GPU4S (GPUS FOR SPACE): ARE WE THERE YET?

Leonidas Kosmidis[2,1], **Iván Rodriguez-Ferrandez**[1,2], **Alvaro Jover-Alvarez**[1,2], **Guillem Cabo**[1,2], **Sergi Alcaide**[1,2], **Jérôme Lachaize**[3], **Olivier Notebaert**[3], **Antoine Certain**[3], **and David Steenari**[4]

[1]*Universitat Politècnica de Catalunya (UPC), Barcelona, Spain*
[2]*Barcelona Supercomputing Center (BSC), Barcelona, Spain*
[3]*Airbus Defence and Space, Toulouse, France*
[4]*European Space Agency, The Netherlands*

ABSTRACT

Embedded GPUs have been identified from both private and government space agencies as promising hardware technologies to satisfy the increased needs of payload processing. The GPU4S (GPU for Space) project funded from the European Space Agency (ESA) has explored in detail the feasibility and the benefit of using them for space workloads. Currently at the closing phases of the project, in this paper we describe the main project outcomes and explain the lessons we learnt. In addition, we provide some guidelines for the next steps towards their adoption in space.

1. THE GPU4S PROJECT

The space industry is facing a dramatic increase in the performance required by future missions as forthcoming spacecraft require to acquire orders of magnitude more data compared to existing ones, supporting much higher resolutions, precision and sampling frequencies. Moreover, other types of space missions like robotic exploration such as the ExoMars rover [1] and new types of space missions and concepts like the space tug [2] and active debris removal [3] or the Mars Helicopter [4] that uses a COTS processor for operation, all of them require highly autonomous operations, which need significant on-board processing capabilities.

Embedded Graphics Processing Units (GPUs) have shown a great potential in high performance processing in temperature and battery-constrained devices, following the widespread and successful use of GPUs in high-performance domain. For this reason, there is a lot of investment in GPU studies funded from government space agencies [5][6][7][8] as well the private sector [9][10]. Some of these works focus on radiation studies of certain GPU products, while others implement closed source space applications. However, each of these works is isolated and uses its own hardware and software, the latter frequently limited from export control due to space's sensitive nature in defence applications, which makes impossible to compare against one another and draw general conclusions about the use of GPUs and their adoption in space.

The GPU4S (GPU for Space) [7] project funded by the European Space Agency (ESA) which is currently at its closing phase, aims at evaluating the potential of embedded GPUs for use in space, for the first time and define the roadmap of GPU adoption in space. In order to achieve this long term goal, several intermediate steps had to be achieved.

First, we had to confirm that the existing and mainly future space software can be effectively parallelised in order to exploit the performance advantage of GPUs – as they are known to work well with certain types of algorithms which require massively parallel processing, but also exhibit regular behaviour in memory accesses and branching.

To achieve this, we needed to ensure that programming GPUs can be mastered with reasonable effort by industry, and therefore highly efficient software versions can be achieved without excessive investment on the development cost. For example, the Cell Broadband Engine [11] (CBE) jointly designed by IBM, Sony and Toshiba was one of the most powerful and energy efficient architectures of its time, but it was proven notoriously difficult to program [12], reducing its industry adoption beyond the gaming sector.

Second, we needed to be able to experimentally evaluate various embedded GPUs to identify the most promising candidates for space use, primarily based on their performance and energy efficiency, as well as their software tools and libraries. This resulted in a selection of the most promising IP (intellectual property) GPU solution as well as the most promising COTS (commercial off-the-shelf) GPU solution. The former can be used for the development of a radiation hardened version of an embedded GPU IP, based on European technology, which is the leader in embedded GPU designs, to support Europe's non-dependence in the space domain, while the latter can be used in the shorter term to enable the fast adoption of GPUs in space.

Next, in order to perform a comparison between various GPUs and their GPU programming models for the space domain, we created an open source benchmarking suite for GPU on board-processing, named GPU4S Bench [13], which was used for the benchmarking of the selected GPU devices. In addition to this benchmarking suite, we developed several demonstrators of space algorithms ported on embedded GPUs [14] [15].

Finally, we define a roadmap regarding the adoption of GPUs in space. In this paper, we describe the main outcomes of the above performed activities in the GPU4S project and we present the lessons we have learnt throughout the duration of the project.

2. MAIN OUTCOMES AND LESSONS LEARNT

2.1. Space Software Survey

One of the first tasks of the project was the study of existing space software as well as the requirements of future missions. For this reason, we have conducted a theoretical analysis of algorithms found in several space domains, identifying potential candidates for GPU acceleration based on their processing characteristics. The outcome of the survey has been published in two parts, in [7] and in [16].

In particular, we observed that most of the existing space algorithms are a good fit for the GPU programming model, especially the ones used in on-board image processing. As these algorithms are mainly working on several pixels independently, there is abundant parallelism to be exploited by the massively parallel GPU hardware. This has been also confirmed at a later stage of the project, by our demonstrator of the Euclid NIR (Near Infrared) ESA software, which we ported on an embedded GPU [14].

On the other hand, one type of algorithms we initially identified as not a good candidate for GPU parallelisation was compression algorithms such as the ones found in the CCSDS standard. The reason for this was the low-parallelism exhibited in such operations, due to the serial operation of the algorithm and dependencies between parts of the data.

However, later in the project we implemented another set of demonstrators, porting the CCSDS 121 and CCSDS 122 space compression algorithms, which showed that even these algorithms could be accelerated on a GPU [15]. This was achieved by exploiting coarse grained parallelism, similar to the batching performed in neural network processing on GPUs.

Regarding the requirements of future missions, we observed a trend towards increasing the amount of acquired data from scientific instruments, as well as of the computational power required to support advanced functionalities such as autonomous navigation, which could be provided by GPUs.

Lesson 1: Modern GPUs can accelerate a wide range of existing and future on-board algorithms, even when their parallelism is not inherent.

Lesson 2: The only certain way to verify whether an algorithm can significantly benefit from a GPU, is to actually port it to a GPU.

2.2. Embedded GPU Hardware Survey

Another important project milestone was the study of existing embedded GPUs in order to select the most appropriate GPU IP of a European Embedded GPU vendor for a future radiation-hardened implementation by ESA or the most appropriate COTS embedded GPU for short term adoption. One of the criteria established by ESA for the selection was the availability of the IP in order to produce an FPGA prototype, while another one was the software ecosystem of each GPU.

We started by surveying embedded GPUs using their public information and by producing a classification of embedded GPUs and GPU-like architectures for space [7]. In particular, we identified low-end and high-end GPUs based on their ability to support general purpose programming languages or only graphics. Moreover, we defined additional classes such as COTS, soft IP products as well as high-level synthesis.

Meanwhile, we established links the licensing departments of embedded GPU vendors and suffered long delays until Non Disclosure Agreements (NDAs) were in place, as we explain in [16]. However eventually it became apparent that no commercial silicon IP vendor is willing to share any non-public details about their product, including their price, without an upfront commitment to buy their IP, even under NDA. This creates the counter-intuitive situation in which one IP customer has to select an IP vendor without being able to know their exact benefit or drawbacks compared to an equivalent product from a competitor. However, once the customer decides to do business with an IP provider, then they can get access to the full portfolio of that vendor, in order to select the most appropriate product offering to their needs, from that company.

Lesson 3: There is no way to select a priori the most appropriate commercial silicon IP from different vendors.

For this reason, the selection of proprietary IPs by semiconductor companies is performed with obscure criteria e.g. using consulting companies or personal decisions from high-level executives.

Another important finding from our experience is that the current licensing models of most IP vendors are focused around royalties arising from large volume markets such as consumer electronics or automotive, and therefore they are not prepared to work with low-volume markets like space, nor to customise their designs. There are also several challenges for the production of FPGA prototypes with commercial GPU IP. First, modern embedded GPU designs cannot fit on existing FPGAs, without producing a considerably reduced version, e.g. in terms of cache size, number of shader cores etc. Moreover, these reduced version can only fit on specific FPGA boards approaching the cost of $50K. Obviously, this cannot be achieved with the budget of a small exploratory project such as GPU4S (150K euros).

> **Lesson 4:** A project responsible to license or produce a demonstrator with commercial silicon or FPGA IP needs to have considerable budget available, specifically allocated for the cost of the IP vendor.

For this reason, it is not surprising that open source hardware is increasingly considered for low volume markets. For this reason, we extended our survey to open source GPU designs. As we reported in [16], there are several open source GPU or GPU-like designs.

The outcome of our survey however is not positive either. Most of the existing GPUs are incomplete, model obsolete GPUs with deprecated software stacks, lack documentation and support. Moreover, the ones which replicate commercial designs are potentially subject to patent infringements. However, the most important limitation of open source designs is their non-commercial friendly licenses such as GPL, which make impossible to use in a niche domain such as space, where several proprietary IPs are combined in an FPGA or ASIC.

> **Lesson 5:** Existing open-source GPU designs cannot be used for our purpose.

Fortunately, meanwhile the open source hardware community has grown and matured significantly with the proliferation of the RISC-V movement. These last couple of years several successful examples of open source processor IPs with liberal licenses have been included in commercial designs, which gives hope that in the future there may be similar designs in the GPU domain.

> **Lesson 6:** The RISC-V movement can create opportunities for a commercially-friendly open source GPU.

Apart from soft GPU IPs, we have also explored the possibility of High-Level Synthesis (HLS), which allows to program modern FPGAs in an easy way using OpenCL similar to GPUs, instead of hardware description languages. However, HLS is only available in the latest COTS FPGAs but not yet in space-grade FPGAs. Moreover, this solution does not offer the fast reconfiguration of GPU kernels, since the synthesis and reconfiguration times of HLS kernels are significantly longer, in the order of seconds or minutes. Last but not least, HLS kernels written in OpenCL require heavy annotations or modifications to achieve high performance, so that their code is considerably different that GPUs. However, HLS has a potential to facilitate FPGA development for space and it is worth to be explored in a separate ESA study.

> **Lesson 7:** HLS is not equivalent to a soft GPU.

Since the soft GPU survey did not culminate in a viable GPU product, we decided to consider COTS GPUs. In terms of performance, our analysis showed that NVIDIA's embedded products are the ones providing the highest theoretical performance, something we also confirmed at later stages with our benchmarking. Moreover, NVIDIA's proprietary programming language, CUDA, is the one with the largest developer base.

NVIDIA's embedded GPUs have been used in several rugged products and have been chosen for NASA Missions, however they suffer from short product lines with short market availability window. Therefore, component obsolescence in critical domains can be an issue. In addition, NVIDIA supplies their latest GPU SoCs only as part of modules, which limits the possibility for dedicated designs for space, both in terms of form factor and use of other COTS devices (e.g. power ICs) that need their own screening for the use in space.

Embedded AMD products have been also used in rugged environments, especially in aerospace. AMD supports the open standard OpenCL as well as a CUDA compatible language called HIP [17]. In addition, AMD provides more information regarding its architecture, and for this reason open source drivers as well as third party drivers such as the safety certified drivers from CoreAVI [18]. In terms of product lines offer longer availability for products used in critical domains (10 years).

A recent study on the radiation tolerance of COTS GPU SoCs from several vendors, revealed some concerns with certain NVIDIA GPUs which were not as apparent in AMD GPUs, due to being implemented with other foundry processes [19].

> **Lesson 8:** NVIDIA embedded products provide higher performance and software tooling, are only available as modules, which make space qualification more difficult.

> **Lesson 9:** AMD provides a more open approach in terms of module availability, open hardware/-software specification, open source and safety certified drivers, which make them more attractive for space.

From European vendors, ARM dominates the embedded GPU market, while Imagination Technologies was at the time of our survey the only one to offer an ASIL-B safety certified GPU in the Renesas R-CAR H3 platform and was designing new products with ASIL-D certification. However, ARM recently announced a GPU design compliant with ASIL-D. In terms for absolute performance, European GPUs are outperformed by NVIDIA and AMD products.

2.3. Embedded GPU Benchmarking

The next important milestone has been the performance and performance efficiency benchmarking of the selected platforms: NVIDIA Xavier, ARM G-72 found in the HiKey 970 and the Imagination PowerVR GX6650 found in Renesas's R-CAR H3.

However, the cancellation of the manufacturing of the R-CAR H3 line led us to replace it with the NVIDIA TX2. Moreover, in the meanwhile, an AMD Embedded Ryzen platform V1605B has been released. For this reason, we extended our analysis to this platform as well, by purchasing one of the first production units. Unfortunately, setting up a software environment with a working OpenCL GPU driver has been very challenging. We had faced similar issues with setting up the environment of the HiKey. On the other hand, setting up the NVIDIA platforms was seamless. The reason is that NVIDIA has full control over its platforms, while other GPU design firms depend on integration with third part IPs and face software fragmentation.

Lesson 10: NVIDIA's vertical integration results in tighter control of product releases in both hardware and software.

In order to benchmark the selected embedded GPUs, we identified the need for a relevant on-board benchmarking suite.

However, we noticed a considerable lack of standard benchmarking solutions for payload processing in space, especially regarding GPUs. To deal with this, we have created the GPU4S Bench [13], an open source benchmark suite of representative space algorithms across different space domains based on the space software survey we conducted in the first months of the project. In addition to GPU benchmarking, GPU4S Bench can be also used for the evaluation of GPU programming models for space payload processing. The reason why we designed GPU4S as Open Source with ESA's GPL-like license, is in order to be free of company IP rights or government export controls, which are dominating the space domain.

Lesson 11: Complex space application software is subject to restrictions.

Open source benchmarks maximise the potential of becoming the de-facto means of performance and energy efficiency comparison between embedded GPUs for space, as well as to allow reproducibility and crowd-sourcing results from new architectures. Being able to directly compare results from the same suite among different targets saves time and reduces costs, while it enables taking more straightforward decisions for the hardware of future space programs. Such a benefit has been already observed with the NPB benchmarks from NASA in supercomputing [20] as well as the NIR HAWAII-2RG BM algorithm [21], which has been used in several internal and ESA-funded activities for comparative analysis of numerous platforms. Such an algorithm is much more useful compared to an advanced and complete but proprietary processing space application e.g. [5] or [22], which cannot be reproduced in future studies performed by different contractors.

Details regarding the design principles and the implemented algorithms can be found in [13] while a summary of the coverage of different space domains can be seen in Table 1.

GPU4S Bench and CCSDS [15] implementations developed in GPU4S have been the basis of another open source benchmarking suite developed by ESA, OBPMark [34]. OBPMark provides complete space applications which can be implemented with the GPU4S Bench optimised algorithmic building blocks and aims on the benchmarking of any type of on-board computing platform. OBPMark replaces older ESA benchmarking suites and is the recommended way of benchmarking on-board devices. Both benchmark suites are closely related, share common code and benchmarking infrastructure. The source code of both suites is co-hosted and can be obtained from `http://OBPMark.org`. Moreover the website provides a result database and information about contributing and result reporting.

Lesson 12: Open source benchmarks, such as GPU4S Bench [13] and OBPMark [34], are required to circumvent space software restrictions and maximise benefit from public funding.

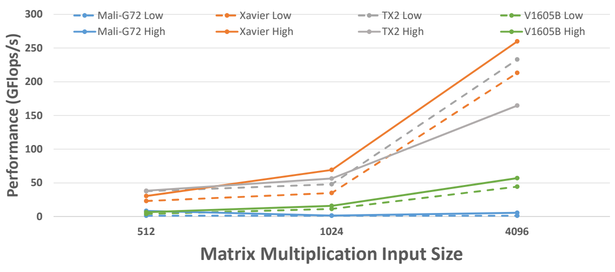

Figure 1. Performance of matrix multiplication on multiple platforms with different sizes.

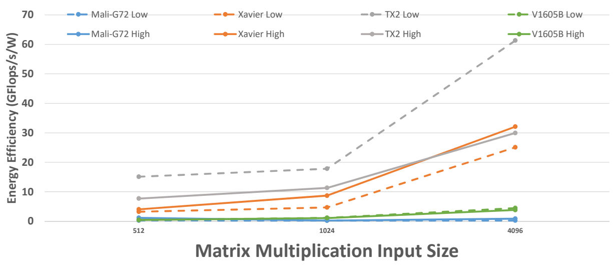

Figure 2. Performance efficiency of matrix multiplication on multiple platforms with different sizes.

Using the GPU4S benchmark suite allowed us to compare results of the different GPUs. All measurements

Table 1. Building blocks of GPU4S Bench [13] extracted from current and future payload applications across all space domains.

Domains / Building Blocks	Compression	Vision Based Navigation	Image Processing	Neural Network Processing	Signal Processing
Fast Fourier Transform			SAR [23], GENEVIS [24]		ADS-B [25], NGDSP [26]
Finite Impulse Response Filter		MER [27]			ADS-B [25], NGDSP [26]
Discrete Wavelet Transform	CCSDS 122 [28]				
Matrix Computation		MER [27], GENEVIS [24]	HTI [29]	Inference [30][31]	
Convolution		OpenCV	HTI [29], GENEVIS [24]	Inference [30][31]	
Correlation		OpenCV	GO3S [32], GENEVIS [24]		ADS-B
Max Detection and DNN Primitives		MER [27]	GO3S [32]	Inference [30][31]	ADS-B[25]
Synchronisation Mechanism		GENEVIS [24]	EUCLID NIR [21], GO3S [32]	TensorFlow	ADS-B [25], NGDSP [26]
Memory Allocation		CERES [33], OpenCV	EUCLID NIR [21], GO3S [32]	TensorFlow	ADS-B [25], NGDSP [26]

were obtained using the same power budget. Power measurements were also performed when possible to compute the energy efficiency.

The NVIDIA Xavier has outperformed all platforms in terms of performance as it is shown in Figure 1 for the matrix multiplication benchmark, while in terms of energy efficiency, the best choice has been either the NVIDIA Xavier or the TX2 as shown in Figure 2. Regarding the AMD V1605B platform we found out that the unofficial custom GPU driver we used for OpenCL support since AMD did not officially support OpenCL by the time of our experiments resulted in lower performance than expected. However the issue is being investigated and we believe that with proper software support, its performance will be higher.

In terms of maximum power consumption, our power measurements indicate that the GPU boards consume in total up to 15W, as indicated by their TDP, therefore confirming our initial selection that they can fit in the power budget of an on-board system.

Lesson 13: Embedded GPUs comply with on-board power requirements.

Apart from the benchmarking of GPUs, the development of the GPU4S Bench allowed us to evaluate also aspects related to the programming model of the GPUs as well as to their software ecosystem. First, the development, debugging and maintenance of CUDA and its equivalent open source version from AMD, HIP, are easier than OpenCL, since the latter is a lower-level API and therefore each CUDA/HIP statement corresponds to multiple OpenCL statements. In general, there is portability between the two languages, but not 100% guaranteed even on the same platform. For example, we encountered a corner case that prevented execution of the OpenCL since the implementations make an implicit assumption that the number of threads in a kernel have to be multiple of 32 although there is no such limitation stated in the standard.

Lesson 14: CUDA and HIP offer easier programmability than OpenCL.

In addition, since we developed hand-written, hand-optimised and vendor libraries versions of our benchmarks, we were able to assess whether it is possible to obtain high performance on GPUs with reasonable programming effort. In this aspect, we found two counter-intuitive situations. First, vendor optimised libraries are not always the fastest option. Most of these libraries are optimised for long, repetitive executions, so they exhibit a long initialisation cost that can exceed the actual processing cost of small amount of data or for small periods of time. This can be an issue in cases that the GPU applications vary in time during the mission, or simply because an application or the platform needs to restart due to a radiation fault. In these cases, hand written implementations are a more appropriate solution, even if they provide significantly lower performance, especially for small sizes. However, we noticed cases in which our implementations outperformed the vendor library such as in double precision floating point, probably because the library was not optimised until then for our target platforms, which are not frequently used for such calculations.

Lesson 15: Vendor optimised GPU libraries have a large initialisation cost, so they are not always the best choice, but they depend on the application scenario.

Lesson 16: It is possible to obtain high performance with reasonable GPU development effort.

Lesson 17: The only way to assess the obtained performance is through actual implementation.

Apart from individual algorithmic building blocks used in several space applications, we also ported space-relevant full applications to embedded GPUs. This includes an inference chain designed for the CIFAR-10 data set [13] which is part of the GPU4S Bench, the Euclid NIR ESA application [14][35] and implementations of the CCSDS 121 and 122 compression standards [15].

Our results have shown that in all cases, the NVIDIA GPUs are able to provide significant speedups, both compared to their CPUs as well as compared to existing space processors. Figure 3 shows the performance results obtained with the Euclid NIR application on the GPU. For detailed results please refer to [35]. The same applications on the AMD platform provide lower GPU performance, but strong parallel CPU performance compared to the parallel CPU performance of the NVIDIA plat-

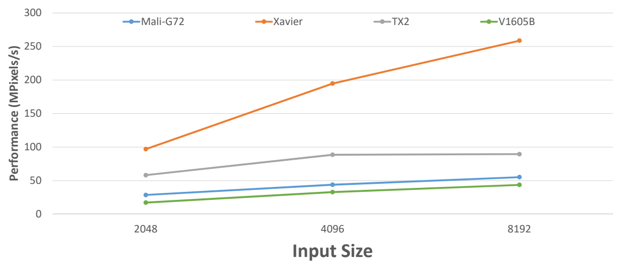

Figure 3. Performance of Euclid NIR on multiple platforms with different sizes.

forms. Again, the main reason for the low AMD GPU performance is identified in the unofficial custom driver we used. Once AMD offers official support for OpenCL on V1605B, we expect to see higher performance. Overall however, the performance of the NVIDIA Xavier's GPU outperformed all other versions and platforms.

GPU4S Bench has been also ported to OpenMP and was used to compare the performance capabilities of multi-core CPUs and GPUs in the candidate platforms of the H2020 UP2DATE project [36], which have been similar to the platforms considered in GPU4S: Xilinx Zynq Ultrascale+, NVIDIA Xavier, NVIDIA TX2. Again, the benchmarking results confirmed that the NVIDIA Xavier's GPU outperformed all other CPUs and GPUs.

Lesson 18: GPUs can significantly accelerate complex space processing compared to other technologies.

2.4. Radiation Effects

As already mentioned, a synthesis of radiation testing results of embedded GPUs are available [19], with indication that certain AMD devices have more promising radiation tolerance than some recent NVIDIA devices, mainly due to effects related to the used process node technologies from competing foundries. Both types of GPUs have been considered for experimental launches of national agencies and nanosatellites. These missions are going to provide valuable information regarding the radiation tolerance of these devices in the space environment. However, as GPU SoCs are complex devices to test for radiation effects - and currently publicly available radiation test results do not cover all needed parameters for future space qualification - more radiation tests are essential for their general use in space. ESA is currently planning further radiation tests of embedded GPUs.

Meanwhile, our analysis from the use of COTS devices in space shows that solutions based on redundancy can mitigate radiation effects. However, hardware triplication as implemented in some space-grade systems although it is effective, it will probably exceed significantly the power budget of an on-board space system. For this reason, other solutions leveraging the inherent parallelism of the GPU hardware and software can achieve similar degree of reliability. For example, hardware [37] or software [38] GPU reliability solutions proposed for the automotive sector can be reused in space as well. Software solutions [39] are particularly interesting, since they can be applied directly in existing systems. However, all these potential proposals need to be assessed in a relevant space environment as part of a radiation testing campaign and/or experimental missions.

Lesson 19: GPU reliability solutions for the automotive domain can be adopted for use in space.

3. ROADMAP AND WAY FORWARD

Based on the project's outcomes and the lessons learnt that we have presented, we conclude that Embedded GPUs are an appropriate solution for high-performance processing in space.

However, in order to be adopted we need to solve some of their identified issues. First, it seems that the only potential option at least at the moment is to adopt them as COTS components. Therefore, we need to address the up-screening of these devices, particularly in terms of radiation effects, as well as their long term availability. COTS devices are becoming more accepted for the use in space, mainly driven by adoption in nano-satellites. ESA is also working on specific guidelines for the use of COTS devices in space.

Moreover, the available software fault detection, isolation, and recovery (FDIR) techniques need to mature and be tested in relevant environments. Such activities are already planned by national agencies and can be complemented by flight heritage results from commercial and nanosatellite missions using GPUs.

Regarding the domains of COTS GPU adoption in space, institutional missions for example are rarely based on such technologies in order to reduce risk. However, if they are proven as an enabling technology by the use in other type of missions, they may be considered for wider adoption.

Earth Observation commercial missions are more likely to adopt GPUs since their typical processing tasks can be easily parallelised. Moreover, the recent experimental use of AI processors in space such as the Intel Movidius in Φsat-1 for cloud removal in Earth Observation shows that this domain is open to COTS hardware and software. With the increased adoption of AI techniques in space, GPUs have the potential to be used. The reason is that GPUs are good accelerators for AI taks as we have shown also with our CIFAR-10 demonstrator. Although AI accelerators are more efficient for this type of processing, layer operations of AI algorithms are constantly changing, so the flexibility of GPUs provides a long term solution for AI processing as well as for other image processing too.

Nano-satellites already heavily rely on COTS components and can directly benefit from embedded GPUs, and there are several missions which already adopt embedded GPUs. The main benefit in that case is the reduction of cost both in terms of hardware as well as of software, which can be developed fast and there is large availability of GPU software developers. This applies also to constellation missions, where cost is an important driver, too, although at a smaller extent.

The mission duration and orbit are also important for the possible adoption of GPUs. Low-earth orbit missions (LEO) and missions with short duration will likely adopt COTS GPUs first, due to the more limited exposure to radiation.

Finally, another domain that can benefit from GPUs is New Space. Several commercial satellite operators envision the use of satellite-as-a-service in which they rent time of their platform to customers. Since the processing requirements of each customer are different and are not known a priori, the ability of the fast reprogramming offered by the GPU can be an important feature for these missions.

ACKNOWLEDGMENTS

This work is funded by ESA under the GPU4S (GPU for Space) project (ITT AO/1-9010/17/NL/AF). It is also partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under grants PID2019-107255GB and FJCI-2017-34095 and HiPEAC, as well as the European Commission's Horizon 2020 programme under the UP2DATE project (grant agreement 871465).

REFERENCES

[1] Kevin McManamon, Richard Lancaster, and Nuno Silva. ExoMars Rover Vehicle Perception System Architecture and Test Results. *Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2017.

[2] M Mammarella, CA Paissoni, N Viola, A Denaro, E Gargioli, and Massobrio. The Lunar Space Tug: A sustainable bridge between low Earth orbits and the Cislunar Habitat. *Acta Astronautica*, 138:102 – 117, 2017.

[3] Satomi Kawamoto, Yasushi Ohkawa, Hiroyuki Okamoto, Kentaroh Iki, Teppei Okumura, Yasuhiro Katayama, Masato Hayashi, Yuta Horikawa, Hiroki Kato, and Naomi Murakami. Current Status of Research and Development on Active Debris Removal at JAXA. *7th European Conference on Space Debris (SDC7)*, 2017.

[4] Bob Balaram, Timothy Canham, Courtney Duncan, Håvard F Grip, Wayne Johnson, Justin Maki, Amelia Quon, Ryan Stern, and David Zhu. Mars Helicopter Technology Demonstrator. In *AIAA Atmospheric Flight Mechanics Conference*, 2018.

[5] Powell, Wesley and Campola, Michael and Sheets, Teresa and Davidson, Abigail and Welsh, Sebastian. Commercial Off-The-Shelf GPU Qualification for Space Applications. Technical report, NASA, 2018.

[6] Edward Wyrwas. Body of Knowledge for Graphics Processing Units (GPUs). Technical report, NASA, 2018.

[7] Leonidas Kosmidis, Jérôme Lachaize, Jaume Abella, Olivier Notebaert, Francisco J Cazorla, and David Steenari. GPU4S: Embedded GPUs for Space. In *Digital System Design (DSD) Euromicro Conference*, 2019.

[8] Nan Li, Aimin Xiao, Mengxi Yu, Jianquan Zhang, and Wenbo Dong. Application of GPU On-orbit and Self-adaptive Scheduling by its Internal Thermal Sensor. In *International Astronautical Congress (IAC)*, 2018.

[9] Kevin Peterson. Fly Me to the Moon: The Role of GPUs in Lunar Exploration. *GPU Technology Conference*, 2015.

[10] Daniele Luchena, Vincenzo Schiattarella, Dario Spiller, Marco Moriani, and Fabio Curti. A New Complementary Multi-Core Data Processor for Space Applications. In *International Astronautical Congress (IAC)*, 2018.

[11] M. Gschwind, H. P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki. Synergistic Processing in Cell's Multicore Architecture. *IEEE Micro*, 26(2):10–24, March 2006.

[12] Abraham Arevalo, Ricardo M. Matinata, Maharaja Pandian, Eitan Peri, Kurtis Ruby, Francois Thomas, and Chris Almond. *Programming the Cell Broadband Engine Architecture: Examples and Best Practices*. IBM Red Books, 2008.

[13] I. Rodriguez et al. GPU4S Bench: Design and Implementation of an Open GPU Benchmarking Suite for Space On-board Processing. Technical Report UPC-DAC-RR-CAP-2019-1, Universitat Politecnica de Catalunya. `https://www.ac.upc.edu/app/research-reports/public/html/research_center_index-CAP-2019,en.html`.

[14] Iván Rodriguez, Leonidas Kosmidis, Olivier Notebaert, Francisco J Cazorla, and David Steenari. An On-board Algorithm Implementation on an Embedded GPU: A Space Case Study. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1718–1719, 2020.

[15] Iván Rodriguez, Alvaro Jover, Leonidas Kosmidis, and David Steenari. On the Embedded GPU Parallelisation of On-Board CCSDS Compressors: a Benchmarking Approach. *International Workshop on On-Board Payload Data Compression (OBPDC)*, Sep 2020. https://atpi.eventsair.com/QuickEventWebsitePortal/obpdc-2020/eventwebsite/Agenda/AgendaItemDetail?id=e9be1307-8762-43b8-985e-46fdb5f48548.

[16] L. Kosmidis, I. Rodriguez, A. Jover, S. Alcaide, J. Lachaize, J. Abella, O. Notebaert, F. J. Cazorla, and D. Steenari. GPU4S: Embedded GPUs in Space - Latest Project Updates. *Elsevier Microprocessors and Microsystems*, 77, Sept 2020.

[17] AMD. ROCm Developers tools HIP. 2019. https://github.com/ROCm-Developer-Tools/HIP.

[18] Marc Benito, Matina Maria Trompouki, Leonidas Kosmidis, Juan David Garcia, Sergio Carretero, and Ken Wenger. Comparison of GPU Computing Methodologies for Safety-Critical Systems: An Avionics Case Study. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[19] Ian Troxel, J. Schaefer, Matt Gruber, and Patrick Gauvin. Summary of Radiation Test Data for Several GPUs. In *2019 Radiation Hardened Electronics Technology (RHET)*, 2014.

[20] Yan-Tyng Chang, Robert T Hood, Haoqiang Jin, Steve W Heistand, Samson H Cheung, Mohammad J Djomehri, Gabriele Jost, and Daniel S Kokron. Evaluating the Suitability of Commercial Clouds for NASA's High Performance Computing Applications: A Trade Study. Technical Report NAS-2018-01, NASA, 2018.

[21] Andreas Jung and Pierre-Elie Crouzet. The H2RG Infrared Detector: Introduction and Results of Data Processing on Different Platforms. Presentation, European Space Agency (ESA), 2012. http://www.esa.int/Our_Activities/Space_Engineering _Technology/ Onboard_Data_Processing/General_Benchmarking _and_Specific_Algorithms.

[22] Daniel Hellström and Fabrice Cros. RTEMS SMP Final Report: Development Environment for Future Leon Multi-core. Technical Report RTEMSSMP-FR-00, Final Report, ESA-ESTEC, 2015.

[23] Clay Stewart, V Larson, VK Madisetti, and DB Wollians. Synthetic Aperture Radar Algorithms. *The Digital Signal Processing Handbook*, 2000.

[24] Roland Brochard, Jérémy Lebreton, Cyril Robin, Keyvan Kanani, Grégory Jonniaux, Aurore Masson, Noela Despré, and Ahmad Berjaoui. Scientific Image Rendering for Space Scenes with the SurRender Software. In *International Astronautical Congress*, 2018.

[25] M. Strohmeier, M. Schfer, V. Lenders, and I. Martinovic. Realities and Challenges of Nextgen Air Traffic Management: The Case of ADS-B. *IEEE Communications Magazine*, 52(5):111–118, 2014.

[26] ESA. On-Board Data Processing - Benchmarks, 2012. `https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Onboard_Data_Processing/General_Benchmarking_and_Specific_Algorithms`.

[27] Steven B Goldberg, Mark W Maimone, and Larry Matthies. Stereo Vision and Rover Navigation Software for Planetary Exploration. In *IEEE Aerospace Conference*, 2002.

[28] The Consultative Committee for Space Data Systems. *Image Data Compression Recommended Standard CCSDS 122.0-B-2*, 2017.

[29] Andrew C Pineda, Jesse K Mee, Phillip M Cunio, and Reed A Weber. Benchmarking Image Processing for Space: Introducing the SPACER architecture laboratory. In *IEEE Aerospace Conference*, 2016.

[30] Vipul Mani, Amogh Kulkarni, Mahish Guru, Raghav Pathak, and Shashank Pathak. Exploration of Mars through an Autonomous and Machine Learning enabled Constellation of Drones. In *International Astronautical Congress*, 2018.

[31] Ewan Reid and Michele Faragalli and Kaizad Raimalwala and Evan Smal. Machine Learning Applications for Safe and Efficient Rover Mobility Operations and Planning. In *International Astronautical Congress (AIC)*, 2018.

[32] Eric Maliet, Anthony Villien, Gregory Pedersen, and Philippe Charvet. Geostationary Observation Space Surveillance System (GO3S) Real Time Video From Space. In *65th International Astronautical Congress (IAC)*, 2014.

[33] CNES. CERES: Three satellites to boost Frances intelligence capabilities, 2019. `https://ceres.cnes.fr/en/ceres-2`.

[34] David Steenari, Leonidas Kosmidis, Ivan Rodriquez, Alvaro Jover, and Kyra Förster. OBPMark (On-Board Processing Benchmarks) - Open Source Computational Performance Benchmarks for Space Applications. In *European Workshop on On-Board Data Processing (OBDP)*, 2021.

[35] Iván Rodriguez. An On-board Algorithm Implementation on an Embedded GPU: A Space Case Study. Master's thesis, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, February 2021. `https://upcommons.upc.edu/handle/2117/344892`.

[36] Alvaro Jover-Alvarez, Alejandro J. Calderon, Ivan Rodriguez, Kosmidis Leonidas, Kazi Asifuzzaman, Patrick Uven, Kim Grttner, Tomaso Poggi, and Irune Agirre. The UP2DATE Baseline Research Platforms. In *Proceedings of the Design, Automation & Test in Europe (DATE)*, 02 2021.

[37] S. Alcaide, L. Kosmidis, C. Hernandez, and J. Abella. High-Integrity GPU Designs for Critical Real-Time Automotive Systems. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019.

[38] S. Alcaide, L. Kosmidis, C. Hernandez, and J. Abella. Software-only Diverse Redundancy on GPUs for Autonomous Driving Platforms. In *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 90–96, 2019.

[39] Fredrik C Bruhn, Nandinbaatar Tsog, Fabian Kunkel, Oskar Flordal, and Ian Troxel. Enabling Radiation Tolerant Heterogeneous GPU-based Onboard Data Processing in Space. *CEAS Space Journal*, pages 1–14, 2020.