SATELLITE INSTRUMENT CONTROL UNIT WITH ARTIFICIAL INTELLIGENCE ENGINE ON A SINGLE CHIP: ICU4SAT

Gianluca Giuffrida[1], Pietro Nannipieri[1], Lorenzo Diana[1], Silvia Panicacci[1], Luca Fanucci[1], Gionata Benelli[2], Giuseppe Gentile[2], Marcelo Brandalero[3], and Michael Hübner[3]

[1]*Dept. Information engineering, University of Pisa, 56122, Pisa, Italy*
[2]*Ingeniars S.r.l, 56121, Pisa, Italy*
[3]*Brandenburg University of Technology, 03046, Cottbus, Germany*

ABSTRACT

Artificial Intelligence (AI) is spreading into every field of application, achieving, in some cases, better performance than ordinary algorithms. One of the most successful areas in which AI has introduced innovation is computer vision, outperforming the older predecessor algorithms. However, the computational capacity required by those algorithms is still a major limitation to the adoption of these techniques, especially in extreme cutting-edge applications such as space. The ICU4SAT embedded system presented in this paper aims at changing the rules of image/data management and processing on board satellites, overcoming the drawbacks of the current limitations both at technological and industrial level. The ICU4SAT system leverages on three highly innovative, open-source components, which are integrated into a modular and flexible FPGA. The core of the system is a soft-programmable GPU-like hardware, called soft-GPU, which can analyse and process the data acquired directly on-board, and programming it via high-level artificial intelligence and computer vision frameworks. In addition, the soft-GPU can be fully re-configured and re-targeted according to missions or monitoring needs, even temporarily, fitting perfectly with mission objectives. At a broader industry level, the commercial availability of such a solution would translate into more room for Small Medium Enterprises for leading independently space missions and projects, expanding business opportunities in the space domain. Finally, the innovation provided by the ICU4SAT can be a key element in the Satellite as a Service paradigms, where the Artificial Intelligence and smart applications play a key role.

Key words: Earth observation; Artificial Intelligence; small satellite; FPGA; radiation hardened.

1. INTRODUCTION

In recent years, interest in space missions and their possible applications has been growing exponentially in the wake of the new space economy. This represents a profound transformation of the space field, which aims to be integrated into various industrial and economic sectors such as transport, aviation, and meteorology. In particular, the use of small- micro-satellites has brought a new vision in the Earth-Observation (EO) missions: SATellite as a Service (SATaaS). This paradigm sees satellites as reusable resources, not tied to a purpose, but as reconfigurable devices, that simplify access to space even for medium and small companies that cannot afford to build their own satellites. To make the most out of this paradigm, new methodologies for implementing and exploiting satellites are needed, allowing fully customisability by the end user even after the mission has started. AI algorithms represent the most viable solution to provide great performance, easy customisation, and minor time-to-market for the new generation mission payloads. In fact, they have already shown their adaptability in several applications for different purposes, such as image segmentation, signal analysis, classification, and regression. On the other hand, these algorithms require high performance hardware devices due to the quantity of operations involved. To overcome the low computational capacity, dedicated COTS hardware accelerators have been deployed, obtaining a good trade-off between inference speed and power consumption, but they are not designed to be flexible or used in space environment. Indeed, there is a lack of standardised and efficient general-purpose solutions to face this issue and industries are still in most cases designing and developing custom ad-hoc Instrument Control & Data Processing Units (ICU-DPU) which can be usually employed only for the single mission they have been designed for. ICU4SAT is the game changer, providing reconfigurability and low power consumption embedded in an FPGA technology. It provides high performance through customised hardware design implementation, and fast deployment time thanks to its dedicated software compiler. In addition, by exploiting radiation hardener FPGA, ICU4SAT is robust to the harsh space environment. As follow, a briefly introduction on HW accelerators and AI workflow is described in Sec.2. The detailed information regarding the ICU4SAT and the AI software workflow generally used is described in Sec.3. Finally, results and conclusion are reported respectively on Sec.4 and Sec.5.

2. AI AND HW ACCELLERATORS

The high number of operations involved in AI algorithms has limited the spread of these techniques in sectors where the power budget matters. Historically, AI models were trained and run only on GPU-farms or datacentres, principally focusing on the performance at the cost of poor usability and portability on the embedded world. On the other hand, lite version of state-of-the-art AI algorithms have achieved great performance, although lower than the original ones. An example is YOLOv3 [1], whose lite version is called TinyYOLOv3 [2] and is dedicated to embedded devices with limited memory storage and computational power.

2.1. HW accelerators

The COTS accelerators are easily classifiable by their processors[3, 4] as: VPU, TPU, FPGA, and the most known GPU. The first two processors have the best performance in terms of power per inference since they have been devised to speed up inferences. Instead, FPGAs and GPUs are more general purposes and they are the most powerful in term of computational capabilities.

A **VPU**: VPUs represent a new class of processors able to increase the speed of visual processing as CNN, SIFT [5], Sobel and similar. The most promising accelerators in this category are the Intel Movidius Myriad VPU currently available in two versions: the Myriad 2 [6] and the Myriad-X. The *core* of both processors is the computational engine that uses groups of specialised vectors of Very Long Instruction Word (VLIW) processors called SHAVEs capable of consuming only a few watts (W) [7]. Myriad 2 has 12 SHAVEs while the Myriad-X has 18 SHAVEs. They show better performance, when accelerating CNNs model or other supported layers, than mobile CPUs or general-purpose low-power processors.
To reduce the computational effort, all the registers and operation within the processor use 16 bits floating point arithmetic. Moreover, the Myriad 2 processor has already passed the preliminary *radiation tests* at CERN [8].

B **TPU**: TPU is an innovative hardware accelerator dedicated to a particular data structure: *Tensors* [9]. Tensors are a base type of the TensorFlow framework [10] developed by Google. The standard structures and the dedicate libraries for GPU and VPU make tensors and consequently TensorFlow very powerful tools in the Machine Learning (ML) world. The Coral Edge TPU is an example of an edge hardware accelerator whose performances are very promising, especially in the static images processing acceleration e.g., CNN, FCN. The best performances of this hardware platform are reached exploiting TensorFlow Lite and 8 bits integer quantization, even if the latter could have a big impact on the model metrics.

C **FPGA**: FPGAs are extremely flexible hardware solutions, which could be completely customised. This customizability, however, represents the bottleneck for a fast deployment [11].
In fact, the use of an FPGA requires many additional design steps compared to COTS ASIC, including the design of the architecture of the hardware accelerator and the quantization of the model, for approaches exploiting fixed-point representation.
FPGAs are produced by numerous companies such as Xilinx, MicroSemi, and Intel. Some FPGAs, like RTG4 or Brave, are also radiation-hard/tolerant; which means these boards can tolerate the radiations suffered during the life of the mission as explained in [12, 13].

D **GPU**: GPUs [14] are the most widely used to carry out both inference and training process of the typical ML models. Their computational power is entrusted to the parallel structure of the hardware that computes operations among matrices at a very high frequency. Nvidia and AMD lead the market of the GPU for ML training using respectively *CUDA Core* (Nvidia) and *ROCm* (AMD), as shown in [3, 4]. Moreover, several frameworks allow to use the potentiality offered by GPUs, including TensorFlow, TensorFlow Lite, and PyTorch. This hardware can train, infer, quantize the model and run inferences supporting a wide range of computational accuracies e.g., 32 and 16 bits floating point, 16, 8, 4, and 2 bits integer. On the other hand, these solutions consume huge amount of power, reaching a peak of 200 W and therefore cannot be used for on the edge applications.

2.2. Artificial Intelligence workflow

The greatest limitation of these accelerators is the software developing approach. In fact, the actual COTS hardware accelerators use dedicated tools to bring the algorithm on the hardware. Generally, to port a state-of-the-art algorithm in an embedded hardware device or FPGA requires the following three steps:

1. Formally description of the network using a NN framework (TensorFlow, Pyorch, etc.): The network is described layer by layer using a dedicated framework and trained exploiting very high speed hardware such as GPU. The generated weights are then stored in 32/64 bits floating point vector files.
2. Quantize the weights of the network defined at step 1. The 32/64 bits floating point arithmetic does not represent an optimal solution to be integrated within target hardware due to the limited number of resources available. To reduce the number of bits used to represent each weight, three main types of *quantization* processes are available in literature:

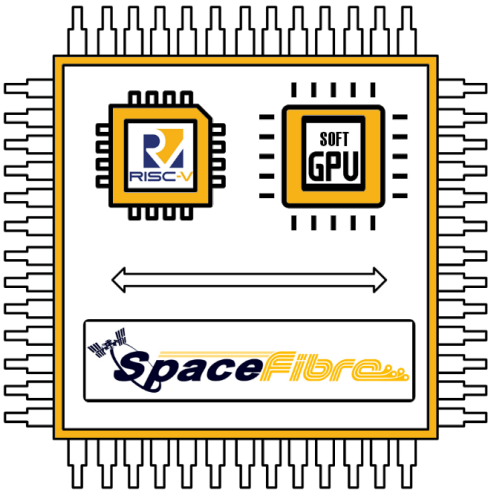

Figure 1. High level soft-GPU system architecture.

a. Quantization aware training which introduces the internal constraints of the target device as part of the network flow in the training process.

b. Post-training quantization which quantizes the weights of the network trading-off the minimum number of integer bits per layer and the model accuracy.

c. Post-training statistical quantization which exploits the same procedure of step 2.b but it also considers the statistical properties of the dataset.

3. Port the network trained at step 2 into the target device.

This procedure could lead to possible failures due to the non compatibility with the target hardware e.g., not enough memory, higher inference time, or drastically reduction of the accuracy of the algorithms.

3. ICU4SAT

ICU4SAT aims to change the drawbacks of the actual systems by redesigning both hardware and software. It is based on three highly innovative components: RISC-V, soft-GPU, and SpaceFibre/SpaceWire, as shown in Fig. 1. In literature, there are several attempts to develop a GPU-like system based on FPGA technologies [15, 16, 17], but they lack of dedicated frameworks and the incompatibility with the high level programming languages (e.g., TensorFlow, PyTorch, Caffe), reducing the adoption from the programmer community. To overcome this main limitation, and avoid compatibility obsolescence we decided to exploit LLVM compiler toolkit in conjunction with the XLA tool provided by TensorFlow which provide an interface to the LLVM compiler, as detailed in Sec.3.2.

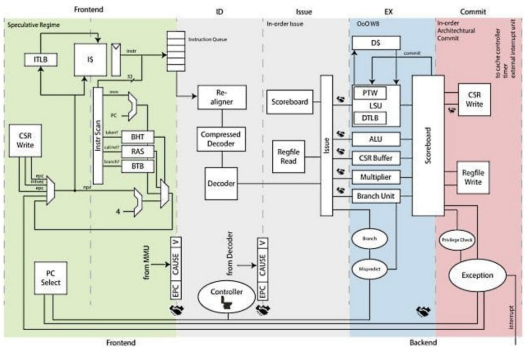

Figure 2. Detailed architecture of the RISC-V

3.1. Hardware architecture

The HW structure of the ICU4SAT is composed by three IP cores. For each component, a simple description is proposed as follows.

RISC-V

The RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration, experiencing rapid uptake in both industry and academia. It is based on the fifth generation of RISC design from UC Berkeley and its schematic is shown in Fig.2. The ICU4SAT system mounts the CVA6 (Ariane) RISC-V processor that has the following characteristics:

- 64-bit RISC-V ISA, Open Source;
- 6-stage, single issue, in-order CPU;
- It fully implements I, M, A and C extensions;
- Three privilege levels M, S, U to fully support a Unix-like operating system (Linux capable).

soft-GPU

soft-GPU, shown in Fig.3, is a soft GPU-like architecture for FPGAs.It is programmed using OpenCL and TensorFlow kernels, and represents the most innovative component of the ICU4SAT system. Its configuration can be customized according to application needs.

- Portable, scalable and flexible. Dynamically HW Reconfigurable;
- Multiple-Thread (SIMT) processor preliminary described in VHDL;
- Power savings between 3.2x and 4.5x, with respect to NEON ARM extension;
- Speedups between 10.6x and 48.5x, with respect to NEON ARM extension;
- Area overhead between 3.0x and 17.7x.

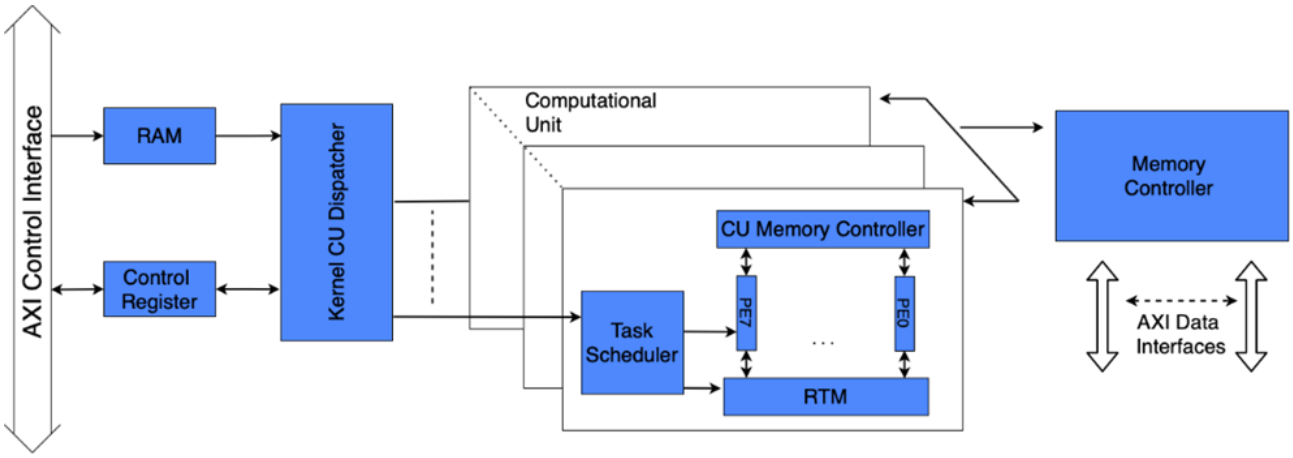

Figure 3. Detailed architecture of the soft-GPU accelerator

To reflect as much as possible a real GPU architecture, the soft-GPU implements a predefined number of Computation Units (CUs). Each CUs contains 8 Processing Elements, whose are responsible to execute the general purpose, and matrix operations. Actually, it is possible to synthesised up to 16 CUs, and exploit different arithmetic precision within the PEs, i.e., 32, and 16 bits floating point, 16, 8, and 4 bits integer. The number of CUs synthesised on the FPGA effectively determines the power consumption. In fact, the number of CUs can be modified directly during the mission, adapting the computational needs to the power budget available.

Communication module

SpaceFibre, described in Fig.4, is an open protocol developed under the ESA supervision, which standardisation process ended in May 2019. It can reach very high-speed throughput i.e., up to 100 Gbps directly on-board satellites. SpaceFibre characteristics are:

- Meant to be used for Very high speed (up to 100Gbps) satellite On-Board Data-handling;

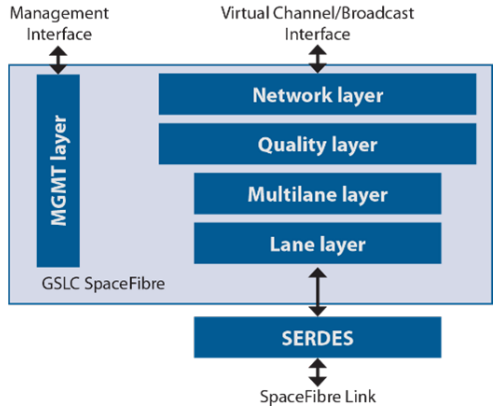

Figure 4. High level architecture of SpaceFibre.

- Full HDL implementation, with AXI Slave I/F;
- Built-in Quality of Service, Fault Detection Isolation & Recovery;
- First In-Orbit demonstration mission is about to be launched;
- Overcome limitation (Bandwidth, Flexibility, etc.) of State-of-the-art solution.

Since the hardware system is composed by three independent IP cores the first goal was to unify and integrate them in a single IP solution. To reduce the integration time and simplify the routing of the components, each of them was implemented with an AXI bus as part of its description. The brain of the system SoC is the RISC-V general purpose processor. It is responsible to schedule the processes for soft-GPU, verify the internal functionalities of the SoC, and, control the memory bus shared among the components. The soft-GPU, instead, is responsible to accelerate the computer vision or artificial intelligence inference tasks. It is started, stopped and reconfigured through the RISC-V processor, when needed, and it is programmed through OpenCL and TensorFlow high level languages. The communication module acts as input/output interface with other modules of the satellite. Furthermore, the three components, above mentioned, have been integrated in a space-qualified FPGA, with the big advantage of being suitable for any kind of missions.

In particular, entire system is able to:

1. Execute computer vision, compatible with OpenCL standard, and artificial intelligence tasks;
2. Start or stop the inference process at any time;
3. Execute internal functionalities check;
4. Input/output data from a SpaceFibre/SpaceWire module;

5. Soft-Reconfigure the soft-GPU IP via the RISC-V processor.

The final result is a general-purpose Instrument Control & Data Processing System on Chip which can be totally programmed exploiting the powerful LLVM compiler infrastructure built appositely on top of it. The latter represents an innovative yet essential layer which enables the programmability of the ICU4SAT through general-purpose high-level programming languages, e.g., Python, TensorFlow, PyTorch. This will open the doors for intelligent applications to end users with expertise in Artificial Intelligence but not familiar with hardware platforms.

3.2. Software architecture

The software part of ICU4SAT is composed by a compiler, dedicated to the generation of the soft-GPU executable kernels, and a development tool, offered to end-users for deploying new applications with minimum effort.
Considering the compiler part, we decided to exploit LLVM project, which is a collection of modular and reusable compiler and toolchain technologies, adding our dedicated target hardware. More in detail, LLVM is composed by three independent and complementary components: the FrontEnd (FE), the Optimiser (OP) and the BackEnd (BE). FE is responsible to parse the high-level programming language (e.g., OpenCL, C++, Python) in an Intermediate Representation (IR), a pseudo assembly language that abstracts the generated code from the hardware target. The IR code is then fed by the OP, which removes unused/unnecessary functions/operations and generate an optimised version of IR, the so-called optimised IR. Finally, the latter is given as input to the BE, which targets the optimised IR to the selected hardware, generating the runnable code.
Since the soft-GPU is a new target hardware (obviously it is not included in the official LLVM target BEs), we have added a new BE to LLVM, independently from the FE, in order to translate the IR code in the soft-GPU executable file. Of course, BE is strictly related to the specific hardware specifications, in order to generate the most optimised code for the target soft-GPU. Therefore, it has been co-designed and developed following the hardware flow.
Moreover, the LLVM compiler allows to i) compile OpenCL kernels to be run on the soft-GPU, ii) compile TensorFlow compiled (XLA) kernels to be run on the soft-GPU, iii) alert the user of incompatibility between the kernels and the soft-GPU HW, and iv) generate soft-GPU assembly files for debugging purposes.

Finally, to provide a good experience to the end users, a simple preliminary development tool to generate the soft-GPU scheduling and dispatching for the applications has been developed and implemented. The tool can generate an empty application skeleton, providing the task scheduling for the RISC-V and soft-GPU (i.e., the tasks for the RISC-V and the tasks to be executed on the soft-GPU). This allows to directly use OpenCL and TensorFlow (not completely for the moment) to create neural networks to be ran directly on the ICU4SAT system, hiding the complete process of compiling, optimising, generating executable code and scheduling tasks for the target hardware.

3.3. Innovation of ICU4SAT system

The ICU4SAT ecosystem has three key innovative elements: I) flexibility, II) modularity, III) customisability, and IV) on-board design.

Flexibility
The entire HW structure of ICU4SAT is described in HDL, allowing its customisability at any level. This feature allows the HW developer to customise the HDL in case some operations should be optimised. Additionally, the LLVM BE, which is written in C++, can be easily updated with the new instructions, or a new sub-target HW BE description can be implemented. This high flexibility solution makes ICU4SAT very interesting both for those who want to use it as it is and for those who are interested in strongly customising it.

Modularity
Although it seems to be a common custom, using IP cores allow to partially re-configure the soft-GPU, increasing the modularity of the system, i.e., selecting only the components that the user needs. The system becomes then independent from the FPGA platform. Furthermore, in case the users will not need SpaceFibre/SpaceWire as communication module, they shall remove them and/or include their IP communication module. Nevertheless, the possibility of commercializing the single IP core allows interesting collaboration with big space system integrators, which can be interested only in the soft-GPU accelerator.

Customisability
At the contrary of the dedicated HW accelerators developed on top of FPGA, the soft-GPU has its own developer workflow. In the latter years, high-level frameworks for AI applications have simplified the developing of these algorithms, making the developers life very simple. Following the same example, we have provided a basic preliminary framework, which will be improved to be compatible with the standard frameworks already available (e.g., TensorFlow, PyTorch, and Caffe), able to port and run computer vision and AI algorithms directly on the ICU4SAT system.

On-board design
The new satellites era seems to be characterised by a strong use of AI algorithms for the post-processing of data on ground. This process requires to download the data acquired by on-board sensors before processing it, even when it is not relevant for the mission or not usable, e.g., covered by cloud, corrupted, distorted, etc. Thanks to the soft-GPU of ICU4SAT, part of the processing can be shifted from ground to edge, relaxing bandwidth, and

storage pre-requisites. Moreover, the soft-GPU can execute computer vision kernels, leading to improvements in the standard algorithms run on-board satellite, i.e., optical flow, re-binning etc.

4. RESULT

Preliminary performance results have been evaluated by comparing the ICU4SAT against ARM with the NEON vector engine and cache enabled, and a MicroBlaze configured for maximum performance. The tests of the ICU4SAT system have been carried out implementing its soft core on Xilinx ZC706 FPGA using 250MHz of time constraint, while ARM core ran at 667 MHz and the MicroBlaze at 185 MHz. The soft-GPU runs OpenCL kernels compiled via LLVM ad-hoc compiler, while the two processors run dedicated equivalent implementations of those kernels compiled to achieve maximum performance.

Fig. 5 shows the soft-GPU speedup over the ARM processor when 8 CUs are synthesised. For each kernel, the maximum, minimum, and the average speedup for different problem sizes ranging from 256 to $256K$ are reported. It is worth to note that for some kernels a minimum problem size is required to achieve a positive speedup for the soft-GPU.

Fig. 6 shows the speedup of the soft-GPU over the MicroBlaze according to the number of CUs synthesised. The reported speedup is the average speedup over different problem sizes ranging from 256 to $256K$, where higher the number of CUs, the better.

Furthermore, thanks to the efficient use of the cache memory, the soft-GPU achieves considerable speedup also for operations that are less computationally intensive. The achieved speedup improves as the ratio of the number of performed arithmetic operations to the required number of memory accesses increases.

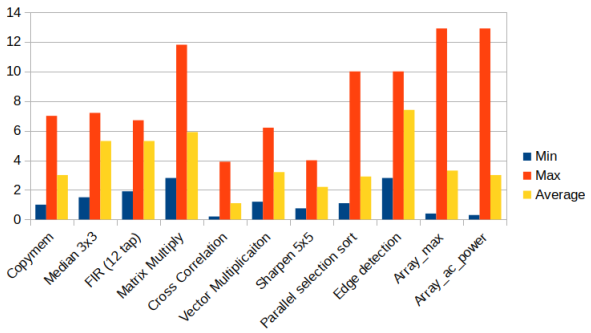

Figure 5. Speedup of soft-GPU using 8 CUs over ARM + NEON

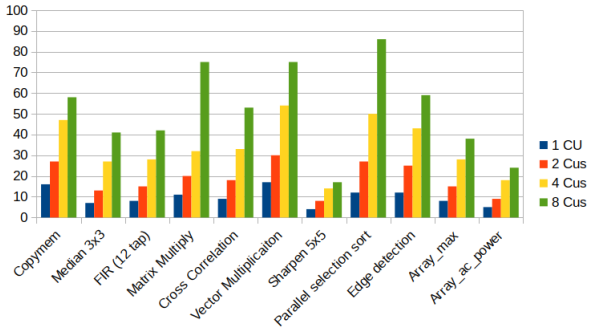

Figure 6. Speedup of soft-GPU over MicroBlaze

5. CONCLUSION

The proposed system is able to manage the Command & Control needs of an ICU on one side, and on the other side to analyse and process, directly on-board, data acquired by the instrument sensors through AI and computer vision algorithms, discarding meaningless data with a consequently optimisation of memory, bandwidth efficiency, and final data readiness. Additionally, the soft-GPU partial reconfigurability ensures the power efficiency in different load conditions. The hardware solution, coupled with a dedicated LLVM compiler infrastructure, allows an easy and fast software development, leaving to the final users the possibility of focusing only on their applications. In fact, the ICU4SAT has two LLVM FrontEnd interfaces dedicated to AI and computer vision, respectively TensorFlow and OpenCL frameworks. The key advantage of the ICU4SAT is to integrate all these functionalities in a single chip and, more in particular, on an FPGA (even space grade ones). The result is an all-in-one embedded reconfigurable smart ICU for image-based space missions in a single chip. This result has been achieved by integrating four hardware/software building blocks:

- The open source, fully customisable RISC-V processor as control unit in charge of data processing and handling;
- soft-GPU core as hardware accelerator in charge of implementing AI and computer vision algorithms;
- The SpaceFibre/SpaceWire IP-core as communication module in charge of providing standardised external high-speed interface to the other instrument and satellite modules.
- Dedicated LLVM compiler and framework for the ICU4SAT, which with its native compatibility with OpenCL and TensorFlow XLA, allows to port and compile directly different AI algorithms.

The proposed solution brings about a significant technology improvement in space applications especially at hardware accelerator technology level.

REFERENCES

[1] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[2] Pengyi Zhang, Yunxin Zhong, and Xiaoqiong Li. Slimyolov3: Narrower, faster and better for real-time uav applications. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[3] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Survey and benchmarking of machine learning accelerators. In *2019 IEEE high performance extreme computing conference (HPEC)*, pages 1–9. IEEE, 2019.

[4] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

[5] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[6] Brendan Barry, Cormac Brick, Fergal Connor, David Donohoe, David Moloney, Richard Richmond, Martin O'Riordan, and Vasile Toma. Always-on vision processing unit for mobile applications. *IEEE Micro*, 35(2):56–66, 2015.

[7] Sergio Rivas-Gomez, Antonio J Pena, David Moloney, Erwin Laure, and Stefano Markidis. Exploring the vision processing unit as co-processor for inference. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 589–598. IEEE, 2018.

[8] Gianluca Furano, Gabriele Meoni, Aubrey Dunne, David Moloney, Veronique Ferlet-Cavrois, Antonis Tavoularis, Jonathan Byrne, Léonie Buckley, Mihalis Psarakis, Kay-Obbe Voss, et al. Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. *IEEE Aerospace and Electronic Systems Magazine*, 35(12):44–56, 2020.

[9] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[10] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[11] Gianmarco Dinelli, Gabriele Meoni, Emilio Rapuano, Gionata Benelli, and Luca Fanucci. An fpga-based hardware accelerator for cnns using on-chip memories only: design and benchmarking with intel movidius neural compute stick. *International Journal of Reconfigurable Computing*, 2019, 2019.

[12] Mark P Baze, Steven P Buchner, and Dale McMorrow. A digital cmos design technique for seu hardening. *IEEE Transactions on Nuclear Science*, 47(6):2603–2608, 2000.

[13] L Sterpone, S Azimi, and B Du. A selective mapper for the mitigation of sets on rad-hard rtg4 flash-based fpgas. In *2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pages 1–4. IEEE, 2016.

[14] John Nickolls and William J Dally. The gpu computing era. *IEEE micro*, 30(2):56–69, 2010.

[15] Raghuraman Balasubramanian, Vinay Gangadhar, Ziliang Guo, Chen-Han Ho, Cherin Joseph, Jaikrishnan Menon, Mario Paulo Drumond, Robin Paul, Sharath Prasad, Pradip Valathol, et al. Enabling gpgpu low-level hardware explorations with miaow: An open-source rtl implementation of a gpgpu. *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(2):21–1, 2015.

[16] Kevin Andryc, Murtaza Merchant, and Russell Tessier. Flexgrip: A soft gpgpu for fpgas. In *2013 International Conference on Field-Programmable Technology (FPT)*, pages 230–237. IEEE, 2013.

[17] Muhammed Al Kadi, Benedikt Janssen, and Michael Huebner. Fgpu: An simt-architecture for fpgas. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 254–263, 2016.