of threads in a warp. A warp consists of a maximum of 32 threads. When the number of threads is small or is not divisible by 32, the warp is composed of fewer than 32 threads, resulting in lower warp_execution_efficiency. "eli-

TABLE III
SCHEDULER RELATED METRICS.

program	blocks	warps per block	dispatched warps	sm efficiency	warp_execution efficiency	eligible_warps per_cycle
mmu32	200	32	512	99.1%	100.0%	1.89
quicksort	100,000	1	128	51.3%	7.6%	0.08
mergesort	64	8	512	99.3%	83.5%	3.18
sha256	1	1	4	12.6%	3.1%	0.06
vectoradd	49	32	512	46.8%	100.0%	2.27
mmucublas	30	4	120	90.6%	100.0%	9.38

TABLE IV
PROPORTIONS OF INSTRUCTIONS.

program	INT/ FLOAT	LDG/ STG	LDS/ STS	PREDI CATE	CONT ROL	MOVE
mmu32	46.4%	2.2%	43.5%	1.1%	3.3%	0.4%
quicksort	43.1%	11.6%	0.0%	14.2%	8.6%	19.7%
mergesort	27.7%	0.5%	13.1%	15.9%	10.8%	11.4%
sha256	72.7%	10.2%	0.0%	4.7%	7.2%	4.8%
vectoradd	52.2%	13.0%	0.0%	4.4%	8.7%	4.4%
mmucublas	88.3%	2.2%	7.6%	1.3%	0.5%	0.1%

gible_warp_per_cycle" is the number of warps allocated in each active cycle. Programs with high eligible_warp_per_cycle have high execution efficiency because warps are allocated frequently.

3) Instruction proportion: Table IV shows the percentage of instructions of each application. INT/FLOAT are integer and floating-point operations, LDG/STG are load/store from/to DRAM, and LDS/STS are load/store instructions from/to shared memory. PREDICATE is instructions that calculate and assign the value of the predicate register. CONTROL is instructions for branching, jumping, and function calls, and MOVE is an instruction used for initializing and assigning variables.

In sha256, which executes complex formulas, and mmucublas, which performs optimized matrix multiplication provided in libraries, the proportion of INT/FLOAT instructions is large. Meanwhile, in sorting programs, PREDICATE for comparison, CONTROL for branching, and MOVE for handling a large number of variables are frequently executed.

4) Stall: Table V shows the stall rates of the applications. "inst_fetch" occurs when the next assembly instruction has not been fetched, and "exec_dependency" occurs when the input required for the execution of the instruction is not ready. "memory_dependency" is a stall originating from memory operations, and "sync" is a stall where the warp is waiting for synchronization. "constant_memory_dependency" is a stall related to the constant memory operations providing constants. "pipe_busy" is a stall caused by a busy pipeline. "not_selected" is a stall caused by warp deactivation because another warp is to be executed.

5) AVF: Table VI shows the AVFs when the register values are flipped by 1 bit using NVBit, an architecture-level fault injection tool. The number of fault injections is more than 10,000 for all applications. The AVF results are categorized into masks that do not affect the output values, SDCs with different output results, and DUEs such as crashes and hangs. SDC is considered to occur when a register value is being computed, or when a wrong address is accessed when loading or storing a value. On the other hand, registers are also used to calculate load/store addresses and jump destination addresses, which may cause access to prohibited memory or prohibited addresses, and thus DUE is considered to occur.

C. Constructed models

1) Model 1: Table VII lists the correlation coefficients between the cross sections of SDC and DUE measured in the irradiation experiments and the GPU application information, in the order of the absolute value of the correlation coefficient with the cross section of SDC. A higher correlation coefficient means a more significant contribution to the soft error rate. Table VII shows a solid correlation between the cross sections of SDC and DUE, meaning that SDC-vulnerable applications are often DUE-vulnerable. The correlation coefficient of warps per block is high for both SDC and DUE. Therefore, we choose warps per block as the primary factor. Apart from that, the correlation coefficients of dispatched warp and warp_execution_efficiency are high, suggesting that the number of warps is related to the soft error rate. Also, there is a correlation to gld_efficiency and gst_efficiency. This indicates that load and store between DRAM and L2 cache likely affect SDC and DUE. On the other hand, the correlation coefficient of AVF is minimal, indicating that the AVF results have little impact on the soft error rate.

Table VIII shows the SDC cross sections estimated by Model 1 and their errors. The largest error is 39.7 % for sha256. This indicates that the SDC cross section could be mostly estimated with only one factor of warps per block. The correlation coefficient of the estimates is 0.930.

Table VIII also shows the estimated DUE cross sections and errors. The largest error is 211.0 % for quicksort. Considering that the next largest error is 42.9 % for sha256, only the estimation error for quicksort is very large. In other words, soft error rates other than quicksort are correlated with warps per block, but the correlation with warps per block is low for quicksort. This result indicates that there may be other application information besides warps per block that significantly impacts the soft error rate in the case of quicksort.

2) Model 2: For all the available application information factors, we picked up one factor as the second factor, and evaluated the correlation of the model estimate to the measured SDC cross section. However, all the remaining factors did not

TABLE V
STALL METRICS.

program	inst fetch	exec dependency	memory dependency	sync	constant_memory dependency	pipe busy	not selected
mmu32	3.4%	37.1%	11.2%	32.4%	0.5%	1.5%	1.2%
quicksort	15.3%	21.4%	52.9%	3.1%	1.4%	0.3%	0.0%
mergesort	4.6%	22.1%	21.4%	41.9%	0.0%	0.6%	1.3%
sha256	14.5%	12.4%	68.4%	0.0%	0.0%	1.3%	0.0%
vectoradd	16.6%	15.7%	37.5%	0.0%	11.8%	0.4%	1.7%
mmucublas	5.6%	6.0%	4.8%	14.7%	0.6%	4.4%	54.0%

TABLE VI
AVF OBTAINED BY 1-BIT FAULT INJECTIONS TO REGISTERS.

program	mask	SDC	DUE
mmu32	52.2%	38.7%	9.1%
quicksort	39.0%	27.6%	33.4%
mergesort	15.5%	4.6%	79.9%
sha256	8.1%	78.6%	13.2%
vectoradd	11.3%	51.5%	37.1%
mmucublas	27.9%	22.2%	49.8%

TABLE VII
CORRELATION COEFFICIENTS TO SDC AND DUE.

	Corr. Coef. (SDC)	Corr. Coef. (DUE)
(SDC cross section)	1.00	0.96
warps per block	0.98	0.96
(DUE cross section)	0.96	1.00
dispatched warps	0.80	0.89
gld_efficiency	0.76	0.80
gst_efficiency	0.75	0.80
warp_execution_efficiency	0.75	0.80
stall_constant_memory_dependency	0.70	0.59
register per block	0.62	0.62
stall_memory_dependency	-0.45	-0.49
INST: MOVE	-0.44	-0.51
INST: LDS/STS	0.44	0.53
INST: PREDICATE	-0.42	-0.37
stall_exec_dependency	0.42	0.46
register per thread	-0.41	-0.45
blocks	-0.37	-0.55
sm_efficiency	0.31	0.39
shared_efficiency	0.31	0.39
INST: INT/FLOAT	-0.23	-0.31
stall_pipe_busy	-0.16	-0.17
stall_sync	0.14	0.33
L2 hit	0.14	-0.07
stall_not_selected	-0.13	-0.18
AVF: DUE	-0.13	-0.03
stall_inst_fetch	-0.09	-0.24
INST: CONTROL	-0.09	-0.02
eligible_warps_per_cycle	0.06	0.06
shared memory	-0.05	-0.02
INST: LDG/STG	0.02	-0.15
AVF: SDC	0.00	-0.04

help improve the model-estimate correlation. This suggests that warps per block mostly explains SDC error rate, and the impact of other factors is not significant.

As for the DUE error rate, we found two factors that improved the model-estimate correlation, as shown in Table IX. The correlation coefficients are 0.984 for DUE AVF and 0.969 for L2 hit, and they are larger than the correlation coefficient of 0.926 for Model 1. On the other hand, the maximum error

TABLE VIII
SDC AND DUE ESTIMATION RESULTS (MODEL 1).

	warps per block	SDC cross section		DUE cross section	
		estimate	error	estimate	error
mmu32	32	4.8E-09	14.2%	6.0E-09	12.1%
quicksort	1	1.7E-09	-7.9%	1.6E-09	211.0%
mergesort	8	2.3E-09	-3.5%	2.1E-09	-37.2%
sha256	1	1.9E-09	39.7%	1.4E-09	42.9%
vectoradd	32	4.2E-09	-11.8%	5.7E-09	-2.9%
mmucublas	4	1.8E-09	-22.7%	1.6E-09	-21.0%
corr. coeff.		0.930		0.926	

TABLE IX
DUE ESTIMATION RESULTS (MODEL 2).

Second factor	DUE AVF		L2 hit	
	estimate	error	estimate	error
mmu32	5.1E-09	-6.2%	6.1E-09	12.8%
quicksort	1.1E-09	113.7%	7.6E-10	44.1%
mergesort	3.1E-09	-5.0%	3.8E-09	16.0%
sha256	7.3E-10	-27.4%	1.2E-09	21.5%
vectoradd	6.2E-09	6.3%	5.1E-09	-12.9%
mmucublas	1.7E-09	-15.6%	1.6E-09	-20.0%
corr. coeff.	0.984		0.969	

is 113.7% when DUE AVF is added and 44.1% when L2 hit is added, where both of them are better than the maximum error of Model 1, 211.0%. Comparing DUE AVF and L2 hit, the correlation coefficient is larger for DUE AVF, but the maximum error is smaller for L2 hit. DUE AVF is an application factor that affects the DUE cross section, but it is not sufficient to explain the DUE error of quicksort. Instead, L2 hit has a higher contribution to quicksort. Comparing the maximum errors, 113.7% with DUE AVF added and 44.1% with L2 hit added, L2 hit could be a more appropriate second factor because the maximum error of DUE AVF is substantial.

3) Model 3: We applied Model 3 to DUE estimation, where, according to the results above, warps per block, DUE AVF, and L2 are used for Model 3 construction. Table X lists the values estimated by Model 3 and their errors. The total correlation coefficient is 0.971. The maximum error is 95.0 % for quicksort. Compared to Model 2 with DUE AVF, the maximum error is smaller in Model 3. The correlation coefficient is larger in Model 3 compared to Model 2 with L2 hit. This is because the L2 hit reduces the maximum error of quicksort, and DUE AVF improves the accuracy of most applications. On the other hand, the maximum error of Model 3 is 95.0%, while that of Model 2 with L2 hit is 44.1%. Therefore, Model 2 with L2 hit could be more suitable for

TABLE X
DUE ESTIMATION RESULTS (MODEL 3).

	estimate	error
mmu32	5.8E-09	7.1%
quicksort	1.0E-09	95.0%
mergesort	4.1E-09	25.2%
sha256	8.5E-10	-15.9%
vectoradd	5.3E-09	-8.6%
mmucublas	1.7E-09	-17.1%
corr. coeff.	0.971	

estimating the DUE cross section than Model 3.

D. Discussion

This work showed that warps per block strongly affected the soft error rate. This observation is consistent with [8], which investigated the effect of the scheduler. However, as the number of warps changes, the number of registers, memory efficiency, and stall rate also change accordingly. Therefore, even if the correlation with warps per block is high, we could not conclude that the root cause is the scheduler since warps per block could be a factor that associates with and represents other factors. On the other hand, from VII, the correlation coefficients to the percentage of branch and jump instructions are small for both SDC and DUE. This indicates that the soft error rate of GPU applications may be little affected by instructions, which is different from the expectation of [6].

V. CONCLUSION

This work investigated GPU application factors that characterize SDC and DUE cross sections obtained by neutron irradiation experiments with six applications. We obtained a number of metrics that characterize the applications using profiler outputs and results of fault injection to registers. The model construction results show that "warps per block" is a primary factor in modeling both SDC and DUE cross sections, and L2 hit improves DUE estimation.

REFERENCES

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

[2] A. Lotfi, S. Hukerikar, K. Balasubramanian, P. Racunas, N. Saxena, R. Bramley, and Y. Huang, "Resiliency of automotive object detection networks on gpu architectures," in *2019 IEEE International Test Conference (ITC)*. IEEE, 2019, pp. 1–9.

[3] K. Andryc, M. Merchant, and R. Tessier, "Flexgrip: A soft gpgpu for fpgas," in *2013 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2013, pp. 230–237.

[4] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 2009, pp. 163–174.

[5] D. Kirk *et al.*, "Nvidia cuda software and gpu parallel computing architecture," in *ISMM*, vol. 7, 2007, pp. 103–104.

[6] D. A. G. G. de Oliveira, L. L. Pilla, T. Santini, and P. Rech, "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 791–804, 2015.

[7] C. Lunardi, F. Previlon, D. Kaeli, and P. Rech, "On the efficacy of ECC and the benefits of FinFET transistor layout for GPU reliability," *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1843–1850, Aug 2018.

[8] 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 *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 455–466.

[9] F. F. dos Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2018.

[10] K. Ito, W. Liao, M. Hashimoto *et al.*, "Characterizing neutron-induced sdc rate of matrix multiplication in tesla p4 gpu," Proceedings of European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2019.

[11] S. K. S. Hari, T. Tsai, M. Stephenson, S. W. Keckler, and J. Emer, "Sassifi: An architecture-level fault injection tool for gpu application resilience evaluation," in *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2017, pp. 249–258.

[12] O. Villa, M. Stephenson, D. Nellans, and S. W. Keckler, "Nvbit: A dynamic binary instrumentation framework for nvidia gpus," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 372–383.

[13] Y. Ibrahim, H. Wang, M. Bai, Z. Liu, J. Wang, Z. Yang, and Z. Chen, "Soft error resilience of deep residual networks for object recognition," *IEEE Access*, vol. 8, pp. 19 490–19 503, 2020.

[14] 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. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.* IEEE, 2003, pp. 29–40.

[15] B. Du, J. E. R. Condia, and M. S. Reorda, "An extended model to support detailed gpgpu reliability analysis," in *2019 14th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*. IEEE, 2019, pp. 1–6.

[16] J. E. R. Condia and M. S. Reorda, "Testing permanent faults in pipeline registers of gpgpus: A multi-kernel approach," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2019, pp. 97–102.

[17] Y. Sakemi, M. Itoh, T. Wakui *et al.*, "High intensity fast neutron beam facility at cyric," 2014.

[18] "Cudasha256," 2021. [Online]. Available: https://github.com/Horkyze/CudaSHA256

[19] "Profiler user's guide," 2021. [Online]. Available: https://docs.nvidia.com/cuda/profiler-users-guide/index.html