Lindoso, A., Garcia-Valderas, M., Entrena, L., Morilla, Y. & Martin-Holgado, P. (2018). Evaluation of the Suitability of NEON SIMD Microprocessor Extensions Under Proton Irradiation. *IEEE Transactions on Nuclear Science*, 65(8), pp. 1835–1842.

DOI: 10.1109/tns.2018.2823540

Evaluation of the suitability of SIMD microprocessor extensions in radiation environments

A. Lindoso, M. García-Valderas, L. Entrena, Y. Morilla, P. Martin-Holgado

***Abstract*—This work analyzes the suitability of SIMD (Single Instruction Multiple Data) extensions of current microprocessors under radiation environments. SIMD extensions are intended for software acceleration, focusing mostly in applications that require high computational effort, which are common in many fields such as computer vision. SIMD extensions use a dedicated coprocessor that makes possible packing several instructions in one single extended instruction. Applications that require high performance could benefit from the use of SIMD coprocessors, but their reliability needs to be studied. In this work NEON™, the SIMD coprocessor of ARM microprocessors, has been selected as a case study to explore the behavior of SIMD extensions under radiation. Radiation experiments of ARM CORTEX™-A9 microprocessor have been accomplished with the objective of determining how the use of this kind of coprocessors can affect the system reliability.**

***Index Terms*—SIMD, Fault tolerance, software hardening, ARM, NEON.**

I. Introduction

Modern complex microprocessors provide SIMD (Single Instruction Multiple Data) extensions that can improve computation performance through data parallelism. SIMD makes vectorization possible: one single operation can be executed at the same time for several data. SIMD achieves a performance speed-up that is needed for high computational load applications. Computer vision, digital signal processing and any application that requires performing the very same operation over large amounts of data can achieve significant performance benefits from the use of SIMD extensions.

The use of SIMD architectures dates back to the time of vector supercomputers of the early 70s which could operate on a vector of data with a single instruction. Later, it reached the desktop computer as a means to support the increasing demand for high-performance multimedia processing. Currently, many widely-used microprocessors have upgraded their architectures to include specific SIMD extensions, such as NEON™ in the case of ARM microprocessors [1] or SSE [2] in the case of Intel and AMD microprocessors. These extensions consist in specific coprocessors that allow for vectorization.

The computational requirements for space missions are steadily increasing. Innovative space instruments generate large data volumes that need to be processed on board and the results may even be required to be used in real time for autonomous mission control operations. As the processing requirements often exceed the capabilities of space qualified processors, high-performance alternatives need to be considered, such as Digital Signal Processors (DSPs), FPGAs, multi-core processors and Graphics Processing Units (GPUs) [3], [4]. Among these alternatives, there is special interest in microprocessors endowed with a SIMD coprocessor in order to obtain a performance increase for more complex and fast sensors and circuits connected to the microprocessor.

Missions involving image processing in an autonomous way, such as RAVEN of NASA [5], are expected to become usual in the near future. NASA and ESA are developing new fault-tolerant multi-core processors in order to increase their computational power and at the same time cope with errors due to the high radiation environment [6], [7]. As a matter of fact, the use of commercial processors is being considered in space missions in order to benefit from their higher performance in comparison with rad-hard microprocessors. Even though they are not protected against errors, they could be used for less critical computations with system redundancy or in low-orbit missions with limited budget [8], and some mission types that have moderate radiation dose requirements as well as high processing power needs [9]. Their use can reduce cost, weight and at the same time increase notably the system capabilities.

When SIMD coprocessors are used in application fields that require a high reliability, such as space, it is necessary to know if they are more or less error prone than conventional processing architectures. In this work, we explore how radiation affects the system behavior when SIMD coprocessors are in use. We have selected NEON™ SIMD coprocessor of ARM microprocessor as a case study and evaluated the cross-section and performance tradeoffs. To the best of our knowledge, the suitability of SIMD coprocessors with respect to SEEs has never been tested before under radiation.

Our aim in this work is to study the effects of radiation for

Manuscript received September 29, 2017. This work has been supported in part by the Spanish Ministry of Economy and Competitiveness under projects ESP2015-68245-C4-1-P and ESP2015-68245-C4-4-P.

A. Lindoso, M. Garcia-Valderas and L. Entrena are with the Department of Electronic Technology, Universidad Carlos III de Madrid, Avda. Universidad 30, Leganés, Madrid, 28911 Spain (e-mail: alindoso@ing.uc3m.es, mgvalder@ing.uc3m.es, entrena@ing.uc3m.es).

Y. Morilla and P. Martín-Holgado are with the Centro Nacional de Aceleradores (Universidad de Sevilla, CSIC, JA). Avda. Tomás Alba Edison nº 7, E-41092 Sevilla, Spain (telephone: +34954460553, e-mail: ymorilla@us.es).

applications with intensive SIMD use and compare the results with the very same application implemented in a conventional way without SIMD extensions. To this purpose, radiation experiments have been conducted with protons at Centro Nacional de Aceleradores (Spain). The evaluation was performed with two different application benchmarks. The effects of errors were classified into different categories and analyzed for several implementations with data sets of different sizes. This information is intended to provide insight into the tradeoff between reliability and performance when SIMD extensions are used.

This paper is organized as follows. Section II summarizes the related work. Section III provides an overview of the NEON™ coprocessor. Section IV summarizes the experiments, introducing the software benchmark and the experimental setup. Section V presents the experimental results. Finally, section VI presents the conclusions of this work.

II. RELATED WORK

Errors induced by radiation in a microprocessor may cause wrong computations or even losing control of the entire system. Therefore, there is a high interest in evaluating soft error rates in microprocessors and developing techniques to mitigate them [10].

Error mitigation in microprocessors can be achieved with software and hardware techniques, or a combination of both. From the hardware point of view, hardening can be achieved by using rad-hard technologies or Radiation Hardening By Design (RHBD) at several abstraction levels. However, most RHBD techniques focus on the lower levels of abstraction: developing hardened cell libraries, adding hardware redundancy (TMR), etc. Approaches at the architectural level are not so common, mainly because architectural optimizations typically imply re-designing the processor and convey a large development effort. This kind of approaches are not suitable for COTS (Commercial Off The Shelf) devices because in such a case the architecture cannot be easily modified. Recent work proposes the use of external IPs to monitor the microprocessor behavior in a non-intrusive way [11].

There is a growing interest in evaluating and comparing existing processor architectures in order to determine which processors are best suited for high-reliability applications [12],[13]. Modern microprocessor architectures have many complex components and their usage is application dependent. In this work, we want to explore the relationship between the usage of a part of the architecture and the reliability of the microprocessor. Other works have explored architectural components. In [14] and [15] cache memories are studied to determine their resilience to radiation environments. In [16] operating systems are studied to characterize their behavior under radiation.

As the processing needs increase in many critical applications, new high-performance approaches are being used and their reliability is being measured. Among these approaches we can include multi-core and many-core architectures [17], multithreading [18], Graphics Processing Units (GPUs) [19], [20], etc.

In this work we explore SIMD coprocessors that are common for applications in many fields [21], [22]. Their usage in high reliability applications could improve considerably the system performance but it could be hazardous in terms of reliability. Xilinx has studied the reliability of Zynq SoCs with multiple radiation experiments [23]. The used workload included SIMD instructions but in a very small quantity and experiments were not carried out specifically for applications with intensive use of SIMD.

III. THE NEON™ COPROCESSOR

ARM SIMD coprocessor (NEON™) was introduced with ARMv7 architecture. The NEON™ coprocessor has a specific register bank that can be addressed as thirty-two 64-bit double word registers (D0-D31) or sixteen 128-bit quad-word registers (Q0-Q15) or a combination of both [24]. Every Q register consists of two consecutive D registers. This register file is also used for floating point operations. Figure 1 shows the structure of the register file and the structure of vector operation.

In addition to the register file, the components of NEON™ coprocessor are the following: integer execute pipeline, single precision floating-point execute pipeline and load/store and permute pipeline.

NEON™ vectors consist of several data with the same data type that includes 8-bit, 16-bit, 32-bit, 64-bit (signed or unsigned) and single precision floating point. The different data elements of a NEON™ vector are named lanes. A single instruction is used to specify a common operation which is performed for all lanes in parallel. This way, applications that must perform the same operations over large data sets can be accelerated. The performance speed-up depends on the capability of packaging data into NEON™ vectors (number of lanes).

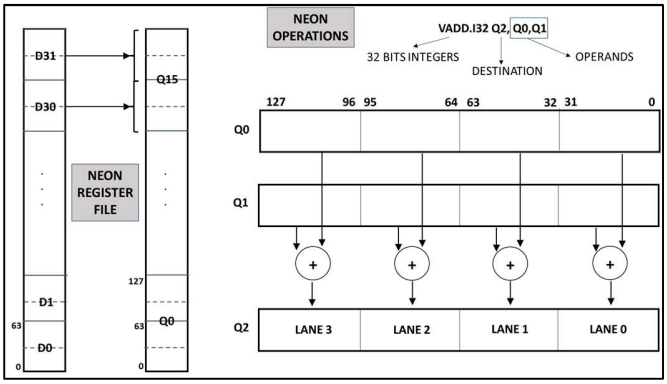

Fig. 1. NEON™ Register file and NEON™ operations.

NEON™ vectors are processed with specific vector instructions. The NEON™ instruction set includes addition, multiplication, multiplication and accumulation (MAC), multiplication and subtraction, shifting, logical operations, arithmetic operations, minimum and maximum, polynomial computations and load and store for vectors and lanes. Standard ALU flags cannot be used with SIMD instructions but a subset of the instruction set provides special comparison operations.

This kind of operations evaluate SIMD data and store results as a bitmask in a NEON™ vector. The resulting bitmask can be used with bitwise selection instructions.

Figure 1 shows an example of a NEON™ instruction that adds 4 32-bits integers with a single NEON™ instruction. All NEON™ mnemonics start with a V. The example in Figure 1 shows VADD instruction which is the mnemonic for SIMD addition. After the mnemonic, several data types can be specified. In this example, it is .I32 which stands for 32 bit integers. In the example shown in Figure 1, three Q registers (128 bits) are selected as operands and destination respectively, making possible to compute 4 additions of 32 bits with one single instruction.

Generally, NEON™ instructions use 128 bits which can be typically configured as 8 lanes of 16 bits, 4 lanes of 32 bits or 2 lanes of 64 bits. Instructions can also use modifiers that can affect the result: Q (saturation), H (halved), D (doubled before saturation) and R (rounded). There are other modifiers to change the data type of the result with respect to the operands. The relationship between operands and result is named shape and can be narrow (-N) if the width of result elements is half of the operands elements, wide (-W) if the width of result elements width doubles the operands elements, and long (-L) if the width of result and the first operand are twice the width of the second operand.

SIMD in ARM microprocessors can be used with three different approaches:

- Assembly code. Then NEON™ coprocessor has a specific extension of the instruction set. When high performance requirements must be met, ARM recommends implementing the critical parts of the code directly in assembly.
- Intrinsics. Intrinsic data types and functions provide an intermediate approach between high-level source code and assembly. Intrinsics can be seen as C functions that provide access to the NEON™ assembly set, but the compiler frees the programmer from making some low-level decisions such as register selection.
- Automatic vectorization. The compiler is prepared to vectorize software into the SIMD instruction set. To activate automatic vectorization, maximum optimization and specific compilation flags regarding NEON™ must be used. If this is the preferred choice, the code must be adapted to maximize the compiler capability to detect vectorization. When automatic vectorization is selected, the assembly code must be checked to make sure that vectorization has been correctly accomplished by the compiler.

IV. Description of the Experiments

The evaluation has been carried out with two different benchmarks that are suitable for the utilization of SIMD. Several variations of the benchmarks have been developed according to the needs of the different performed experiments. In the following subsections the different benchmarks and their variations are explained in a detailed manner. All the software benchmarks have been adapted for the experimental setup described in section IV.C.

A. Matrix multiplication

This benchmark is generally proposed as an SIMD application example and is also commonly used for radiation testing of microprocessors [13].

Three different versions of the matrix multiplication code are compared:

- Matrix multiplication.
- NEON™ version of matrix multiplication. This code is a modified version of matrix multiplication software using SIMD extensions. Data and loops were arranged in order to exploit parallelism on 32-bit multiply, addition and accumulation for groups of four 32 bits data. Intrinsics and automatic vectorization have been used to implement this software.
- Data-flow hardened matrix multiplication. In order to be able to categorize the possible errors, we have developed another software version of matrix multiplication that implements data-flow hardening. As we are studying SIMD that implies intensive data handling, our purpose is to focus on data errors. The hardening approach is based in data-flow duplication [25] and inverted branches [26].

The main characteristics of the matrix multiplication benchmark versions in terms of performance are shown in Table I. Columns show the characteristics of the different matrix multiplication software codes: MMULT (matrix multiplication), MMULT N20 (NEON™ implementation) and MMULT DH (data-flow hardened). Table I rows show respectively the number of clock cycles, the execution time at the nominal clock frequency (650 MHz,) and performance improvement of MMULT N20 and MMULT DH implementations with respect to the basic implementation (MMULT). Results of Table I have been obtained with 20x20 arrays of 32 bits input data. Cache memories were disabled in all cases, as they could introduce significant performance variations.

TABLE I: BENCHMARKS SUMMARY I (MMULT)

	MMULT	**MMULT N20**	**MMULT DH**
#cycles	50442	20877	183190
Time(μs)	77.60	32.12	281.83
Performance increase	-	2.42x	0.28x

Table I shows that MMULT N20 software version achieves 2.42x performance speed-up. The execution time reported in Table I for all benchmarks does not include boot and initial configuration of microprocessor elements, such as communication with an external monitor, that are used in the code. It must be noted that the reported time in the case of MMULT N20 software includes data arrangement to improve parallelization. The hardened software version (MMULT DH) presents a performance penalty of 3.63 times with respect to the original version.

Additional benchmarks were obtained by varying the size of the matrices being multiplied. Table II reports the performance characteristics of the MMULT NEON™ software for the selected matrix sizes: 8x8 (MMULT N8), 48x48 (MMULT N48) and 128x128 (MMULT N128). The last row of Table II reports the relative performance with respect to the benchmark MMULT N20 (20x20 matrix) whose characteristics are reported in Table I. As expected, the time required to execute each benchmark grows proportionally to the number of required MAC operations.

TABLE II: BENCHMARKS SUMMARY II (SIZE VARIATION)

	MMULT N8	**MMULT N48**	**MMULT N128**
#cycles	1566	275403	7672813
Time(µs)	2.41	423.70	11804.33
Performance increase	32.21x	0.18x	0.01x

B. Wavelet Transform

Wavelet transform is commonly used in the field of image processing [27]. Wavelet performs iterative 2-dimensional filters of different subsets of pixels of an image. The selected family of filters and the way data is selected for the computations may vary. Wavelet transform can be accurate in time or frequency and is very useful when multiresolution is required. Wavelet operation is based in filtering, which is also a typical application example for SIMD instructions.

For the experiments, two different versions of the wavelet transform benchmark have been developed, namely wavelet transform and NEON™ implementation of the wavelet transform. Our wavelet implementation uses a Level 1 dyadic Haar wavelet [27].

The main characteristics of the software versions in terms of performance are shown in Table III. Columns report the characteristics of the two versions: wavelet transform using fixed point data (WT) and NEON™ implementation of wavelet transform (WT N). Results provided in Table II do not use cache memories and input data are 32x32 arrays of 16 bits. The last row of Table III reports the performance increase of WT N benchmark with respect to WT benchmark. The NEON™ wavelet implementation increases performance by a factor of 5.15 times.

TABLE III: BENCHMARKS SUMMARY (III) (WAVELET)

	WT	**WT N**
#cycles	261029	50716
Time(µs)	401.58	78.02
Performance increase	-	5.15x

C. Experimental setup

Experiments took place during two different proton irradiation campaigns in April and August of 2017 at CNA (Centro Nacional de Aceleradores), Spain. The experiments were performed using the external beam line installed in the 18/9 IBA compact cyclotron. The DUTs were irradiated in open air with up to 15.4 MeV protons, with 300 keV energy spread. The uniformity of the flux was better than 90% in the exposed area of interest, maintaining the flux stability within 5% during each experiment.

For the experiments, one Zybo board [28] was used per software code, giving a total of 8 boards. Zybo contains a XC7Z010 device from Zynq-7000 FPGA family [29] of Xilinx, which contains a dual core ARM CORTEX™-A9 with one NEON™ coprocessor for SIMD per core. One single core of the Zybo board was used for each experiment running at the nominal 650 MHz clock frequency. The Zybo board is connected through USB with an external monitor that controls the experiment. The external monitor collects the errors that are sent from Zybo through USB connection and resets and program again the device when the error is not recoverable. Boot, bitstream and software code are stored in an SD card that is copied to OCM (On Chip Memory) in the programming and reset process. The programmable logic was not used during the experiments. Both the Zybo board and the external monitor were biased independently. For the experiments all algorithms of the different software versions were executed in an infinite loop.

A picture of the experimental setup is shown in Figure 2.

Fig. 2. Experimental setup

D. Error classification

Observed errors during the experiments were categorized as follows:

- SDC (Silent Data Corruption) errors: After every execution of the algorithm the result is compared with a golden result. Whenever there are differences between the actual result and the golden results, an SDC error will be triggered by the microprocessor and sent to the external monitor.
- Hang errors: The microprocessor is stuck at an infinite loop or produces an abnormal termination that requires external intervention, in our case, reprogram and boot from the external monitor. Hang errors are detected by a

timeout condition that is working as a watchdog timer. Our external monitor has also a timeout condition, so that Hang errors can be detected even in the case the internal mechanism does not work properly.

- Communication errors: The microprocessor cannot communicate correctly with the external monitor. This type of events typically represent a very small amount of the observed errors and they do not occur in all the experiments. In this category we have also included possible problems when device programming takes place. During the complete set of experiments we observed some errors that affected the selection of the location of the programming file of the device. It provoked the loss of control of the circuit by the external monitor. This type of event only occurred twice and took place in two different experiments.
- Data errors: Software detects a data error. This type of errors can only be detected by the software version that is data-flow hardened (DH). Our hardened software duplicates all variables and instructions, and compare variable values with the copies. Whenever a discrepancy appears, the microprocessor triggers a data error that will be collected by the external monitor.

In the following section, cross-sections and percentages of the different types of errors observed for all the experiments are reported. The cross section of a single error category is computed by dividing the number of observed events of every error type by the total fluence, similar to the Hang cross-section reported in [30].

V. Experimental results and discussion

A. Matrix multiplication results

Table IV reports at every column a summary of the parameters used for the experiments of matrix multiplication software, namely: MMULT (matrix multiplication), MMULT N20 (Matrix multiplication optimized with NEON™) and MMULT DH (Data-flow Hardened version of matrix multiplication). For these experiments, all the software codes used 20x20 arrays of 32-bit data.

TABLE IV: EXPERIMENTS PARAMETERS I (MMULT)

	MMULT	**MMULT N20**	**MMULT DH**
Energy (MeV)	15.4	15.4	15.4
Flux (p/cm^2·s)	$1.7 \cdot 10^{8}$	$5.2 \cdot 10^{8}$	$5.2 \cdot 10^{8}$
Fluence (p/cm^2)	$1.2 \cdot 10^{12}$	$1.0 \cdot 10^{12}$	$9.4 \cdot 10^{11}$

Table V shows the cross-section (cm^2) obtained in the experiments for codes MMULT, MMULT N20 and MMULT DH. For the sake of comparison, cross-section is reported for all error categories and for the total number of observed events. The last column provides the total cross-sections and the 95% confidence intervals. These confidence intervals are computed using the standard deviation with a normal approximation of the Poisson mean.

The final row of Table V reports the increase in cross-section that is produced when MMULT software is compared with MMULT N20 software for all the different error categories.

TABLE V: RADIATION TEST RESULTS I (MMULT)

	SDC (cm^2)	**HANG** (cm^2)	**Comm.** (cm^2)	**TOTAL** (cm^2)
MMULT	$6.67 \cdot 10^{-11}$	$1.08 \cdot 10^{-11}$	$5.83 \cdot 10^{-12}$	$8.33 \cdot 10^{-11}$ ($6.70 \cdot 10^{-11}$, $9.97 \cdot 10^{-11}$)
MMULT N20	$1.12 \cdot 10^{-10}$	$4.70 \cdot 10^{-11}$	$7.00 \cdot 10^{-12}$	$1.67 \cdot 10^{-10}$ ($1.41 \cdot 10^{-10}$, $1.91 \cdot 10^{-10}$)
MMULT DH	$8.08 \cdot 10^{-11}$	$5.00 \cdot 10^{-11}$	$3.19 \cdot 10^{-12}$	$2.90 \cdot 10^{-10}$ ($2.56 \cdot 10^{-10}$, $3.25 \cdot 10^{-10}$)
MMULT N20 /MMULT	1.68	4.34	1.20	1.99

The results show that MMULT N20 code presents a significant increase in cross-section for all categories except for communication. Anyhow, the cross-section of communication errors is very similar in all cases and much smaller than that of of SDC or Hang errors. Considering the total cross-section, we observe an increase of 1.99 times for MMULT N20 code. The increase is 1.68 times for SDCs and 4.34 times for Hangs. It is interesting to note that the performance increase provided by the MMULT N20 code (2.4 times) is higher than the cross-section increase (1.99 times), so that the Mean Work To Failure (MWTF), defined as the amount of work completed between failures [31], slightly improves when NEON™ coprocessor is used.

For the MMULT DH benchmark, cross-section increases 3.49 times with respect to the MMULT benchmark, which approximately matches the performance decrease (3.63) reported in section IV.A. However, the cross-section of data detected errors for this benchmark is $1.57 \cdot 10^{-10}$, which stands for 53.85% of the observed errors. Therefore, the cross-section of undetected errors reduces to $1.34 \cdot 10^{-10}$. Even so, data-flow hardening was not sufficient to improve cross-section with respect to the conventional MMULT code and it is only slightly better than the NEON™ code, MMULT N20.

With regard to the error categories, both SDC and Hang errors increase for the MMULT DH benchmark, but the relative relevance of each of them varies significantly. This benchmark presents a large reduction of SDC errors when compared with MMULT benchmark: from 80% (MMULT) to 27.84% (MMULT DH). This reduction is almost equivalent to the percentage of detected errors, which can be expected because the data-flow hardening techniques mainly mitigate this type of errors. In the case of Hang errors, the variation is small and remains in the same order of magnitude, from 13% (MMULT) to 17.21% (MMULT DH). The increase of Hang errors seems to be related to the increase of execution time of the DH benchmark. Communication errors in the case of MMULT DH are reduced to 1.1% while MMULT has 7%. The variation of communication errors seems to be affected by the execution time of the benchmark that is irradiated, with a tendency to

show a smaller proportion of errors when execution time is larger.

B. *Variation with the size of the data set*

Another set of experiments were performed in order to evaluate the cross-section variation with the size of the data set used by the software. For these experiments, we selected the matrix multiplication benchmarks optimized for NEON™ for several matrix sizes: 8x8 (MMULT N8), 20x20 (MMULT N20), 48x48 (MMULT N48) and 128x128 (MMULT N128).

Table VI reports a summary of the parameters used in the experiments and Table VII shows the cross section (cm^2) for every error category and for the total errors. The last column of Table VII reports the total cross-section and the 95% confidence intervals. MMULT N20 parameters and cross-section have been included respectively in Tables VI and VII for the sake of comparison.

Table VI shows that the flux was reduced for the experiments with larger data sets. During the experiments we observed a significant increase in the error rate in these cases, and we had to reduce the flux in order to be able to properly monitor the events.

TABLE VI: EXPERIMENTS PARAMETERS II (SIZE VARIATION)

	MMULT N8	**MMULT N20**	**MMULT N48**	**MMULT N128**
Energy (MeV)	15.4	15.4	15.3	15.3
Flux ($p/cm^2 \cdot s$)	$4.6 \cdot 10^8$	$5.2 \cdot 10^8$	$1.7 \cdot 10^8$	$9.5 \cdot 10^7$
Fluence (p/cm^2)	$8.3 \cdot 10^{11}$	$1.0 \cdot 10^{12}$	$1.5 \cdot 10^{11}$	$1.0 \cdot 10^{11}$

TABLE VII: RADIATION TEST RESULTS II (SIZE VARIATION)

	SDC (cm^2)	**HANG** (cm^2)	**Comm.** (cm^2)	**TOTAL** (cm^2)
MMULT N8	$2.53 \cdot 10^{-11}$	$3.25 \cdot 10^{-11}$	$7.23 \cdot 10^{-12}$	$6.51 \cdot 10^{-11}$ ($4.77 \cdot 10^{-11}$, $8.24 \cdot 10^{-11}$)
MMULT N20	$1.12 \cdot 10^{-10}$	$4.70 \cdot 10^{-11}$	$7.00 \cdot 10^{-12}$	$1.67 \cdot 10^{-10}$ ($1.41 \cdot 10^{-10}$, $1.91 \cdot 10^{-10}$)
MMULT N48	$1.6 \cdot 10^{-10}$	$7.40 \cdot 10^{-10}$	$1.33\ 10^{-11}$	$9.13 \cdot 10^{-10}$ ($7.60 \cdot 10^{-10}$, $1.07 \cdot 10^{-9}$)
MMULT N128	$3.00 \cdot 10^{-11}$	$1.18 \cdot 10^{-9}$	0	$1.21 \cdot 10^{-9}$ ($9.94 \cdot 10^{-10}$, $1.43 \cdot 10^{-9}$)

Table VII shows that the total cross-section increases with increasing array size. This increase roughly correlates with the dimension of the matrices.

Regarding the error type, there is a remarkable increase of Hang errors with increasing array size, while SDC errors show a much moderate increase or even a decrease in the case of the largest tested matrix size (128x128). Figure 3 shows the percentage of errors of the different categories (SDC, Hang and communication) for the NEON™ implementations of MMULT with different matrix sizes (N8, N20, N48 and N128). For the smaller data set (N8), the percentages of SDC and Hang errors are of the same order of magnitude. The percentage of SDC errors first grows as the size increases (N20) but then Hang errors clearly dominate for the larger sizes.

A possible explanation for this behavior is the fact that the use of NEON™ instructions increases together with the execution time for large data sets. When the SIMD coprocessor is used more intensively, events can cause errors in the SIMD coprocessor pipeline that lead to Hang errors. The compiler optimization level for all the NEON™ optimized benchmarks was required to be the maximum (-O3). When this optimization level is used, program instructions are reordered to speed up execution. In this case, there is a higher probability for errors to affect the control flow.

These experiments show that data size is very relevant when SIMD coprocessor is used and must be considered when software is designed.

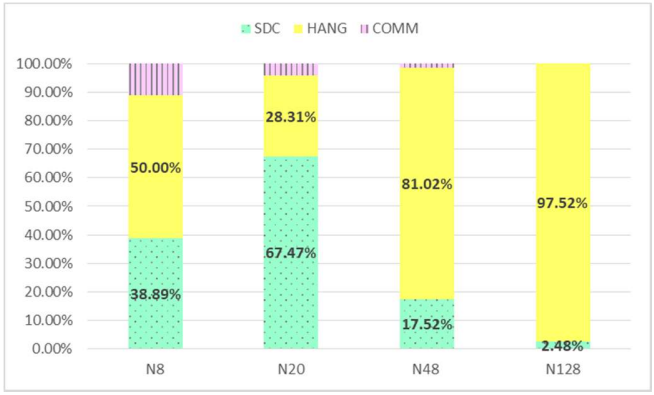

Fig. 3. MMULT size variation results (Percentage of observed errors)

C. *Wavelet Transform*

Table VIII reports a summary of the main parameters for the experiments using wavelet transform benchmarks, namely WT and WT N (NEON™ version of wavelet transform), and Table IX shows the measured cross sections (cm^2). Input data size for both software versions was 32x32 arrays of 16 bit data. The last column of Table IX reports the total cross-section and the 95% confidence intervals.

TABLE VIII: EXPERIMENTS PARAMETERS III (WAVELET)

	WT	**WT N**
Energy (MeV)	15.3	15.3
Flux ($p/cm^2 \cdot s$)	$4.5 \cdot 10^8$	$4.3 \cdot 10^8$
Fluence (p/cm^2)	$8.8 \cdot 10^{11}$	$8.6 \cdot 10^{11}$

TABLE IX: RADIATION TEST RESULTS III (WAVELET)

	SDC (cm^2)	**HANG** (cm^2)	**Comm.** (cm^2)	**TOTAL** (cm^2)
WT	$1.16 \cdot 10^{-10}$	$3.40 \cdot 10^{-11}$	$6.81 \cdot 10^{-12}$	$1.57 \cdot 10^{-10}$ ($1.30 \cdot 10^{-10}$, $1.82 \cdot 10^{-10}$)
WT N	$1.03 \cdot 10^{-10}$	$4.30 \cdot 10^{-11}$	$5.81 \cdot 10^{-12}$	$1.52 \cdot 10^{-10}$ ($1.26 \cdot 10^{-10}$, $1.78 \cdot 10^{-10}$)
WT N/ WT	0.89	1.26	0.85	0.97

The results of Table IX show that the NEON™ version of the wavelet benchmark (WT N) slightly reduces the cross-section. Regarding the different error categories, both SDC and communication errors slightly decrease but Hang errors increase by a factor of 1.26. This increase of Hang errors when NEON™ is in use shows the same behavior as the MMULT benchmark. The total cross-section slightly decreases due to the smaller quantity of SDC and communication errors.

Taking into account both the cross-section and the speed-up, it is clear that the use of the NEON™ coprocessor is highly beneficial for this benchmark, because it can be sped up by more than 5 times with a similar cross-section.

Figure 4 shows a comparison of the percentages of every type of error for MMULT, MMULT N20, WT and WT N. We can observe that when the SIMD coprocessor is used, the percentage of SDC errors decreases and the percentage of Hang errors increase for both benchmarks. The increase of Hang errors is more noticeable in the case of the matrix multiplication benchmark. Figure 4 also shows that the NEON™ versions of both benchmarks provide similar results in percentage. However, when the SIMD coprocessor is not used, the wavelet transform benchmark presents a larger percentage of Hang errors.

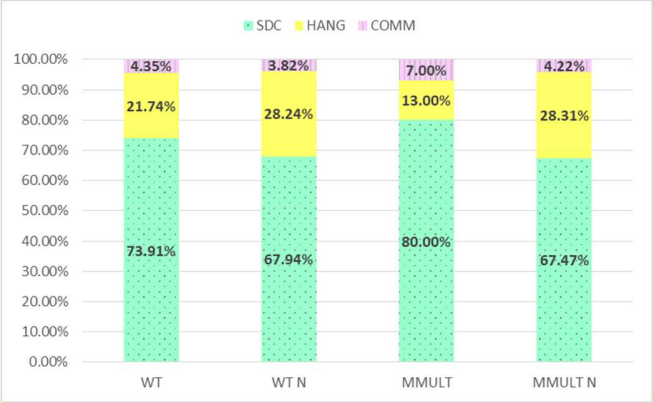

Fig. 4. Comparison of error categories for MMULT and WT vs. NEON™ implementations.

VI. Conclusions

This paper has analyzed the suitability of SIMD (Single Instruction Multiple Data) extensions under radiation with a case study based on ARM CORTEX™-A9 and its NEON™ SIMD coprocessor. SIMD extensions are present in many modern microprocessors, including ARM, Intel and AMD architectures, in order to improve their performance for applications with a high computational load, such as multimedia applications. As the computational requirements for space missions are steadily increasing, SIMD extensions may become a solution for onboard processing of large amounts of data.

Zynq SoC devices with ARM Cortex™-A9 hard-core microprocessors were irradiated with protons with an energy up to 15.4 MeV. Eight different boards were used with several versions of two typical data-intensive code benchmarks, namely matrix multiplication and wavelet transform.

Experimental results demonstrate that the use of the NEON™ coprocessor notably improves performance but can also increase cross-section. However, the performance increase is generally higher than the cross-section increase, so that the Mean Work To Failure improves when NEON™ coprocessor is used.

Experiments were also performed for data sets of different sizes in order to evaluate how this may affect cross-section. The results show that the total cross-section increases with the size of the workload. In particular, there is a remarkable increase of Hang errors for large data sets, which make a more intensive use of the SIMD coprocessor. These results show that data size is very relevant when SIMD coprocessor is used and must be considered when the software application is designed.

References

[1] ARM Inc, “ARM architecture reference manual”, 2000.

[2] Intel Inc., “Intel® 64 and IA-32 Architectures, Software Developer’s Manual, Volume 1: Basic Architecture”, 2016.

[3] R. Trautner, "ESA's roadmap for next generation payload data processors". Proc. Data Systems In Aerospace (DASIA), 2011.

[4] R. L. Davidson, C. P. Bridges, "Adaptive multispectral GPU accelerated architecture for Earth Observation satellites," 2016 IEEE International Conference on Imaging Systems and Techniques (IST), Chania, pp. 117-122, 2016.

[5] M. Strube, R. Henry, E Skeleton, J. V. Eepoel, N. Gill, R. McKenna, “Raven: An On-Orbit Relative Navigation Demonstration Using International Space Station Visiting Vehicles”, AAS Guidance and Control Conference, 2015.

[6] J. Andersson, J. Gaisler, R. Weigand, “The next generation multipurpose microprocessor”, Proc. of Data Systems In Aerospace (DASIA), 2010.

[7] R. Doyle, et al, “High performance spaceflight computing (HPSC) next-generation space processor (NGSP): a joint investment of NASA and AFRL”, Proc. of the Workshop on Spacecraft Flight Software, 2013.

[8] D. Sinclair, J. Dyer. “Radiation effects and COTS parts in SmallSats”, Small Satellite Conference, 2013.

[9] R. Trautner, R., Vitulli, "Ongoing developments of future payload data processing platform at ESA", Second International Workshop on On-Board Payload Data Compression (OBPDC), 2010.

[10] M. Nicolaidis, “Soft errors in modern electronic systems”, Springer, 2011.

[11] L. Parra, A. Lindoso, M. Portela-Garcia, L. Entrena, B. Du, M. Sonza Reorda, L. Sterpone, “A New Hybrid Nonintrusive Error-Detection Technique Using Dual Control-Flow Monitoring”, IEEE Transactions on Nuclear Science, vol. 61, no. 6, pp. 3236-3243, Dec. 2014.

[12] H. Quinn, Z. Baker, T. Fairbanks, J. L. Tripp, G. Duran, “Software Resilience and the Effectiveness of Software Mitigation in Microcontrollers”, IEEE Transactions on Nuclear Science, vol. 62, no. 6, pp. 2532-2538, Dec. 2015.

[13] H. Quinn et al. “Using Benchmarks for Radiation Testing of Microprocessors and FPGAs”, IEEE Transactions on Nuclear Science, vol. 62, no. 6, pp. 2547-2554, Dec. 2015.

[14] L. Antunes Tambara, F. L. Kastensmidt, N. H. Medina, N. A. Vitor, A. P. Aguiar, F. Aguirre, E. L. A. Macchione, M. A. G. Silveira, “Heavy Ions Induced Single Event Upsets Testing of the 28 nm Xilinx Zynq-7000 All Programmable SoC”, IEEE Radiation Effects Data Workshop (REDW), pp. 1 – 6, 2015.

[15] T. Santini, P. Rech, G. L. Nazar, F. Rech, “Beyond Cross-Section: Spatio-Temporal Reliability Analysis”, ACM Transactions on Embedded Computing Systems, vol. 15, no. 1, Article 3, 2015.

[16] T. Santini, L. Carro, F. R. Wagner, P. Rech, “Reliability Analysis of Operating Systems and Software Stack for Embedded Systems”, IEEE Transactions on Nuclear Science, vol. 63, no. 4, pp. 2225 – 2232, Aug. 2016.

[17] V. Vargas et al., "Radiation Experiments on a 28 nm Single-Chip Many-Core Processor and SEU Error-Rate Prediction," IEEE Transactions on Nuclear Science, vol. 64, no. 1, pp. 483-490, Jan. 2017.
[18] S. S. Mukherjee, M. Kontz and S. K. Reinhardt, "Detailed design and evaluation of redundant multi-threading alternatives", Proc. 29th Annual International Symposium on Computer Architecture, pp. 99-110, 2002.
[19] P. Rech, T. D. Fairbanks, H. M. Quinn, L. Carro, "Threads Distribution Effects on Graphics Processing Units Neutron Sensitivity," IEEE Transactions on Nuclear Science, vol. 60, no. 6, pp. 4220-4225, Dec. 2013.
[20] D. A. 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, March 2016.
[21] R. Wang, J. Wan, W. Wang, Z. Wang, S. Dong, W. Gao, "High definition IEEE AVS decoder on ARM NEON platform", IEEE International Conference on Image Processing, pp. 1524 – 1527, 2013.
[22] H. Yong, R. Wang, W. Wang, Z. Wang, S. Dong, B. Han, W. Gao, "Acceleration of HEVC transform and inverse transform on ARM NEON platform", International Symposium on Intelligent Signal Processing and Communication Systems, pp.169-173, 2013.
[23] A. Lesea, W. Koszek, G. Steiner, G. Swift, D. White, "Soft error study of ARM SoC at 28 nanometers", Proc. IEEE Workshop on Silicon Errors in Logic – System Effects, 2014.
[24] ARM Inc., "NEON programmer's guide", 2013.
[25] M. Rebaudengo, M. S. Reorda, M. Torchiano, M. Violante, "Soft-error detection through software fault-tolerance techniques," Proc. International Symp. on Defect and Fault Tolerance in VLSI Systems, pp. 210–218, 1999.
[26] J. R. Azambuja et al., "A fault tolerant approach to detect transient faults in microprocessors based on a non-intrusive reconfigurable hardware," IEEE Transactions on Nuclear Science, vol. 59, no. 4, pp. 1117–1124, Aug. 2012.
[27] R.C. Gonzalez, R.E. Woods, "Digital Image Processing", Pearson, 2008.
[28] Digilent Inc, "Zybo reference manual", 2014
[29] Xilinx Inc.: "Zynq-7000 All Programmable SoC: Technical Reference Manual", UG585, 2016.
[30] F. Irom, "Guideline for Ground Radiation Testing of Microprocessors in the Space Radiation Environment", Jet Propulsion Laboratory, 2013.
[31] G. A. Reis, J. Chang, N. Vachharajani, S. S. Mukherjee, R. Rangan, and D. I. August, "Design and evaluation of hybrid fault-detection systems," Proc. International Symp. on Computer Architecture, pp. 148–159, 2005.