2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)

Reliability Evaluation of Mixed-Precision Architectures

Fernando Fernandes dos Santos, Caio Lunardi, Daniel Oliveira, Fabiano Libano, Paolo Rech
Institute of Informatics - Universidade Federal do Rio Grande do Sul (UFRGS)
{ffsantos, cblunardi, dagoliveira, fplibano, prech}@inf.ufrgs.br

ABSTRACT

Novel computing architectures offer the possibility to execute float point operations with different precisions. The execution of reduced precision operationsis likely to reduce both the execution time and power consumption. However, the application's error rate and the device's reliability can also be impacted by these precision changes.

In this paper, we study the impact of data and operation precision changes on the reliability of modern architectures. We consider Xilinx Field-Programmable Gate-Arrays (FPGA), Intel Xeon Phis, and NVIDIA Graphics Processing Units (GPUs) executing a set of codes implemented in double, single, and half-precision IEEE754-compliant float point data. On FPGAs, the reduced area and performance improvements brought by reduced precision operations increase reliability. On Xeon Phis the compiler biases significantly double- and single-precision instructions execution. This raises the drawback of increasing single-precision error rates when compared to double-precision. NVIDIA GPUs make use of dedicated mixed-precision cores, which draw nontrivial effects on the device reliability. In general, on GPUs half-precision allows a higher number of executions to be correctly completed before experimenting a failure.

Finally, we also evaluate how transient faults impact the output correctness. Our study shows that for most applications faults in a single or half-precision data or operation are more likely to significantly modify the output value than errors in double-precision data.

1. INTRODUCTION

Recently approximate computing has been extensively applied to both High-Performance Computing (HPC) and safety-critical applications, including neural-networks-based object detection for autonomous vehicles. For some applications, an exact output is not strictly required. Device architects and software developers have been proposing solutions to execute operations and represent data with a precision tailored to the application needs to save power, area, and eventually reducing the execution time.

Major hardware vendors are making available devices that enable the execution of operations in mixed-precision. The user can select the precision at which the operations are executed, tuning the hardware utilization and the execution time with the application needs. Mixed-precision operations are extensively used in neural networks. It has been shown that, on GPUs, good object detection accuracy can be achieved through neural network representing data in half-precision float point (16 bits) or even in short integer (8 bits) [1]. On FPGAs, researchers have reduced data precision to the limit (binary representation) and were still able to achieve satisfactory accuracy [2]. Several HPC applications could also be executed in reduced-precision, significantly improving the computing efficiency [3, 4].

While it has been proved that mixed-precision operations are very beneficial in terms of computing and power efficiency, their impact on the devices and applications reliability has not been thoroughly investigated, yet. As reliability is a primary concern for both safety-critical and HPC applications, it is mandatory to understand if, and how, mixed-precision influences the reliability behaviors of devices and codes.

The reduced area required to implement the reduced-precision hardware is expected to lower the error rate. Additionally, higher performances are likely to improve the number of correct executions completed before experiencing a failure (the Mean Executions Between Failures, MEBF). Depending on the application, some transient errors could be tolerated. Errors that affect only slightly the application output, in fact, could be less critical than others. The impact of corruption on a reduced-precision data could be more severe than corruption in a full-precision data. A fault on 64 bits, in fact, could affect only the least significant positions of the mantissa resulting in a (corrupted) value that is still sufficiently close to the expected one. As precision is reduced, the probability for the fault to change the output value significantly is expected to increase.

In this paper, we aim at deeply investigating, through accelerated neutron beam experiments and software fault-injection, the impact of mixed-precision on the reliability of modern devices and applications. We consider FPGAs implementing two circuits, including a Convolution Neural Network (CNN) with different precisions. Then, we run a set of benchmarks (including YOLOv3 CNN) using double, single, and half-precision data types on NVIDIA Volta GPUs and using double and single precisions on Intel KNC Xeon Phis. We measure the error rate for all the configurations and evaluate faults propagation probabilities for GPUs and Xeon

2378-203X/19/$31.00 ©2019 IEEE
DOI 10.1109/HPCA.2019.00041

Phis. Additionally, we evaluate through the MEBF metric the trade-off between performance and reliability of reducing the operations and data precision. Finally, we inspect how operations and data precision affects the criticality of faults by measuring the differences between the corrupted and the expected values.

The main contributions of this paper are: (1) The experimental evaluation of the impact of mixed-precision architecture in applications executed in FPGAs, GPUs, and Xeon Phis. (2) The evaluation of the performance-reliability trade-off of using mixed-precision data and operations. (3) The study of the impact of data types in the output errors criticality.

The remainder of the paper is organized as follows. Section 2 presents the radiation effects on computing devices and mixed precision architectures concepts. Neutron beam experiments and fault-injection setups are detailed in Section 3. Sections 4, 5, and 6, present experimental results and analytical analyses on mixed-precision architectures reliability. Finally, Section 7 reviews the main considerations drawn from the presented results and Section 8 concludes the paper.

2. BACKGROUND AND RELATED WORK

2.1 Radiation Effects on Computing Devices

A terrestrial neutron strike may perturb a transistor's state, generating bit-flips in memory or current spikes in logic circuits that, if latched, lead to an error. Neutron-induced events are typically *soft* errors in the sense that the device is not permanently damaged: a new write operation will correctly store the value on the struck memory cell and a new operation using the struck logic gate will provide the correct result. On FPGAs neutrons can corrupt the configuration memory, changing the implemented circuit. Such an event is still *soft* but *persistent* as the configuration memory will remain corrupted until the load of a new bitstream.

A soft error leads to: (1) no effect on the program output (i.e., the fault is masked) (2) a Silent Data Corruption (SDC) (i.e., an incorrect program output), or (3) a Detected Unrecoverable Error (DUE) (i.e., an uncorrectable memory event, a crash, or device reboot).

Modern parallel computing architectures such as GPUs or Xeon Phi have some inherent reliability weaknesses [5]. A single particle-induced fault in the scheduler or shared memory is likely to affect the correctness of several parallel threads or kernels, leading to the corruption of multiple output values [6]. Additionally, a single corrupted thread could feed thousands of future parallel threads with erroneous data, again leading to multiple errors [7].

Previous studies have already evaluated the reliability of FPGA, GPUs, and Xeon Phi running various codes through radiation experiments [8, 9] or fault-injection [10, 11, 12]. In this paper, we extend the reliability knowledge by investigating if, and how, mixed-precision data and operations affect the error rate of modern architectures.

SDCs are not always critical as some output errors can be tolerated. If the corruption affects only the least significant positions of the mantissa of a float point number the results could still be inside the intrinsic variance of float point operations. Some physical simulations accept as correct values in a range that can be as high as 4% for wave simulations [4]. Additionally, approximate computing is gaining interest in various HPC applications [3]. Moreover, the output of most neural-networks-based object detection frameworks is a vector of *tensors* containing the probability of eligible objects. Objects identified with a sufficiently high probability are classified and eventually detected. Transient faults that modify the probability without altering an object's rank or that change the coordinates of a low-probability objects are not considered critical. In this paper, we show that the FIT rate reduction as a function of tolerated output value variation depends not only on the used data precision but also on the application and on how the hardware and compiler implement operations on the different data precisions.

2.2 Mixed-precision

The adoption of approximate computing in a growing number of applications pushed the demand for mixed-precision platforms. Some applications do not require the full precision provided by double or single precisions. As the float point functional units area grows quadratically as the precision increase, using double and single-precision data even if lower precision is sufficient would add unnecessary overhead. Mixed precision have been shown to bring significant performance improvement on HPC [13], Deep Learning [14], physics simulation [15], and approximate computing [16].

Mixed-precision architectures can be used in low power devices [17], FPGAs [18], GPUs [19], general purpose CPUs [20], etc. The architecture is said to be mixed precision if it has support for at least two of the five float point arithmetics defined by the IEEE 754 (16 , 32, 64, 128, and 256 bits) [21]. Modern devices such as NVIDIA GPUs, Intel accelerators, and FPGAs, have support for half, single, and double precisions (Intel has support only for single and double) [18]. Nevertheless, other architectures support different precisions, such as 8 bits operations [1].

Recently, the efforts have been to improve the mixed-precision applications performance on modern devices. However, most of the recent research does not cover how mixed-precision arithmetics impact the device or application reliability, which is the focus of this paper.

3. EVALUATION METHODOLOGY

In this Section, we detail the architectures, benchmarks, and experimental procedure used for our study.

3.1 Architectures and Applications

FPGA: The FPGA we consider is the Zynq-7000, designed by Xilinx using a 28nm technology. The programmable array is composed mainly of Configurable Logic Blocks (CLBs), Digital Signal Processor (DSP), and embedded memory blocks (BRAM). blocks of dif-

ferent types. These components are configured by the bitstream file, which loads the user-defined circuit into the configuration memory. The number of resources used for computation is directly proportional to the described circuit complexity. When implementing the same algorithm in double, single, and half precision we expect the FPGA area used for computation and, thus, its error rate, to be directly related to the data precision (the higher the precision, the higher the area).

Intel Xeon Phi (KNC): The Xeon Phi tested is the coprocessor 3120A, which implements the *Knights Corner* architecture [22]. The Xeon Phi coprocessor has 57 physical in-order cores each of which can execute 16 single precision or 8 double precision per vector operations (half precision is not implemented). The Xeon Phi is built using a $22nm$ technology with Intel's 3-D Trigate transistors. The tested device is protected with Machine Check Architecture (MCA), which includes SECDED ECC in memory structures [22].

NVIDIA Volta: The tested GPU is the NVIDIA Titan V, designed in the Volta architecture. Volta GPUs feature hardware acceleration for three IEEE754 float point precisions: double, single, and half. Volta architecture has separated hardware for double and single/half precision operations (2,688 cores for double versus 5,376 cores for single/half). A thread can use a single precision core to execute the same instruction in two half-precision operands in parallel [23, 24]. Volta also introduces improvements on functional and arithmetical units: most instructions are performed in the same number of cycles in each chosen input precision (i.e., 8 clock cycles for double, 4 for single, and 6 for two half operations) [25, 23]. Additionally, memory hierarchy and cache policies have been improved by preventing large arrays to be evicted from cache memory if a sparse memory access is requested [25].

For our evaluation, we chose five representative benchmarks. LavaMD, Matrix Multiplication (MxM), LU Decomposition (LUD), and Microbenchmarks (Micro-MUL, Micro-ADD, and Micro-FMA). LavaMD, MxM, and Microbenchmarks are implemented in double, single, and half precision on CUDA devices. For KNC devices, LavaMD, MxM, and LUD are implemented using single and double precision. On FPGA and GPU we also test MNIST and YOLOv3 CNNs.

Matrix Multiplication (MxM), also known as GEMM, is an optimized Matrix Multiplication, which serves as a cornerstone code for several applications and performance evaluation tools [26]. GEMM is representative of highly arithmetic compute bound codes and is the core of features extraction in CNNs.

LavaMD calculates particles potential and relocation in a large 3D space due to mutual forces between them [27]. LavaMD kernel computation is mostly dot-products with float point data. LavaMD is representative of Multi-physics Particle Dynamics Code applications which solve a series of ordinary differential equations using Finite Difference Methods [28].

LUD is a linear algebra method that calculates solutions for a square system of linear equations. LUD kernel factors a matrix as the product of a lower triangular matrix and an upper triangular matrix. LUD is representative of highly CPU-bound codes [27].

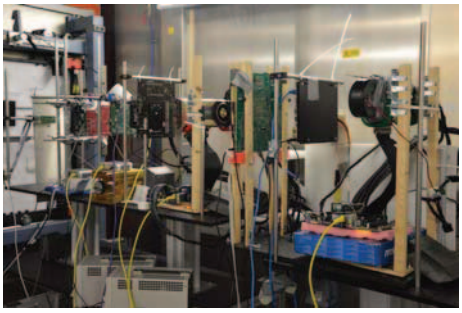


Figure 1: Radiation test setup at ChipIR.

Microbenchmarks are a set of synthetic applications designed to stress specific components of the GPU architecture. Each Microbenchmark performs a billion arithmetic operations (multiplications, additions, or fused multiply-add) on each GPU thread. Microbenchmarks are designed to minimize the stress on GPU's components other than thread's ALU and Control Unit. We keep negligible the use of global memory and caches by forcing threads to use only registers. By dispatching 256 threads per Streaming Multiprocessor we reduce the scheduling of threads [29].

Modified National Institute of Standards and Technology (MNIST) is a CNN with a topology very similar to LeNet [30]. MNIST is used for classifying 28×28 grey-scale images of handwritten digits [31]. We have tested MNIST only on FPGAs as it is a minimal network that would not exercise sufficient hardware resources on GPUs or Xeon Phis.

YOLOv3 is an open-source high accuracy real-time CNN-based object detection framework for automotive applications [32]. We execute YOLOv3 in the three precisions only on GPUs as it would not fit on FPGAs and the execution time would be too high on Xeon Phis. We run YOLOv3 on Caltech dataset [32].

It is worth noting that, to focus only on mixed-precision operation effects on CNN reliability, we do not re-trained our CNNs. We keep the same weights of the single precision version and convert them to double and half. Re-training the CNN would introduce changes in the error rate not solely related to mixed-precision execution. The accuracy of half precision version is less than 2% lower than the double one.

3.2 Neutron Beam Experiments

Neutron beam experiments are one of the most precise ways to evaluate devices and applications error rates. Faults are induced with realistic probabilities and, as the whole chip is irradiated, faults are not restricted to a subset of accessible resources like for most software fault-injection frameworks.

Fig. 1 shows part of our setup at ChipIR facility of the Rutherford Appleton Laboratory (RAL) in Didcot, UK. We irradiate two Xilinx Zynq FPGAs, two Intel Xeon-Phis, and one NVIDIA Titan-V. We select a beam

spot of 2×2 inches, which is sufficient to irradiate the chip uniformly without affecting the onboard control circuit and DRAM for the Zynq and the Xeon-Phis. The GPUs' HBM2 memories are stacked on top of the chip die. As there is no ECC available on the Titan-V, we choose to triplicate any data in the HBM2 to avoid errors in the main memory to bias our analysis.

ChipIR provides a neutron beam suitable to mimic the atmospheric neutron effects in electronic devices. The available neutron flux was about 8 orders of magnitude higher than the terrestrial flux ($13n/(cm^2 \times h)$ at sea level [33]). Each of the 30 configurations was tested for at least 100 hours, which is equivalent to more than 11,000 years of natural exposure. Since the terrestrial neutron flux is low, in a realistic application it is highly unlikely to see more than a single corruption during program execution. We have carefully designed the experiments to maintain this property (observed error rates were lower than 10^{-3} errors/execution).

Each result is compared with a pre-computed golden output. When a mismatch is detected, the execution is marked as affected by an *SDC* and the data is stored for post-processing. Both software and a hardware watchdog are used to identify *DUE* (crashes or hangs).

Experiments aim at measuring the **Failures In Time (FIT)** rate of the device executing a code. We report only *normalized* FIT rate in *arbitrary units (a.u.)* to prevent the leakage of business-sensitive data. FIT depends only on the kind and amount of resources required for computation, without considering the code execution time [34]. As data and operations precision is likely to impact both the error rate and performance, we consider performance-reliability trade-offs by calculating the **Mean Executions Between Failures (MEBF)**. MEBF is obtained dividing the error rate by the execution time and is the number of correct executions completed before experiencing a failure [35].

We also evaluate the impact of faults on the codes' output correctness by measuring how the FIT rate is reduced when we increase the **Tolerated Relative Error (TRE)**. A TRE of 0% considers any mismatch between the corrupted output and the expected value as a critical error. We then relax the correctness constraint accepting (corrupted) values in a given range as tolerable (corrected). As an example, if we consider a TRE of 10%, any outputs value between 90% and 100% of the expected value will be considered as correct or, at least, tolerable. TRE analysis is useful to compare the impact of transient faults in the execution of double, single or half operations. Single and half outputs could be compared to the double expected values to fully account for accuracy loss when moving to a lower precision data type. However, we observe a TRE lower than 2% in the absence of faults if we lower the precision.

3.3 Fault-Injection Framework

To study if and how faults propagation is affected by the use of different data types we inject faults using CAROL-FI [36], an open-source fault injector created with GDB (GNU Project Debugger) and Python.

Table 1: Benchmarks execution time on the Zynq-7000

	Execution time [s]		
Benchmark	**Double**	**Single**	**Half**
MNIST	0.011s	0.009s	0.009s
MxM	2.730s	2.100s	2.310s

An interrupt procedure injects faults in variables or registers as the application is executed on real hardware. Among the available fault injectors, such as SASSIFI [11], FTAPE [37], Ferrari [38], and BIFIT [39], we choose CAROL-FI because it supports both Intel and NVIDIA CUDA platforms and is much faster than other GDB-based fault injectors [36]. In this work, we inject, for each application, more than 2,000 faults for each data type, that is, more than 12,000 faults were injected in 6 different benchmarks. We provide the Architectural or Program Vulnerability Factor (AVF or PVF) which are the probability for a fault in a resource or in the code to affect the output [40, 41].

Beam experiments results depend on both the fault propagation and the amount of exposed resource while fault-injection results depends on fault propagation only. Combining beam and fault injection data provides a deep understanding of devices reliability.

4. FPGA

An algorithm to be executed on FPGAs needs to be synthesized into a circuit described in a *bitstream* and mapped to the device's *configuration memory*. Different algorithms may be implemented in circuits with different areas and different probabilities for faults to affect the output. However, when the same algorithm is synthesized with different data types, the implemented circuits will maintain the same structure but will have different sizes. The higher the data precision, the bigger the circuit size and, thus, a higher error rate [34]. As we will show on the next Sections, the reliability evaluation of mixed-precision in GPUs and CPUs is not as straight-forward, since the execution of mixed precision operations depends on how the device's (fixed) hardware was designed and on how the compiler maps instructions to the available computing cores.

It is important to recall that, unlike in GPUs and Xeon Phis, soft errors can have a persistent effect on FPGAs. This is because the implemented design is specified by the bitstream, which is stored in the configuration memory. If a particle corrupts the bitstream, it could change the implemented circuit. Once the circuit is compromised, the FPGA output will continuously be corrupted. To avoid a stream of corrupted outputs, we reprogram the FPGA at each observed error. Moreover, as configuration memory is the most sensitive resource, in a real application FPGAs usually adopt scrubbing mechanisms to restore corrupted bits in the configuration memory [42]. Allowing errors accumulation after observing an output error would then not provide additional information on FPGAs reliability and would not be realistic. Also unlike in GPUs and Xeon Phis, we have never observed any DUE during our experi-

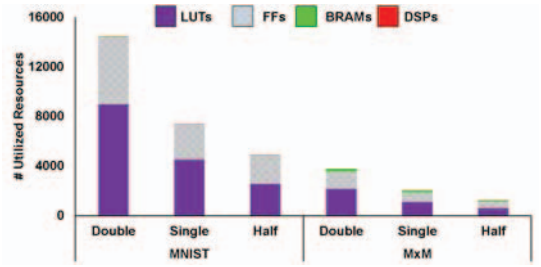


Figure 2: FPGA resources utilization.

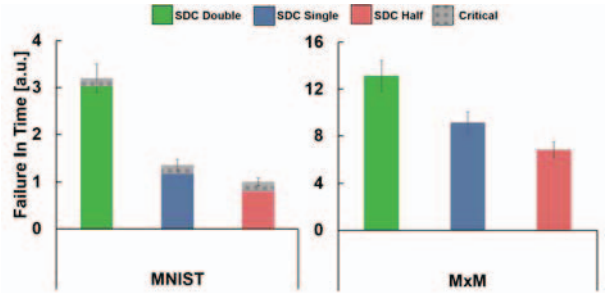


Figure 3: FIT rate of MxM and MNIST on the FPGA. No DUE was observed

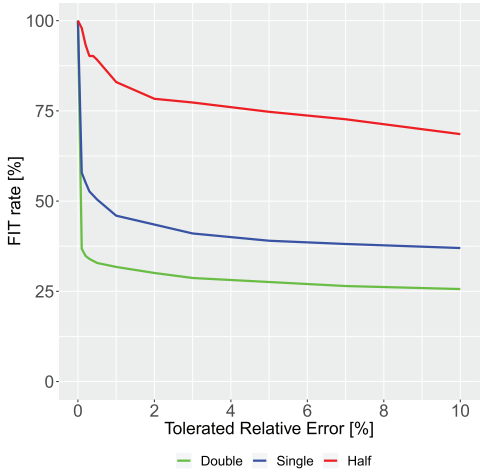


Figure 4: FIT rate reduction of MxM on the FPGA

ments with FPGAs. This is justified by the fact that the FPGA executes the code bare-metal as an implemented circuit and without synchronization nor scheduling involved. However, DUEs could be observed in FPGAs if faults are let to accumulate as after several radiation-induced modifications the circuit stops working [8].

We have implemented two algorithms using three different levels of float point precision: a 128×128 matrix multiplication (MxM) and MNIST, a CNN used for classifying 28×28 grey-scale images of handwritten digits.

4.1 Failure In Time

Table 1 lists the execution times and Figure 2 shows the number of resources (LUTs, DSPs, and BRAM) required to implement each version of our MxM and MNIST circuits. For MxM, we can see that going from double to single-precision reduces 45% the occupied area, while from single to half-precision we safe an additional 36% of area utilization. Also for MNIST, there is a considerable drop in the occupied area from double to single-precision (53%), and an additional 26% reduction when implementing the network in half-precision.

Figure 4 shows the FIT rate of the tested MxM and MNIST designs implemented on the Zynq. The distinction between critical and tolerable errors for the MNIST will be described later in this section. MNIST has a lower FIT rate compared to MxM even if it requires more resources. This is because the approximate computing nature of CNNs enables them to naturally mask a significant amount of faults (making them less architecturally vulnerable). A fault in MNIST is less likely to generate an error than a fault in MxM. In the following discussion, we limit the discussion to the comparison of different precision implementations of the same circuit. The FPGA's FIT rate decreases as we reduce precision which, as already mentioned, is expected since for the same circuit structure the FIT rate is linearly dependent on the exposed area. As a result, the trends in Figure 2 and Figure 3 are very similar. On FPGAs, we know precisely the number of gates used for computation (reported in Figure 2). We can then have a flavor of the per-gate sensitivity variation caused by the data precision dividing the number of resources required to implement the circuit by the circuit error rate. For MxM the per-gate sensitivity is 85.1 for double, 66.2 for single, and 57.1 for half-precision. The per-gate sensitivity for MNIST is 1,392 for double, 1,546 for single, and 1,560 for half-precision. These values are very close, and they confirm that the area is the primary responsible for the different error rates. It is then not surprising that by reducing precision we linearly reduce the number of used resources and, then, the FIT rate. For GPUs and Xeon Phi the relation is less obvious and depends on how the architecture and compiler have been designed.

Figure 4 shows how, by increasing the Tolerated Relative Error (TRE) the FIT rate is reduced for the half, single, and double-precision implementations of MxM. We recall that TRE is the maximum variation from the correct value that is still accepted as correct. As the tolerance level increases to 0.1%, the double-precision implementation perceives a significant FIT reduction (63% of all errors become tolerable). Interestingly, such a FIT reduction is not as significant when using single-precision, and is almost negligible using half-precision. This is because, as data precision is reduced, the probability of a fault to corrupt a more significant portion of the data is increased. Thus, whenever an error happens, output values tend to diverge more from the expected value when using lower precisions.

Figure 3 shows the FIT rate of MNIST designs separating the contribution of critical and tolerable errors. As MNIST is performing classification, the error criticality evaluation does not involve TRE. Then, output errors are marked as **Tolerable Error** if the neural network output is corrupted, but the classification is still

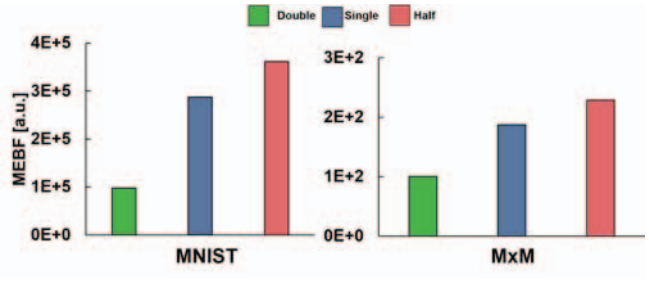


Figure 5: FPGA Mean Execution Between Failure.

Table 2: Benchmarks execution time on Xeon Phi.

	Execution time [s]	
Benchmark	**Double**	**Single**
LavaMD	1.307s	0.801s
MxM	10.612s	12.028s
LUD	1.264s	0.818s

correct or **Critical Error** if the neural network output is corrupted, and the classification is incorrect.

We can see that critical errors are the minority for all the configuration, which is a promising result for CNN-based object detection reliability. However, we can also see that the portion of errors that are to be considered critical (i.e., cause an erroneous classification) increases as we reduce precision. In fact, while only 5% of errors are considered critical using double-precision, this percentage rises to 14% and 20% for single and half-precision, respectively. Even for neural networks, the impact of faults in the output correctness is higher when we reduce the data precision and should be considered when deciding which architecture to choose.

4.2 Mean Executions Between Failures

Besides reducing the number of resources required to implement the circuits, reducing data precision also minimizes the execution time, as reported in Table 1. To evaluate the performance-reliability trade-off, we measure the Mean Executions Between Failures (MEBF) that indicates how many correct executions could be completed before experiencing a failure [35]. Figure 5 shows the MEBF for MxM and MNIST. From the reported data we can conclude that whenever we reduce data precision, we reduce the complexity of the circuit and the number of clock cycles necessary to execute the algorithm. As a result, we see that reducing precision increases the circuit reliability significantly, with the half-precision version of MxM completing about 33% more executions in between errors than single-precision. Likewise, the half-precision version of MNIST achieves, on average 26% more executions between errors than single-precision.

5. XEON PHI

Intel Xeon Phi architecture features a 512-wide Vector Processor Unit (VPU) for each of the 57 available cores. Each VPU is divided into 32 vector registers and can process 16 single precision or 8 double precision elements per operation. There is no dedicated hardware for double or single precision operations: both are executed on the same core. As Xeon Phi does not support half precision operations, for this architecture we limit our study to the different reliability behaviors for double and single. The execution time for LavaMD, MxM, and LUD are listed in Table 2.

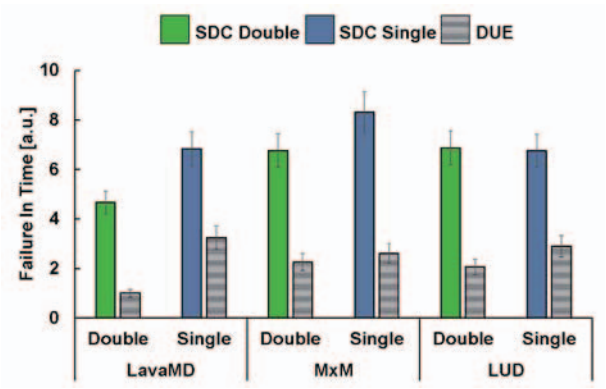


Figure 6: SDCs and DUEs FIT on Xeon Phi.

Whenever possible, the Intel compiler automatically applies optimizations such as loop interchange, unrolling and vectorizations. The compiler also improves the ability of the prefetcher to reduce memory latencies by increasing cache hits. Analyzing the optimization reports from single and double precision compilations, we observe a tendency for single precision to use more vector registers than the double precision. For LavaMD and MxM, the single version uses 33% and 47% more registers than the double version, respectively. For LUD, the main procedure uses the same number of registers for both single and double versions. A small auxiliary procedure in LUD uses more registers for the double version. Nevertheless, this procedure is active for less than 20% of the time and, thus, should not significantly impact the double and single version error rate. These different optimizations and the consequent different number of registers used for double and single precision computations are likely to affect the Xeon Phi error rate.

While Xeon Phi Machine Check Architecture protects register file from being corrupted, the higher number of instantiated registers is a symptom of higher use of hardware resources, such as functional units and internal queues (unprotected). Thus we can expect a higher resource utilization for the single precision versions of LavaMD and MxM, but eventually not for LUD.

5.1 Failure In Time

Figure 6 shows the SDC and DUE FIT rate for LavaMD, MxM, and LUD executed with single and double precision data. The SDC FIT of LavaMD and MxM for single precision is higher than the double precision. It is worth noting that we maintain the same algorithm for double or single precision. The only difference is the amount of hardware resources instantiated by the compiler to compute the two versions. The single ver-

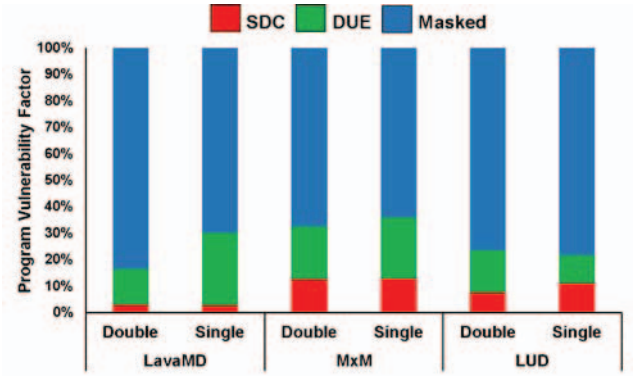


Figure 7: PVF for Xeon Phi.

sion is then more sensitive due to higher area exposed to radiation-induced faults. On the other hand, LUD uses practically the same amount of hardware resources thus resulting in a similar SDC FIT for single and double versions.

The DUE FIT in Figure 6 increases using single precision for all three codes. As control variables in the source code use integer data for both versions the area used for control variables and, then, their sensitivity, should be the same. The different DUE FIT rates are then related to the higher number of simultaneous operations, especially for a vectorized code. The VPU unit, in fact, can process 16 single precision operations or 8 double precision simultaneously. Therefore, 16 single precision ALUs use twice the number of control bits than 8 double precision ALUs, increasing the probability of faults in control bits, causing DUEs.

5.2 PVF

We injected faults using CAROL-FI with the single bit-flip fault model in random variables. Figure 7 shows that the SDC PVF for single and double is similar for each code. In other words, the data precision does not significantly affect the probability of a fault to propagate to the output. This similarity is to be expected since both versions use the same hardware and the same algorithm with negligible masking operations. Thus, for the data precision to have an impact using the same hardware the code needs to have a filter step or operation that masks faults with smaller magnitudes (i.e., faults in the least significant bits) which would reduce the sensitivity of higher precision data.

The FIT difference observed with the beam is then due to the resources usage which affects the probability of a fault to occur. We can conclude that the SDC FIT difference shown in Figure 6 is mainly due to the probability of a fault to occur than the probability of a fault to propagate to the output, which is in accordance with the higher resource utilization for the single version in MxM and LavaMD reported by the compiler.

5.3 Criticality

Figure 8 shows how the SDC FIT rate is reduced when we increase the Tolerated Relative Error (TRE). The impact of single and double precision in the output

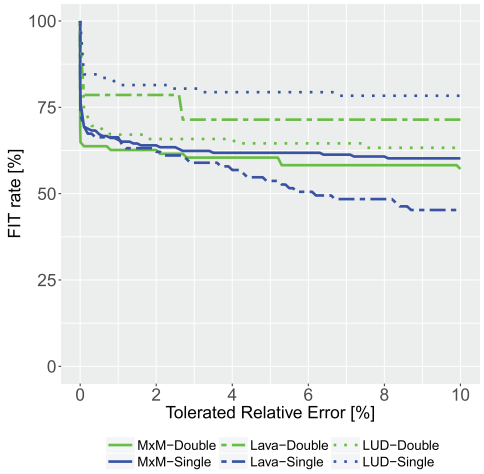


Figure 8: FIT reduction for LavaMD, MxM, and LUD on Xeon Phis.

correctness is not trivial. As seen on FPGAs, errors in double precision circuits are expected to be less critical than errors in single precision. However, on Xeon Phis the same hardware is used to execute double and single precision operations. Moreover, the algorithm also impacts profoundly how the fault propagate.

Double precision shows a better FIT rate reduction for LUD and MxM, but for MxM the difference is almost negligible. For LavaMD, shown in Figure 8, the single version has a better FIT rate reduction than double in contrast to MxM and LUD. The principal difference for LavaMD is the use of the transcendental exponentiation functions. The higher precision of double precision incurs in higher execution time and accuracy of transcendental functions [43]. Then, the stress of a higher precision leads to a worse FIT reduction when compared to single precision.

5.4 Mean Executions Between Failures

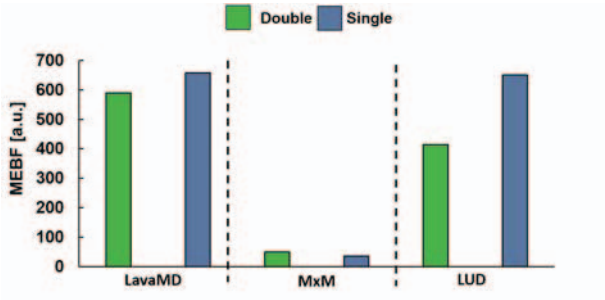


Figure 9: Xeon Phi Mean Executions Between Failure.

Figure 9 shows the MEBF for LavaMD, MxM, and LUD comparing single and double precision data. The single precision execution time gain is about 35% for LavaMD and LUD. The MEBF for LavaMD and LUD is higher for single precision since the performance gain exceed the FIT rate increase (refer to Table 2), as can be seen in Figure 9.

Table 3: Execution time on Volta GPUs.

	Execution time [s]		
Benchmark	**Double**	**Single**	**Half**
Micro MUL	6.001s	3.021s	2.232s
Micro ADD	5.993s	3.024s	2.255s
Micro FMA	5.998s	3.019s	2.260s
LavaMD	1.071s	0.554s	0.291s
MxM	2.327s	1.909s	1.180s
YOLOv3	0.133s	0.079s	0.283s

For MxM, the single precision has a performance loss of about 10%. The major difference on the Intel compiler optimization report is that the prefetch could load more elements for double than single, which results in smaller latencies compared to the single version. Therefore, the double precision shows the higher (and better) MEBF,as can be seen in Figure 9.

6. GRAPHICS PROCESSING UNITS

Unlike Xeon Phi, Volta GPUs have specific hardware cores dedicated to the execution of mixed-precision operations. The number of computing cores, the amount of resources used for computation and, then, the error rate, depends on the performed operation and the precision of data. Each Streaming Multiprocessor can deliver 95,08 GFLOP/s for 64 bits, 191,39 GFLOP/s for 32 bits or 365,71 GFLOP/s for 16 bits operands. On GPUs, the number of instantiated 32 bits registers does not change significantly between single and half precisions while for double it increases of about 2×.

The device FIT rate depends mainly on the number of parallel resources that are active during computation. There are fewer cores that execute double operations versus single and half operations (2,688 versus 5,376). This means that there are fewer active cores that could potentially be corrupted when operations are executed in double. However, as confirmed by fault-injection in Section 6.2, the cores that execute double operations are more complex (and then bigger) than the half ones. Moreover, the number of exposed bits in memory and register file is four times higher for double compared to half. Finally, the number of clock cycles to execute an operation depends only on the data precision (and not on the operation). In fact, most arithmetical operations execution take 8 clock cycles for double, 4 cycles for single, and 6 cycles for half (two half operations are executed in parallel) [25]. There are then conflicting properties of mixed-precision operations that impact the GPU reliability. To evaluate the tradeoff between operation complexity, amount of exposed resources, and active cores we first present the results obtained with specifically designed microbenchmarks. Then, we apply the analysis performed on microbenchmarks to explain experimental data obtained on MxM, LavaMD, and YOLOv3 (LUD was not tested).

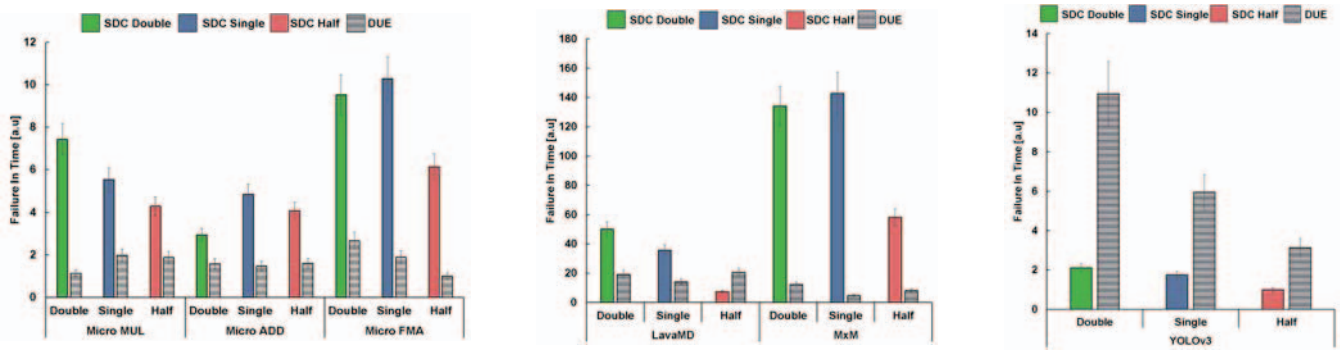
6.1 Failure In Time

Figure 10a shows the Titan V FIT rates when stressed with microbenchmarks (Micro-ADD, Micro-MUL, and Micro-FMA), tested in double, single, half precision. The difference between double, single, and half versions in Figure 10a depends on the specific operation executed and is different from FPGAs or Xeon Phis ones. The data precision impact on GPUs reliability is justified with the GPU microarchitecture characteristics. As discussed previously in this Section, not all CUDA cores are capable of performing double precision arithmetical operations. Also, the amount of per-core resources required for computation depends on the operation and employed data precision. For MUL operations the higher complexity and higher number of bits per register required to execute higher precision operations dominate the FIT rate. For ADD operations we observe the opposite trend, eventually caused by a simpler hardware. The fact of having more active cores for single and half precision masks the benefit of having fewer bits per data representation. This is also confirmed by the fact that single and half precision (both have the same number of cores available [19]) have very similar FIT rates for ADD, even if half precision requires less resources than single precision operations. FMA shows a trend that is a combination of MUL and ADD. Double precision execution has a high FIT rate because of the additional bits and core complexity. Single FIT rate is higher because of the higher number of active cores, while half benefits from the lower amount of hardware resources required for computation.

Results from Figure 10a show that FMA operations have higher FIT rates than MUL, which has higher FIT rates than ADD. This trend is explained by the complexity of the components involved in the execution of each operation. The three operations we are considering (like most operations) have the same execution latency when executed with the same data precision [25]. That is, each CUDA core needs the same number of clock cycles to execute ADD, MUL, or FMA operations using a given precision data. Because of this, the operation execution latency depends on the data precision only. This implies that the complexity and the required component size, and thus the expected error rate, should be proportional to the operation complexity, thus explaining why FMA operations perceive higher FIT rate than MUL and ADD operations. Fault-injection data in Section 6.2 confirms these statements.

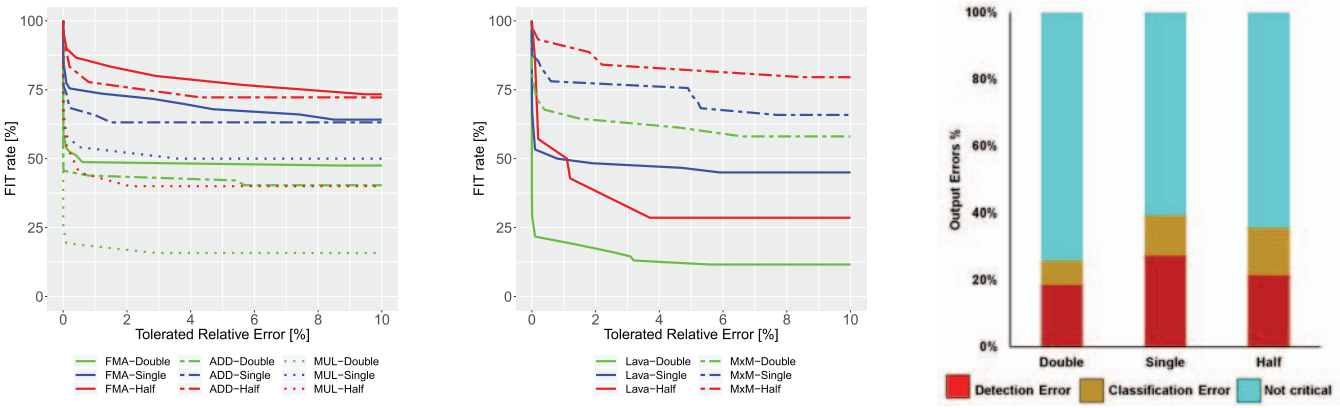
Figure 10b shows the normalized SDC and DUE FIT rates for LavaMD and MxM. From Figure 10b it is clear that MxM has a much higher FIT rate than LavaMD. This is expected as LavaMD is a compute-bound code, while MxM is memory-bound. As MxM does not take advantage of shared memory nor coalesced accesses, it suffers from longer memory latencies. This increases the exposure time of data in caches and registers that waits for main memory data to be digested or to be written back. Errors in this data are likely to propagate to the output and the longer data sitting in caches or registers is exposed, the higher the FIT rate.

The LavaMD and MxM FIT rate trend is very similar to the MUL and FMA microbenchmarks trend, respectively. More than 50% of LavaMD code is composed



(a) Microbenchmarks FIT. (b) LavaMD and MxM FIT. (c) YOLOv3 FIT.

Figure 10: Failure in Time for SDCs and DUE on Volta GPUs. All the values are shown in arbitrary units.



(a) Microbenchmarks FIT Reduction (b) LavaMD and MxM FIT reduction (c) YOLOv3 Error Criticality

Figure 11: GPUs FIT reduction for microbenchmarks, LavaMD, MxM, and error criticality percentage for YOLOv3

of MUL instructions, which is why LavaMD FIT rate trend when executed in double, single, and half follows the MUL microbenchmark one. The same applies to MxM and the microbenchmark FMA trend similarities since matrix multiplication is a series of multiply and accumulate operations, which are implemented as FMA instructions. The main difference between the realistic codes and microbenchmarks in terms of resources consumption is the use of caches and shared memory. An interesting result we get from our beam data is that the use of caches does not modify the trend between data types. This is justified by the regular memory accesses of LavaMD and MxM. Data is requested only when needed by the processing units and, once used, it is written back. In other words, the amount of data and the time data sits in cache depends on how operations are performed, which depends on the data type. Conventional cache replacement policies could undermine the performance of applications with sparse memory accesses as memory pages which will be used on future processing are evicted before being processed. NVIDIA has changed the cache policies replacement (see Section 3.1) to avoid the eviction of large arrays even in the presence of sparse memory accesses [25]. This new replacement policy is likely to maintain the same reliability trend across data types even for applications with non regular memory accesses.

Figure 10c shows the result for YOLOv3 object detection CNN. The trend is confirmed to be similar to MUL and FMA, having the half precision version with a significantly lower FIT than other data types.

GPUs run the codes bare to the metal, without the use of any operating system. The DUE FIT rate, then, has a component that dependents on the hardware and is related to the probability of corruption of the scheduler, memory addresses, and host-device interfaces. The DUE FIT rate is also influenced by a high number of control-flow or branches operations [44]. Microbenchmarks are designed to have a minimal amount of control flow operations and, in fact, their DUE rate is about 1/10 the DUE rate of LavaMD and MxM. Figure 10 reveals that, for most of the tested codes, the DUE rate variations does not significantly change with the data precision. This is expected, as the number of control flow operations is not affected by the data type. FMA and MxM show a $2x$ higher DUE rate for double compared to half, probably caused by the higher execution time and the consequent higher probability of having the controlling hardware corrupted. This trend is exacerbated for YOLOv3. In accordance with previ-

ous data in [45], object detection CNNs have a much higher probability to experience DUEs when compared to arithmetic codes.

6.2 AVF

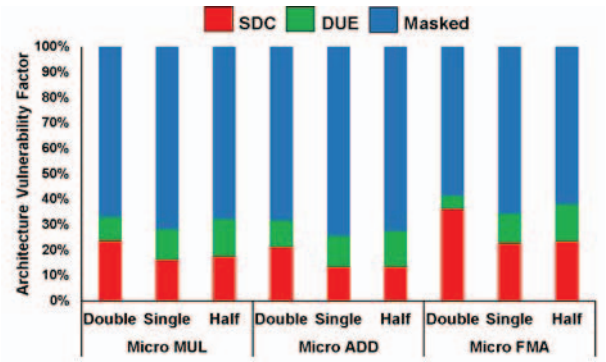


Figure 12: AVF for microbenchmarks on Volta GPUs.

Figure 12 shows the Architectural Vulnerability Factor (AVF) for the microbenchmarks (as detailed in the previous section other codes reliability characteristics could be derived from microbenchmarks). We injected a single bit flip on a randomly selected register in a random application execution time. For GPUs, we inject errors in a lower level of abstraction than for Xeon Phis to better characterize hardware faults propagation. From figure 12 it is clear that, for all the microbenchmarks, single and half precision have very similar AVF while double has a higher AVF. Single and half precision operations, then, have the same per-operation vulnerability (they use the same hardware core) while double operations need a more complex (and vulnerable) core. Combining data in Figure 12 with data in Figure 10a, we can conclude that half has a lower FIT rate than single as it uses half the amount of resources for computing the same amount of operations. Even if double computing hardware is more complex, in Volta architecture there are much fewer cores able to execute double operations compared to single/half. As a result, even if the per-operation AVF of double is higher than single or half, the FIT rate does not follow the same trend as fewer operations can be executed in parallel.

6.3 Criticality

Figures 11a and 11b show the FIT rate reduction for microbenchmarks (MUL, ADD, FMA) and realistic codes. These figures show the error rate dependence on the acceptable approximation margin (the TRE). From the data we can see that half and single have similar FIT reductions as a function of the TRE, while double benefits from a greater reduction. This result is expected since a fault on 64 bits data or on a core that execute a 64 bits operation is likely to have a lower impact on the output value than a fault on a 32 bits or 16 bits data.

ADD and FMA have a lower FIT reduction than MxM as on float point addition operands must be normalized to have the same exponent before being added. This impacts on the criticality of the instruction, eventually resulting in a different trend from MUL.

In Figure 11b we can see that there is a direct correlation between LavaMD and MUL criticality. As stated in Section 6.1, 50% of LavaMD instructions are multiplications, so it is expected that the same trend that is observed on LavaMD and MUL FIT is seen on criticality results. LavaMD FIT reduction seems to decrease faster for GPUs than for Xeon Phis (Figure 8). This trend is expected as Xeon Phi has dedicated units to process transcendental instructions and memory is protected by MCA while GPUs execute functions like exponential in software and Titan-V does not include ECC [9].

For MxM benchmark the criticality results also follow the trend observed on Figure 10b. As for ADD and FMA microbenchmarks, even for MxM half precision is the most critical data type, followed by single and double precision.

Figure 11c shows the percentage of SDCs for YOLOv3 that are tolerable (detection/classification are not affected), that change detection (bounding boxes area or position is modified), and change classification (the class of detected object is wrong). As observed for MNIST on FPGAs, half and single precision have a higher percentage of critical error than double. It is worth noting that detection errors are less dependent on data type as coordinates are expressed integer values and, thus, are not influenced by the number of corrupted digits in the mantissa.

6.4 Mean Executions Between Failures

Figure 13 shows the MEBF rates of LavaMD, MxM, YOLOv3, and microbenchmarks executed on GPUs. The MEBF rates of all tested benchmarks increase significantly when data precision is reduced. This results from the combination of lower FIT rates and shorter execution times for lower precision operations.

MUL, ADD, and FMA operations have all the same operational latencies [25]. Thus all the respective microbenchmarks have the same execution times when executed with the same data precision. Hence, the MEBF rates of microbenchmarks are inversely correlated to operations FIT rate, hence the same explanations given for the FIT trend can be applied here.

It is clear from Figure 13 that also LavaMD, MxM, and YOLOv3 reliability benefits from precision reduction. Even if realistic benchmarks' FIT rate trend is similar to the microbenchmarks one, their MEBF shows a much greater improvement brought by single and half precision, exacerbated by the shorter execution times of the reduced precision versions. The performance gain is mainly brought by the improved memory throughput, since lower precisions require fewer memory resources. On GPUs, then, we can conclude that the use of reduced precision operations could be extremely beneficial in terms of execution time and reliability mainly for codes that benefit from higher memory throughput.

7. DISCUSSION

In this section, we summarize the essential insights

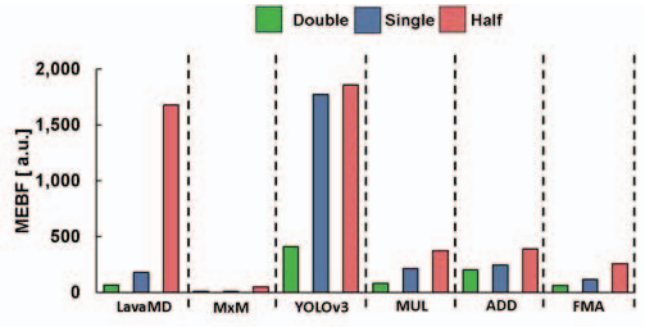


Figure 13: GPU Mean Executions Between Failure.

we can gather from the experimental data and analysis performed on FPGA, Xeon Phis, and GPUs.

As we have seen for FPGAs and Volta GPUs, the impact on the **FIT rate** is intuitive if computing resources are tailored to data precision, resulting in a reduced per-operation FIT rate for lower precisions. When a lower amount of resources is used to compute low precision data we can expect a reduced FIT rate. On the contrary, as seen for Xeon Phi, if the same hardware is used for executing double and single operations, the FIT rate is biased by how the compiler allocates resources. In other words, if the circuit for different precision data is the same, the FIT rate is dependent on how efficiently the software uses the circuit.

The **performance-reliability trade off**, is significantly improved when lower precision data is used. The higher MEBF is a result of the performance benefits of the lower precision, which can be further improved by a reduced FIT rate if the tailored hardware is employed. For FPGAs and GPUs, the smaller area used for computation and, consequently, a smaller exposed area, reduces the FIT while increasing performance. Thus, an expressive gain in MEBF is observed for half and single precision when compared to double. For the Xeon Phi, the lower precision code uses the same or more resources than a higher precision code, the performance gain generally outweighs the FIT increase and thus result in higher reliability in terms of MEBF. As a general result, then, reducing precision increases the MEBF.

A fault in a double value is less likely to significantly impact the **output correctness** than a fault in single/half value. Our data on FPGAs and GPUs with dedicated circuit for lower precision confirms that lower precision operations produce more critical errors. For architectures using the same hardware, like Xeon-Phi, the effect is not trivial and depends on the algorithm, operation's complexity, and the used data type. As Xeon Phi uses the same hardware to execute single and double precision operations, it should have the same criticality for both data precisions. However, the algorithm and operations complexity have a significant impact on the reliability of different precisions. We also show that the use of a transcendental function for Intel Xeon Phi makes the error criticality of single precision to outperform the double precision. In contrast, the GPU's transcendental units occupy just a small portion of the exposed area with a negligible impact compared to the Xeon Phi. For GPUs, we also find out that ADD and FMA operations increase the FIT for single, showing a different behavior than the MUL operation. Finally, we show that the dependence on the data type and the impact of faults in the output correctness is significant for CNNs, for which the adoption of low precision data and operations yield significant benefits.

8. CONCLUSION

We have compared the reliability of Xilinx FPGAs, Intel Xeon Phis, and NVIDIA GPUs executing codes in different float point precisions. We have seen that the use of mixed-precision data or operation significantly affects the device error rate and the code behavior in the presence of transient faults.

The experimental data highlights that the impact of mixed-precision on the device and code reliability depends on the microarchitecture and on the compiler. However, in general reducing precision is beneficial both in term of reduced error rate and improved performance, resulting in a higher number of executions correctly completed before experiencing a failure.

Additionally, we have study the impact of faults in the output correctness. In FPGAs, we have found that the lower the data precision, the higher the impact of transient faults in the output value. On Xeon Phis and GPUs, the different hardware utilization and compiler optimizations bias the fault effects in mixed-precision operations correctness. For Xeon Phis there is no significant difference between double and single while for GPUs double precision executions always benefit from a lower impact of faults in the output correctness.

9. ACKNOWLEDGMENTS

This study was partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. GPUs were donated by NVIDIA thanks to S. Hari, T. Tsai, S. Keckler, and P. Racunas. Neutron beam time was provided by ChipIR (DOI: 10.5286/ISIS.E.87856107) thanks to C. Frost and C. Cazzaniga.

10. REFERENCES

[1] S. Gupta *et al.*, "Deep learning with limited numerical precision," in *Proceedings of the 32Nd Int. Conference on Machine Learning - Volume 37*, ICML'15, pp. 1737–1746, JMLR.org, 2015.

[2] V. T. Lee *et al.*, "Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing," *CoRR*, 2017.

[3] M. A. Breuer, "Multi-media applications and imprecise computation," in *Digital System Design, 2005. Proceedings. 8th Euromicro Conference on*, pp. 2–7, IEEE, 2005.

[4] J. de la Puente *et al.*, "Mimetic seismic wave modeling including topography on deformed staggered grids," *GEOPHYSICS*, vol. 79, no. 3, pp. T125–T141, 2014.

[5] D. A. G. de Oliveira *et al.*, "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Transactions on Computers*, vol. 65, pp. 791–804, March 2016.

[6] P. Rech *et al.*, "Threads distribution effects on graphics processing units neutron sensitivity," *Nuclear Science, IEEE Transactions on*, vol. 60, pp. 4220–4225, Dec 2013.

[7] G. Li, K. Pattabiraman, *et al.*, "Understanding error propagation in gpgpu applications," in *SC16: Int. Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 240–251, Nov 2016.

[8] H. Quinn *et al.*, "Radiation-induced multi-bit upsets in sram-based fpgas," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2455–2461, 2005.

[9] D. Oliveira *et al.*, "Radiation-Induced Error Criticality in Modern HPC Parallel Accelerators," in *Proceedings of 21st IEEE Symp. on High Performance Computer Architecture (HPCA)*, ACM, 2017.

[10] B. Fang *et al.*, "Gpu-qin: A methodology for evaluating the error resilience of gpgpu applications," in *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE Int. Symposium on*, pp. 221–230, March 2014.

[11] S. K. S. Hari *et al.*, "Sassifi: An architecture-level fault injection tool for gpu application resilience evaluation," *Int. Symposium on Performance Analysis of Systems and Software*, Oct 2017.

[12] D. Oliveira *et al.*, "Carol-fi: An efficient fault-injection tool for vulnerability evaluation of modern hpc parallel accelerators," in *Proceedings of the Computing Frontiers Conference*, CF'17, (New York, NY, USA), pp. 295–298, ACM, 2017.

[13] D. GÃűddeke *et al.*, "Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in fem simulations," *Int. Journal of Parallel, Emergent and Distributed Systems*, vol. 22, no. 4, pp. 221–256, 2007.

[14] G. Venkatesh *et al.*, "Accelerating deep convolutional networks using low-precision and sparsity," in *2017 IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2861–2865, March 2017.

[15] S. L. Grand *et al.*, "Spfp: Speed without compromise - a mixed precision model for gpu accelerated molecular dynamics simulations," *Computer Physics Communications*, vol. 184, no. 2, pp. 374 – 380, 2013.

[16] N. Ho *et al.*, "Efficient floating point precision tuning for approximate computing," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 63–68, Jan 2017.

[17] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, June 2000.

[18] U. I. Minhas *et al.*, "Gpu vs fpga: A comparative analysis for non-standard precision," in *Reconfigurable Computing: Architectures, Tools, and Applications*, (Cham), pp. 298–305, Springer Int. Publishing, 2014.

[19] NVIDIA, "Nvidia tesla v100 gpu architecture - whitepaper," Tech. Rep. 1.1, NVIDIA, aug 2017.

[20] C. Lomont, "Introduction to intel advanced vector extensions," *Intel White Paper*, pp. 1–21, 2011.

[21] IEEE, "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug 2008.

[22] Intel, "Intel Xeon Phi Coprocessor System Software Developers Guide," 2014.

[23] N. Ho and W. Wong, "Exploiting half precision arithmetic in nvidia gpus," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, Sept 2017.

[24] NVIDIA, "Floating point and ieee 754 compliance for nvidia gpus."

[25] Z. Jia *et al.*, "Dissecting the NVIDIA volta GPU architecture via microbenchmarking," *CoRR*, vol. abs/1804.06826, 2018.

[26] J. Dongarra *et al.*, "TOP500 Supercomputer Sites: November 2015," 2015.

[27] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *Proceedings of the IEEE Int. Symposium on Workload Characterization (IISWC)*, pp. 44–54, 2009.

[28] L. G. Szafaryn *et al.*, "Experiences with achieving portability across heterogeneous architectures," 2010.

[29] Nvidia, "Achieved occupancy on cuda framework."

[30] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[31] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, 2012.

[32] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

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

[34] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, pp. 305–316, Sept 2005.

[35] P. Rech *et al.*, "Impact of GPUs Parallelism Management on Safety-Critical and HPC Applications Reliability," in *IEEE Int. Conference on Dependable Systems and Networks (DSN 2014)*, (Atlanta, USA), 2014.

[36] D. Oliveira *et al.*, "Experimental and analytical study of xeon phi reliability," in *Proceedings of the Int. Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, (New York, NY, USA), pp. 28:1–28:12, ACM, 2017.

[37] T. K. Tsai and R. K. Iyer, "Measuring fault tolerance with the ftape fault injection tool," in *Int. Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, (Berlin, Heidelberg), pp. 26–40, Springer, 1995.

[38] G. A. Kanawati *et al.*, "Ferrari: A flexible software-based fault and error injection system," *IEEE Transactions on computers*, vol. 44, no. 2, pp. 248–260, 1995.

[39] D. Li *et al.*, "Classifying soft error vulnerabilities in extreme-scale scientific applications using a binary instrumentation tool," in *Proceedings of the Int. Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, (Los Alamitos, CA, USA), pp. 57:1–57:11, IEEE Computer Society Press, 2012.

[40] 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 Int. Symposium on Microarchitecture*, (Washington, DC, USA), pp. 29–, IEEE Computer Society, 2003.

[41] V. Sridharan and D. R. Kaeli, "The effect of input data on program vulnerability," in *Proceedings of the 2009 Workshop on Silicon Errors in Logic and System Effects*, SELSE '09, 2009.

[42] J. Lee *et al.*, "Heterogeneous configuration memory scrubbing for soft error mitigation in fpgas," in *2012 Int. Conference on Field-Programmable Technology*, pp. 23–28, Dec 2012.

[43] J. Harrison *et al.*, "The computation of transcendental functions on the ia-64 architecture," in *Intel Technology Journal*, Citeseer, 1999.

[44] V. Fratin *et al.*, "Code-dependent and architecture-dependent reliability behaviors," in *2018 48th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks (DSN)*, pp. 13–26, June 2018.

[45] F. F. d. Santos *et al.*, "Evaluation and mitigation of soft-errors in neural network-based object detection in three gpu architectures," in *2017 47th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 169–176, June 2017.