6

Edge AI Acceleration for Critical Systems: from FPGA Hardware to CGRA Technology

Pietro Nannipieri, Luca Zulberti, Tommaso Pacini, Matteo Monopoli, Tommaso Bocchi, and Luca Fanucci

University of Pisa, Italy

Abstract

This work discusses advancements in edge AI processing for critical systems, focusing on satellites. We explore hardware solutions for real-time and autonomous data processing using Field Programmable Gate Arrays (FPGAs) and Coarse-Grained Reconfigurable Arrays (CGRAs). The research outlines critical systems' challenges, such as power constraints, radiation tolerance, and the need for reliable AI processing. The paper details the design and implementation of an HDL-based GPU system on an FPGA and the FPGA-AI framework that automates the design of AI accelerators on FPGA. Future research focuses on extending neural network support and exploring collaborations for tape-out opportunities of CGRA prototypes.

Keywords: edge AI, acceleration, space, satellite, critical systems, FPGA, CGRA.

6.1 Introduction

In the rapidly advancing landscape of technological innovation, Artificial Intelligence (AI) is proving to be a pivotal driver for enhancing system autonomy and performance across a wide range of critical applications. These include not only space-based systems, such as satellites used for Earth

observation and scientific research [1], but also autonomous vehicles, defence systems, healthcare technologies, and industrial automation. In these diverse fields, large volumes of complex data must be processed efficiently and in real-time to ensure timely decision-making, reliability, and operational success.

For instance, satellites carry various advanced sensors like cameras, radars, altimeters, scatterometers, and lidars, which produce huge volumes of data to be processed and analysed. Conventionally, these systems rely on limited onboard processing, where raw data is gathered in mass storage for subsequent downlink to ground stations for further analysis. In any event, with the increasing need for timely information related to Earth observation, scientific missions, and space exploration, there is an ever-growing requirement for real-time data processing [2]. This paradigm shift places significant pressure on the avionics subsystems to process and transmit larger quantities of data with minimal delay, necessitating a new approach to onboard computing.

Similarly, in other critical systems such as autonomous vehicles, military defence platforms, and medical devices, AI must function with extreme reliability, power efficiency, and real-time responsiveness [3]. Current solutions, such as data compression techniques or basic on-the-edge pre-processing, are far from giving an adequate answer to these complex demands. Besides, real-time AI processing in critical systems should be resistant to environmental challenges: whether it is radiation tolerance for space systems, power constraints for remote sensing equipment, or safety-critical computation in autonomous driving.

The need for hardware accelerators capable of executing AI algorithms with high efficiency, reliability, and adaptability is increasingly clear. Systems like Field Programmable Gate Arrays (FPGAs) [4], [5] and Coarse-Grained Reconfigurable Arrays (CGRAs) are emerging as leading candidates for such tasks. Because they can support parallel processing, adapt to changing workloads, and provide low-power yet high-performance computation, they are particularly suited to an environment where real-time responses and resilience to failure are paramount.

As industry agendas such as ESA's 'Space for a Green Future' vision for 2025 [6] look toward harnessing AI for environmental monitoring, disaster response, and real-time analytics, these hardware solutions will play a key role in advancing next-generation capabilities. Beyond space, applications such as autonomous navigation, object detection, and emergency response in

smart cities will require similarly robust, power-efficient AI systems that can operate seamlessly in challenging environments.

Edge AI represents a promising solution for developing more autonomous and real-time systems, paving the way for innovative applications that can address the complex challenges of modern critical environments. As we look towards a future with more autonomous and capable critical systems, AI will play a pivotal role in transforming how we observe, interact with, and understand them. However, it is well known that AI algorithms are computationally intensive. Therefore, it is necessary to identify reliable hardware solutions capable of supporting the execution of those algorithms in critical application scenarios.

In the following sections, we will present three ongoing projects to meet this technological need. In the short term, we propose an FPGA-based approach with two possible solutions, the GPU@SAT project and the FPG-AI project, a tool flow to automatically generate Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) hardware accelerators for FPGAs. In the long term, we propose the CGR-AI project to develop a radiation-hardened AI heterogeneous processing platform relying on a reconfigurable CGRA core.

6.2 State of the Art

Edge AI has garnered considerable interest among researchers in the field of space applications, particularly for scenarios where algorithms requiring acceleration must balance high-performance demands with stringent resource constraints. The choice of technology is inherently application-specific and driven by unique requirements, leading to a diverse array of proposed solutions and hardware implementations in the literature.

FPGAs have emerged as energy-efficient platforms for executing artificial intelligence algorithms [4]. Their inherent hardware parallelism aligns well with the computational demands of AI workloads. However, leveraging FPGAs for neural network acceleration poses challenges, primarily due to the significant engineering effort required to design custom, optimized hardware accelerators. To streamline the development and experimentation of Deep Neural Networks (DNNs) on FPGAs, several automation toolflows have been introduced. Tools such as Microchip VectorBlox AI [7], AMD Xilinx Vitis AI [8], and the MATLAB Deep Learning HDL Toolbox [9] support a wide range of models, enabling users, even those without extensive FPGA expertise, to efficiently deploy and accelerate DNNs on these platforms.

Conversely, hardware platforms like Graphics Processing Units (GPUs), which dominate commercial machine learning (ML) applications, are rarely utilized in space missions. This is primarily due to their higher power consumption and vulnerability to Single Event Latchups (SELs). Despite the significant potential of deploying GPUs in space, particularly for Deep Learning (DL) algorithms, to our knowledge, there have been no suitable solutions specifically designed for satellites. Previous research [10], [11] has primarily focused on evaluating the feasibility of using Commercial Off-The-Shelf (COTS) GPUs, despite their inherent limitations for space applications. Similarly, neuromorphic processors are gaining traction as a promising alternative for space-based applications [12].

CGRAs offer a compelling compromise between flexibility and efficiency. By combining spatial and temporal computation, CGRAs optimize performance for specific AI workloads without necessitating task-specific hardware redesign, achieving near-ASIC-level performance with software-like reconfigurability. For instance, the work presented in [13] integrates ML with CGRA compilation, significantly enhancing their accessibility and efficiency for practical applications. Another noteworthy advancement is Eyeriss v2 [14], a state-of-the-art DNN accelerator designed to enhance throughput and energy efficiency for both large-scale models like AlexNet and compact models such as MobileNet.

6.3 FPG-AI: Automation Tool Flow for Efficient Deployment of Pre-trained CNN/RNN Models on FPGA Technology

FPGAs are currently the most suitable solution for bringing efficient and performant AI capabilities onboard space missions. The high degree of parallelism offered by FPGAs is compatible with AI algorithms, allowing for spatial pipelining and achieving high efficiency and performance [4]. Besides this, there are several commercial Radiation-Hardened by Design (RHBD) FPGA solutions, which also make them a good option for space applications. However, the development time for an AI algorithm on FPGAs is long since these platforms involve high design effort with required intensive optimization due to their narrow resource budgets.

This challenge has been partially addressed by developing automatic AI accelerator generation frameworks [15]. These tools facilitate accelerating DNNs on FPGAs by a wide variety of users without any expertise. That is important because it reduces the development time while maintaining high

performance, hence making the use of FPGAs for AI in space missions possible.

FPG-AI [16] is a technology-independent toolflow for automating the acceleration of DNNs onboard FPGAs. It takes as input a pre-trained DNN model along with its application dataset. First, FPG-AI prepares the target network for hardware acceleration by performing optimizations on model topology and shifting arithmetic from Floating-point (FP) to Fixed-point (FXP). Then, a Design Space Exploration (DSE) process selects in the parameters space of the underlying architecture the point that meets the additional constraints provided by the user on application metrics, resource consumption, and performance. In this way, the generation of the target accelerator will be automated, and FPG-AI will be an end-to-end, ready-for-use tool while still keeping a high degree of customization concerning the user directives. Once the set of parameters is identified, DSE produces a configuration file for tuning the hardware architecture. The accelerator in the case of FPG-AI is the MDE, which can be a fully handcrafted HDL-based design that hosts no third-party IPs and can be customized according to the available resource primitives of FPGAs. Thanks to these properties, the MDE imposes no limit on device portability and allows, therefore, the implementation of components from different vendors with disparate resource budgets. Presently, this tool supports AMD XILINX, Microchip, Intel ALTERA, and NanoXplore FPGAs.

FPG-AI generates as output the HDL files describing an accelerator customized for the given model and device that meets the additional directives received by the user. Unlike other solutions, the toolflow provides the HDL sources of the accelerator and not the final bitstream, allowing the user to exploit the unused portion of the FPGA for complementary tasks. Figure 6.1 reports the block diagram of FPG-AI.

The FPG-AI framework is characterized by several key features that make it highly effective for deploying AI in space missions. One of the standout characteristics is technology independence, achieved through the MDE architecture. This solution's handmade HDL code without third-party IPs allows it to target multiple FPGA vendors, making the tool particularly suited for quickly supporting new devices.

Another significant feature is the high degree of scalability and customization of the hardware accelerator core, which allows users to set constraints on resource utilization (DSPs and on-chip-memories), inference time, and application metrics, tailoring the solution to the specific requirements of the space mission.

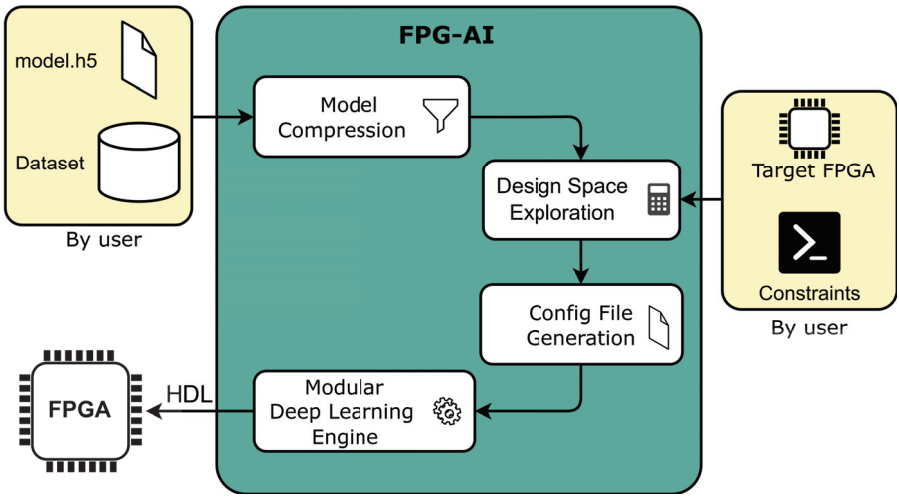

Figure 6.1 FPG-AI block diagram.

The ease of System-on-Chip (SoC) integration is guaranteed by the generation of an AXI Memory-Mapped accelerator which facilitates straightforward embedding into existing systems. In addition, FPG-AI's accelerators are completely autonomous during the computation of a neural network, eliminating the need to split the workload with the host CPU, improving efficiency and performance.

Currently, the FPG-AI solution supports Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), [17] broadening its applicability across different types of AI models. Combined with the aforementioned features, this versatility positions FPG-AI as a robust and adaptable choice for implementing AI in space applications, ensuring high performance and reliability in challenging environments.

6.3.1 Network-in-Network (NiN) case study

In the following, implementation results are reported for an image processing use case to exemplify the workflow of FPG-AI. The model selected for this study is Network-in-Network, a commonly used CNN model for image classification on the CIFAR10 dataset. Each image in CIFAR10 is a 32 $\times$ 32-pixel colour image, which means it has three colour channels (red, green, and blue) for each pixel. These images are divided into ten classes, with each class representing a different object or category. The NiN model involves three triples of Convolutional layers with padding set to "same". The

Convolutional layers are followed by a Global Average Pooling layer which provides predictions. Table 6.1 summarizes the features of the NiN.

As a first step, the pre-trained model undergoes the Model Compression process of FPG-AI to quantize the network and shift from FP to FXP arithmetic. Initially, pre-analyses are carried out to study the effects on model accuracy caused by different bit-width settings of each quantity within the network. In particular, the input dataset quantization pre-analysis evaluates the effects on the application metric depending on the input dataset bit-width and identifies the input dynamic range.

- Weights quantization pre-analysis evaluates the effects on application metrics depending on the weights bit-width and identifies the weights and biases dynamic ranges.
- Activations quantization pre-analysis: studies the effects on application metrics of different activation bit-widths by evaluating one layer at a time and identifies the activation dynamic range.

Table 6.1 Model summary of Network-in-Network.

Model Summary	
Model	NiN
Type	CNN
Dataset	CIFAR10
Input image dimension	$32 \times 32 \times 3$
# Convolutional layers	9
# Convolutional filters for each layer	192, 160, 96, 192, 192, 192, 192, 192, 10
Convolutional filters dimensions	5×5, 1×1, 3×3
# Pooling layers	3
Pooling filters dimensions	2×2, 8×8
Type of pooling	Max Pooling, Average Pooling
# Fully Connected layers	0
# Neurons for each FC layer	-
Total Parameters	969K
Memory for Parameters [Mbit]	3.69 Mb

After the pre-analysis, additional bit-true simulations are carried out to study the combined effects of quantized inputs, weights, and activations. The accuracy trend is analysed in a range across the optimal bit-widths provided by the pre-analyses. In this phase, different truncation settings are applied to modify activation bit-widths. The whole collection of simulation results is finally saved into a table that will be used during the DSE tool phase.The DSE then selects a quantized configuration among the ones generated by the Model Compression step. For this case study, we decided to extract the configuration

with the best application metric, obtaining a final accuracy of 88.88%.Thereafter, the DSE exploits an iterative algorithm to best configure the parameters of the MDE architecture. Once the DSE algorithm converges to a solution, the toolflow runs an automation script that generates all the necessary files for the hardware deployment of the network. As a first step, the chosen quantization parameters and the obtained architectural parameters are written on the MDE configuration file. The latter completes the HDL description of the MDE architecture configured to optimally accelerate the given quantized model on the selected FPGA device. At the end of this automatic process, the user can finally synthesize and implement the accelerator using the vendor-specific tools associated with the chosen FPGA component. Table 6.2 reports the configuration at maximum parallelism for the NiN model on FPGAs from diverse vendors. The results have been obtained by using the following tools with default synthesis and place-and-route strategies: Vivado Design Suite for AMD Xilinx, Synplify Pro for Microsemi, and Quartus Prime for Intel.

Table 6.2 NiN implementation results for maximum parallelism configuration.

Device	**ZU7EV**	**RTPF500T**	**XQRKU060**	**10AX048**
LUT	56375 (24.5%)	63028 (13.1%)	105950 (31.9%)	26115 (14.2%)
FF	14432 (3.1%)	28386 (5.9%)	20012 (3.02%)	12819 (1.7%)
LUTRAM/μSRAM/MLAB	1104 (3.1%)	0(0%)	552 (0.38%)	96 (1.4%)
BRAM/LSRAM/M20K	245.5 (78.7%)	512(33.7%)	277.5 (25.7%)	525(36.7%)
URAM	5 (5.2%)	-	-	-
DSP	1210 (70%)	1214(82%)	2410 (87.3%)	1229 (89.8%)
MDE Frequency [MHz]	126.6	50.8	72.5	83.33
Inference Time [ms]	4.54	11.33	5.39	218.88
Power [W]	2.88	-	3.07	3.07
Accuracy [%]	88.88	88.88	88.88	88.88

These experiments demonstrate the adaptability of the proposed quantization methods and the scalability of the MDE architecture. The accelerated models have been deployed on FPGA devices from different vendors (Intel, Xilinx, Microsemi) and various FPGA families, resulting in diverse characteristics in terms of technology, available resources, and radiation tolerance. This level of adaptability sets FPG-AI apart from other automation tools in the existing literature.

6.4 GPU@SAT: RISC-V Based SoC Featuring a Soft-GPU Hardware Accelerator for Artificial Intelligence On-board

In recent years, General-Purpose Computing on Graphic Processing Units (GPGPU) [18] has been gaining popularity due to its high energy efficiency and computational capabilities. Although Graphic Processing Units were initially utilized only for graphic applications, GPGPU has since bridged the gap into the use of GPUs in a wide range of tasks, including image processing, cryptography, and Machine Learning [19]. GPGPU exploits frameworks such as OpenCL [20] and CUDA [21], which provide high-level programming interfaces, allowing for a significant reduction in development time. Compatibility with open-source tools, including TensorFlow, PyTorch, and Caffe, also supports AI applications. Yet, its performance could be on a par with those of custom hardware accelerators, while - because of the lack of flexibility and customizability - GPGPU is not capable of efficiently executing distinct families of algorithms. As such, combining the advantages of GPGPU with the reconfigurability features of FPGAs could represent a viable solution for the acceleration of a wide range of ML algorithms in applications with strict constraints on cost, size, and power consumption (e.g., in space systems). This approach could offer a cost-effective and scalable solution for space missions that require high-performance computing capabilities. In this sense, soft GPGPU holds the potential to revolutionize the field of space exploration by enabling more complex and data-intensive tasks. In space applications, special attention must be given to power consumption, resource efficiency, and fault-tolerant design. To reduce costs and enable the use of commercial FPGAs for Low Earth Orbit (LEO) missions, it is crucial to prioritize fault-tolerant solutions.

An AI acceleration system compliant with the strict requirements of both low and high-level space missions (e.g., fault-tolerance, resource utilization, power consumption), exploiting the existing GPGPU framework, as well as FPGA reconfigurability, is still missing. As such, we propose the implementation of an SoC featuring the GPU@SAT IP from IngeniArs S.r.l. [22], [23], [24], [25], which can execute OpenCL 1.2 kernels. This could prove to be an efficient solution not only for the acceleration of ML algorithms but also for the execution of any OpenCL kernel in various fields of application (e.g., image processing, consumer applications, cryptography), ensuring a higher degree of flexibility. Figure 6.2 illustrates the proposed base architecture for the SoC. The primary objectives of the project are to develop

an AXI-compatible system incorporating the GPU@SAT IP and the open-source RISC-V architecture, with a strong emphasis on reliability—critical for the successful execution of high-level space missions. Additionally, we aim to characterize the system on various FPGA platforms, assessing performance-to-power efficiency metrics.

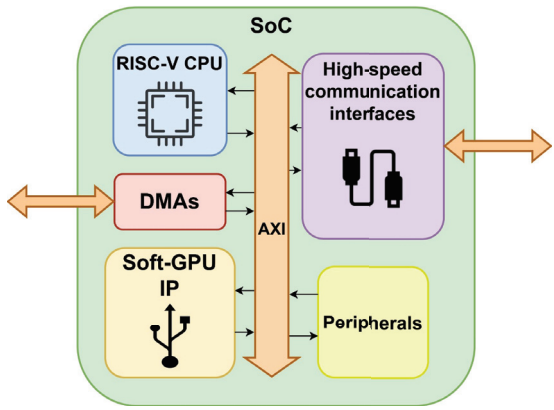

Figure 6.2 Overview of the System-on-Chip based on GPU@SAT.

6.4.1 Enhancing a soft GPU IP reliability against SEUs in space: Modelling approach and criticality analysis on a Radiation-Tolerant FPGA

To enhance the robustness of the GPU@SAT IP against Single Event Upsets (SEUs), we propose a detailed methodology revolving around modelling, fault criticality analysis, and a systematic application of fault mitigation techniques. Figure 6.3 shows a schematic overview of the GPU@SAT architecture and its main design modules.

We employ the Möbius [26] high-level modelling tool to estimate the reliability of the system. The tool supports both Fault Tree Models and Stochastic Activity Networks (SANs), but Fault Trees are used here due to their simplicity and alignment to assess fault impacts without accounting for repair mechanisms. The fault tree model allows the combination of basic events like failures of sub-modules using logical gates (AND, OR, XOR, etc.), making it possible to model how SEUs affect different components of the GPU architecture. Specifically, to quantify the SEU rates, we use data provided by Xilinx on the failure rates at Geosynchronous Earth Orbit (GEO) of the CRAM and Block RAMs in the target FPGA device, in our case the

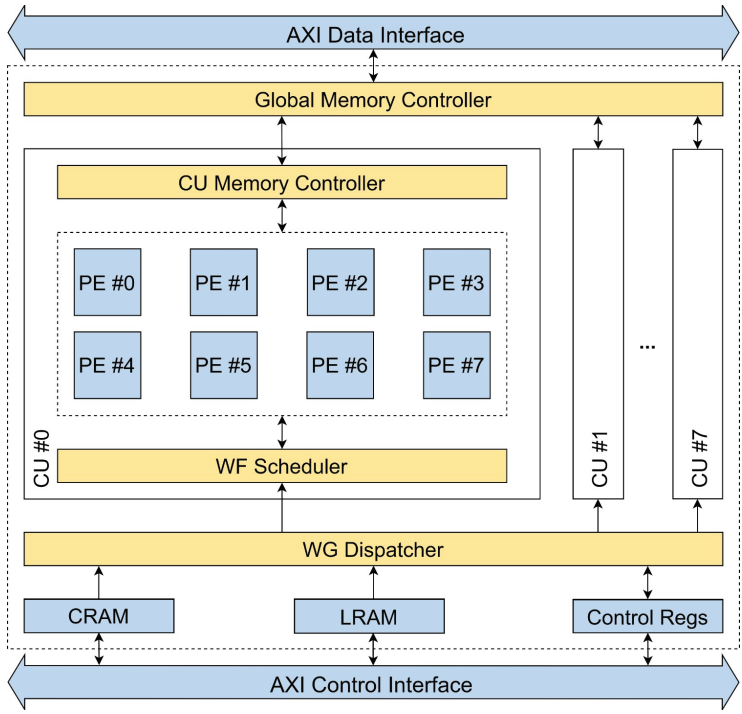

Figure 6.3 Overview of the GPU@SAT architecture.

Radiation-Tolerant Xilinx XQRKU060. These rates are applied to the various GPU components, estimating the worst-case SEU rate for each module based on the number of resources (e.g. LUTs, registers, BRAMs) used in each of them.

As SEU events can be modelled using a Poisson distribution [27], the firing time between them follows an exponential one. With this information, we modelled the entire GPU architecture to estimate a baseline reliability level using the Möbius tool. Table 6.3 presents the reliability estimation results for the GPU and its primary sub-components over 60 days.

The reliability analysis shows that the GPU has a lower reliability than its individual sub-components due to its high vulnerability to SEUs. The Global Memory Controller and CU Memory Controllers were identified as the least reliable parts, with high SEU sensitivity. Applying Triple Modular Redundancy (TMR) directly to these components would be too costly in terms of area and power, so partial redundancy techniques like Distributed TMR and Block TMR with voters are suggested. The results show that,

Table 6.3 Reliability values for the GPU and its components over a 60-day span.

Atomic Model	Time-Averaged Reliability
Workgroup Dispatcher	9.855E-01
Wavefront Scheduler	9.637E-01
Global Memory Controller	5.411E-01
RTM	9.752E-01
PE Array	6.372E-01
Control Interface	8.943E-01
CU Memory Controller	5.011E-01
GPU w/ 1 CU	2.421E-01

while no components exhibited extremely low reliability, strategic application of fault mitigation techniques can significantly increase system robustness without introducing prohibitive overhead.

Given the high resource consumption of a GPU, applying fault tolerance measures universally would be inefficient. Hence, we propose a classification-based methodology [28] that evaluates each component based on:

- **Criticality**: How essential the component is for correct operation.
- **Power/Area Impact**: How much power and area each component consumes on the FPGA.

Each component is classified into four criticality and power/area classes (C1-C4). The classification helps prioritize which components to protect more rigorously and which ones require less redundancy. The criticality of each component is assessed based on:

- **Fault Impact**: Evaluating both the spatial and temporal importance of the module in the GPU architecture.
- **SEU Sensitivity**: Based on the type of primitives (e.g., BRAM, registers, combinatorial logic) and their sensitivity to SEUs.
- **Component Type and Importance**: Importance is determined by how critical a module's function is for the correct operation of the GPU (e.g., if it's responsible for task dispatching or controlling memory access).
- **Power/Area (PA) Impact**: This accounts for how much area and dynamic power each component consumes on the FPGA. The analysis is based on post-place-and-route resource utilization and switching activity extracted from simulation benchmarks (e.g., matrix multiplication, convolution). Components that consume more resources are prioritized for less costly fault mitigation techniques to avoid excessive congestion and power consumption on the FPGA.

After analysing the criticality and PA impact, each component is assigned a suitable fault mitigation technique. These techniques are categorized into four classes, with higher classes corresponding to more impactful and resource-intensive solutions. Two distinct classification tables have been created to differentiate between operational and memory components, as these typically require the application of different mitigation techniques. Table 6.4 and Table 6.5 summarize the proposed techniques associated with different levels of criticality.

Table 6.4 Classification based on criticality, area, and power.

		Criticality			
		Class 1	**Class 2**	**Class 3**	**Class 4**
PA	**Class 1**	C2	C3	C4	C4
	Class 2	C2	C3	C3	C4
	Class 3	C1	C2	C3	C3
	Class 4	C1	C1	C2	C2

Table 6.5 Operational & Memory Components Classes.

Class	Operational Element	Memory
C1	Partial Duplication w/ Self Check	Parity Bit
C2	Partial Distributed TMR	DED Hamming Code
C3	Block TMR w/ 1 Voter	SEC-DED Hamming Code
C4	Block TMR w/ 3 Voters	TMR

6.5 CGR-AI: Innovative Coarse-Grained Reconfigurable Array Platform for Computing Artificial Intelligence On-Board

While FPGAs offer significant advantages for implementing AI in space, they have limitations. Performance and power efficiency are often constrained, and the cost per device, particularly for Radiation-Tolerant (RT) and Radiation-Hardened (RH) chips, can be substantial [29], [30]. To address these challenges, the long-term solution may lie in developing a specific hardware platform tailored for AI applications in space. Such a platform, built on RT/RH technology, could offer the throughput required by applications and the power efficiency required by the environment, justifying the large-scale adoption of AI in space missions [31].

However, maintaining a degree of flexibility is crucial. Designing an Application-Specific Integrated Circuit (ASIC) for a specific Neural Network (NN) accelerator is not sustainable due to the diverse and evolving nature of AI applications [32]. Instead, future hardware platforms should balance high performance with adaptability, supporting various AI models and the ability to update and reconfigure as mission requirements change. This approach would ensure that space AI systems remain robust, efficient, and versatile, capable of meeting the demanding conditions of space exploration and observation [33].

The purpose of the CGR-AI project is to develop an AI engine IP, Complementary to RISC-V/FPGA for AI workload data crunching, powered by a CGRA accelerator, to be implemented in std-cell technology. Such engine shall overcome existing solutions in terms of:

- Flexibility: the platform may be easily adapted to a given application by executing specific firmware to schedule memory transactions and load CGRA configurations.
- Time Predictability: the CGRA-based architecture performs operations in a fixed number of clock cycles. Given the global memory timing model, each AI Engine operation is completed within a deterministic time bound, ensuring algorithm execution for time-critical applications.
- Reliability: library and architectural solutions to withstand radiations typical of higher orbits. Furthermore, CGRA resources can be used to apply dynamic redundancy to the data path, exploiting dedicated hardware voters for error correction.

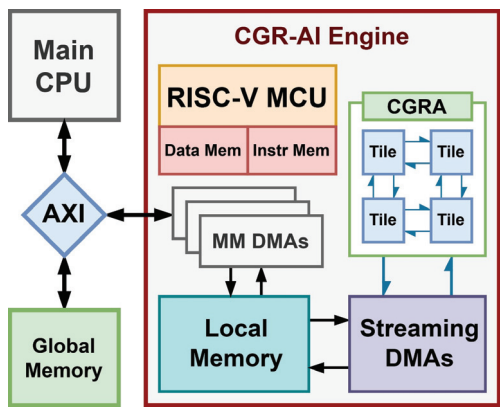

Figure 6.4 CGR-AI Engine block diagram.

Figure 6.4 depicts the proposed CGRA-based AI Engine architecture. The CGRA [34], [35] is an AI accelerator capable of performing linear and approximated nonlinear vector operations with dynamic element width. The RISC-V CPU runs the firmware loaded into the Tightly-Coupled Memories (TCMs) and overlooks all the operations within the architecture. The streaming Direct Memory Accesses (DMAs) provide custom pattern access to/from local memory, and the Memory-Mapped (MM) DMAs provide access to/from external memory. AXI bus is implemented to allow communication with multi-core application class CPU, commercial peripherals, and main memory/DDR controller. The AI Engine can parallelize common workloads in neural network algorithms, like convolutions, pooling, and activation functions. The spatial architecture makes the engine capable of distributing resources for different algorithms, enabling load balancing during different mission stages with dynamic requirements.

The project is currently in development, and we are in the process of setting up an FPGA prototype and characterising the processing core on std-cell technology. As preliminary results, in Table 6.6 we report synthesis data obtained on the 65nm technology for an operating frequency of 625 MHz.

The analysis is focused on the single Tile module since its results can be composed to form the desired matrix.

For each port configuration, we adapt the number of input and output ports of the PE accordingly to the number of Functional Units (FUs) implemented (third column), in order:

1. ambslx: all FUs are present.
2. ambsx: without LUT FU, which is the one with more utilization of resources.
3. ambs: without LUT and MUX FUs.
4. am: only with ADD and MUL (e.g., to perform MAC in convolutional layers)
5. lx: only with LUT and MUX (e.g., to execute activation functions)

We refer to (a) as throughput [GOps]; efficacy is reported in (b) as area efficiency [GOps/mm2], where the theoretical throughput of the configurations is divided by the related area and power metrics, and the best solutions are highlighted in bold, and (c) as energy efficiency [GOps/W], a direct measure of the energy needed for each operation: its inverse is W/GOps = W/(Gop/s) = J/Gop.

Table 6.6 Throughput compared to Area and Energy Efficiencies.

# In # Out	**FUs**	**625 MHz**		
		(a)	(b)	(c)
3 In 2 Out	ambslx	3.75	17.7	88
	ambsx	3.12	19.4	96.3
	ambs	2.50	22.2	111
	am	1.25	22.9	119
	lx	1.25	**27.4**	142
4 In 4 Out	ambslx	3.75	16.1	81
	ambsx	3.12	18.6	93.7
	ambs	2.50	21.4	110
	am	1.25	**26.5**	151
	lx	1.25	**24.8**	129
4 In 4 Out	ambslx	3.75	15.4	80.6
	ambsx	3.12	17.3	90
	ambs	2.50	18	94.9
	am	1.25	16.1	88.4
	lx	1.25	**22.7**	120

6.6 Discussion and Conclusion

The execution of AI algorithms in critical systems is becoming increasingly important as the demand for real-time data processing and autonomous decision-making grows. Satellites with advanced AI capabilities can enhance applications such as Earth observation, scientific research, and space exploration by providing timely insights and efficient data management. However, addressing the diverse requirements of performance, radiation tolerance, and cost requires various solutions.

The GPU@SAT approach represents an immediate answer for applications needing an SW-programmable system, whose reliability has been analysed and improved. Such a solution can be considered particularly interesting thanks to the support both from Rad-hard and rad-tolerant FPGAs, thus suitable for space applications.FPG-AI provides an immediate, flexible solution for these needs. The fact that it can afford high parallelism, ease of customization, and integration makes FPG-AI a very promising candidate for both present and near-future space missions. The radiation-hardened versions of FPGAs can therefore offer such resilience against the harsh conditions in space.However, these have their performance and power efficiency limitations, not to mention the high cost, hence the need for continued innovation. Overcoming these limitations will long-term be so crucial to sustain further advancements in AI across many critical applications. The development

of a dedicated hardware platform like CGR-AI, therefore, emerges as a strong candidate for such a future direction. The objective is to develop a platform that integrates high performance and efficiency, state-of-the-art AI applications, flexibility in handling changing mission needs, and varied AI models.While it may be the case that GPU@SAT and FPG-AI can at present work as a robust and versatile solution for current AI needs onboard satellites, continuous research, and development into more specialized hardware platforms, such as CGR-AI, will be very important. These efforts will ensure that the potential of AI in space is fully realized, supporting more autonomous, efficient, and capable satellite systems that can meet the challenges of tomorrow's space exploration and Earth observation missions.

Acknowledgements

This work is partially supported by the European Space Agency (ESA) in the framework of the Open Space Innovation Platform (OSIP) co-funded research under the ESA contract No. 4000137503, 4000141235, and 4000141108. This study received funding from the European Union - Next-GenerationEU - National Recovery and Resilience Plan (NRRP) – MISSION 4 COMPONENT 2, INVESTMENT N. 1.1, CALL PRIN 2022 PNRR D.D. 1409 14-09-2022 – (OPERAND - a recOnfigurable Platform & framEwork foR Ai infereNce on the eDge) CUP N.I53D23006070001

References

[1] Izzo, Dario, et al. "Selected trends in artificial intelligence for space applications." *Artificial Intelligence for Space: AI4SPACE*. CRC Press, 2022. 21-52.

[2] Q. Li, S. Wang, X. Ma, A. Zhou, and F. Yang, "Towards Sustainable Satellite Edge Computing," arXiv (Cornell University), Sep. 2021, doi: https://doi.org/10.1109/edge53862.2021.00010.

[3] S. Ao, "Building Safe and Reliable AI Systems for Safety Critical Tasks with Vision-Language Processing," *Lecture notes in computer science*, pp. 423–428, Jan. 2023, doi: https://doi.org/10.1007/978-3-031-28241-6_47.

[4] T. Wang, C. Wang, X. Zhou, and H. Chen, "An Overview of FPGA Based Deep Learning Accelerators: Challenges and Opportunities," IEEE Xplore, Aug. 01, 2019. https://ieeexplore.ieee.org/abstract/document/8855594 (accessed Aug. 14, 2023).

[5] El and Mohsine Eleuldj, "Survey of Deep Learning Neural Networks Implementation on FPGAs," Nov. 2020, doi: https://doi.org/10.1109/cloudtech49835.2020.9365911.

[6] "Space for a green future – ESA Vision." https://vision.esa.int/space-for-a-green-future/

[7] "VectorBlox^{TM} Accelerator SDK," *Microchip.com*, 2025. https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/vectorblox

[8] "AMD Technical Information Portal," *Amd.com*, 2024. https://docs.amd.com/r/en-US/ug1414-vitis-ai

[9] "Deep Learning HDL Toolbox," *Mathworks.com*, 2025. https://it.mathworks.com/products/deep-learning-hdl.html

[10] W. Powell, M. Campola, T. Sheets, A. Davidson, and S. Welsh, "Commercial Off-The-Shelf GPU Qualification for Space Applications," Sep. 2018.

[11] Leonidas Kosmidis, J. Lachaize, J. Abella, Olivier Notebaert, F. J. Cazorla, and D. Steenari, "GPU4S: Embedded GPUs in Space," *UPCommons institutional repository (Universitat Politècnica de Catalunya)*, Aug. 2019, doi: https://doi.org/10.1109/dsd.2019.00064.

[12] D. Izzo, A. Hadjiivanov, D. Dold, G. Meoni, and E. Blazquez, "Neuromorphic Computing and Sensing in Space," *CRC Press eBooks*, pp. 107–159, Nov. 2023, doi: https://doi.org/10.1201/9781003366386-4.

[13] Y. Luo *et al.*, "ML-CGRA: An Integrated Compilation Framework to Enable Efficient Machine Learning Acceleration on CGRAs," pp. 1–6, Jul. 2023, doi: https://doi.org/10.1109/dac56929.2023.10247873.

[14] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, Jun. 2019, doi: https://doi.org/10.1109/jetcas.2019.2910232.

[15] S. I. Venieris, A. Kouris, and Christos-Savvas Bouganis, "Toolflows for Mapping Convolutional Neural Networks on FPGAs," ACM Computing Surveys, vol. 51, no. 3, pp. 1–39, Jun. 2018, doi: https://doi.org/10.1145/3186332.

[16] T. Pacini, E. Rapuano, and L. Fanucci, "FPG-AI: A Technology-Independent Framework for the Automation of CNN Deployment on FPGAs," IEEE Access, vol. 11, pp. 32759–32775, 2023, doi: https://doi.org/10.1109/ACCESS.2023.3263392.

[17] T. Pacini, E. Rapuano, L. Tuttobene, Pietro Nannipieri, L. Fanucci, and S. Moranti, "Towards the Extension of FPG-AI Toolflow to RNN Deployment on FPGAs for On-board Satellite Applications," Oct. 2023, doi: https://doi.org/10.23919/edhpc59100.2023.10396607.

[18] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, "GPU Computing Graphics Processing UnitsVpowerful, programmable, and highly parallelVare increasingly targeting general-purpose computing applications," doi: https://doi.org/10.1109/JPROC.2008.917757.

[19] J. D. Owens *et al.*, "A Survey of General-Purpose Computation on Graphics Hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, Mar. 2007, doi: https://doi.org/10.1111/j.1467-8659.2007.01012.x.

[20] A. Munshi, "The OpenCL specification," Aug. 2009, doi: https://doi.org/10.1109/hotchips.2009.7478342.

[21] Kirk, David. "NVIDIA CUDA software and GPU parallel computing architecture." ISMM. Vol. 7. 2007.

[22] M. Monopoli, Luca Zulberti, G. Todaro, Pietro Nannipieri, and L. Fanucci, "Exploiting FPGA Dynamic Partial Reconfiguration for a Soft GPU-based System-on-Chip," Jun. 2023, doi: https://doi.org/10.1109/prime58259.2023.10161859.

[23] G. Todaro, M. Monopoli, G. Benelli, Luca Zulberti, and T. Pacini, "Enhanced Soft GPU Architecture for FPGAs," Jun. 2023, doi: https://doi.org/10.1109/prime58259.2023.10161749.

[24] M. Monopoli, Luca Zulberti, Pietro Nannipieri, L. Fanucci, and S. Moranti, "Exploring Key Aspects of Soft GPGPU Computing for On-board Acceleration of Artificial Intelligence Algorithms in Space Applications," *vol. 28, pp. 1–6, Oct. 2023,* doi: https://doi.org/10.23919/edhpc59100.2023.10396624.

[25] G. Benelli, G. Giuffrida, R. Ciardi, D. Davalle, G. Todaro, and L. Fanucci, "GPU@SAT, the AI enabling ecosystem for on-board satellite applications," vol. 19, pp. 1–4, Oct. 2023, doi: https://doi.org/10.23919/edhpc59100.2023.10396289.

[26] G. Clark *et al.*, "The Mobius modeling tool," Nov. 2002, doi: https://doi.org/10.1109/pnpm.2001.953373.

[27] Gercek, Gokhan. "A method to compute SEU fault probabilities in memory arrays with error correction." Dual-Use Space Technology Transfer Conference and Exhibition, Volume 2. 1994.

[28] Monopoli, M., Biondi, M., Todaro, G., Nannipieri, P., Moranti, S., Bernardeschi, C., & Fanucci, L. (2024). *Enhancing a Soft GPU IP*

Reliability Against SEUs in Space: Modelling Approach and Criticality Analysis on a Radiation-Tolerant FPGA. https://doi.org/10.36227/techrxiv.173273657.76155928/v1

[29] E. Ibe, H. Taniguchi, Yasuo Yahagi, Kenichi Shimbo, and T. Toba, "Impact of Scaling on Neutron-Induced Soft Error in SRAMs From a 250 nm to a 22 nm Design Rule," vol. 57, no. 7, pp. 1527–1538, May 2010, doi: https://doi.org/10.1109/ted.2010.2047907.

[30] T. Li, H. Liu, and H. Yang, "Design and Characterization of SEU Hardened Circuits for SRAM-Based FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, pp. 1276–1283, Jun. 2019, doi: https://doi.org/10.1109/tvlsi.2019.2892838.

[31] G. Furano *et al.*, "Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 44–56, Dec. 2020, doi: https://doi.org/10.1109/MAES.2020.3008468.

[32] Y. Duan, J. S. Edwards, and Y. K. Dwivedi, "Artificial Intelligence for Decision Making in the Era of Big Data – evolution, Challenges and Research Agenda," *International Journal of Information Management*, vol. 48, no. 1, pp. 63–71, Oct. 2019, doi: https://doi.org/10.1016/j.ijinfomgt.2019.01.021.

[33] L. Mandrake *et al.*, "Space Applications of a Trusted AI Framework: Experiences and Lessons Learned," *IEEE Xplore*, Mar. 01, 2022. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9843322

[34] Luca Zulberti, M. Monopoli, Pietro Nannipieri, L. Fanucci, and S. Moranti, "Highly Parameterised CGRA Architecture for Design Space Exploration of Machine Learning Applications Onboard Satellites," *INDIGO (University of Illinois at Chicago)*, pp. 1–6, Oct. 2023, doi: https://doi.org/10.23919/edhpc59100.2023.10396632.

[35] Luca Zulberti, Monopoli, M., Pietro Nannipieri, Moranti, S., Thys, G., & Fanucci, L. (2025). Efficient Coarse-Grained Reconfigurable Array architecture for machine learning applications in space using DARE65T library platform. *Microprocessors and Microsystems*, 105142–105142. https://doi.org/10.1016/j.micpro.2025.105142