electronics

MDPI

Article

GPU@SAT DevKit: Empowering Edge Computing Development Onboard Satellites in the Space-IoT Era

Gionata Benelli [1,2,*], Giovanni Todaro [1,2], Matteo Monopoli [1], Gianluca Giuffrida [2], Massimiliano Donati [1,*] and Luca Fanucci [1]

[1] Department of Information Engineering, University of Pisa, 56122 Pisa, Italy; giovanni.todaro@phd.unipi.it or giovanni.todaro@ingeniars.com (G.T.); matteo.monopoli@phd.unipi.it (M.M.); luca.fanucci@unipi.it (L.F.)

[2] Space Division, IngeniArs S.r.l., 56121 Pisa, Italy; gianluca.giuffrida@ingeniars.com

* Correspondence: gionata.benelli@phd.unipi.it or gionata.benelli@ingeniars.com (G.B.); massimiliano.donati@unipi.it (M.D.)

Abstract: Advancements in technology have driven the miniaturization of embedded systems, making them more cost-effective and energy-efficient for wireless applications. As a result, the number of connectable devices in Internet of Things (IoT) networks has increased significantly, creating the challenge of linking them effectively and economically. The space industry has long recognized this challenge and invested in satellite infrastructure for IoT networks, exploiting the potential of edge computing technologies. In this context, it is of critical importance to enhance the onboard computing capabilities of satellites and develop enabling technologies for their advancement. This is necessary to ensure that satellites are able to connect devices while reducing latency, bandwidth utilization, and development costs, and improving privacy and security measures. This paper presents the GPU@SAT DevKit: an ecosystem for testing a high-performance, general-purpose accelerator designed for FPGAs and suitable for edge computing tasks on satellites. This ecosystem provides a streamlined way to exploit GPGPU processing in space, enabling faster development times and more efficient resource use. Designed for FPGAs and tailored to edge computing tasks, the GPU@SAT accelerator mimics the parallel architecture of a GPU, allowing developers to leverage its capabilities while maintaining flexibility. Its compatibility with OpenCL simplifies the development process, enabling faster deployment of satellite-based applications. The DevKit was implemented and tested on a Zynq UltraScale+ MPSoC evaluation board from Xilinx, integrating the GPU@SAT IP core with the system's embedded processor. A client/server approach is used to run applications, allowing users to easily configure and execute kernels through a simple XML document. This intuitive interface provides end-users with the ability to run and evaluate kernel performance and functionality without dealing with the underlying complexities of the accelerator itself. By making the GPU@SAT IP core more accessible, the DevKit significantly reduces development time and lowers the barrier to entry for satellite-based edge computing solutions. The DevKit was also compared with other onboard processing solutions, demonstrating similar performance.

Keywords: GPU; FPGA; deep learning; edge computing

check for updates

Citation: Benelli, G .; Todaro, G.; Monopoli, M.; Giuffrida, G.; Donati, M.; Fanucci, L. GPU@SAT DevKit: Empowering Edge Computing Development Onboard Satellites in the Space-IoT Era. *Electronics* **2024**, *13*, 3928. https://doi.org/10.3390/electronics13193928

Academic Editor: Silvia Liberata Ullo

Received: 2 September 2024
Revised: 27 September 2024
Accepted: 1 October 2024
Published: 4 October 2024

1. Introduction

The IoT is a rapidly evolving technology that is transforming the landscape of modern technological innovations. Enabling the connection of a multitude of devices simultaneously is fundamentally changing the way technological devices interact with each other and the way we interact with them. The IoT has the potential to revolutionize numerous industries and fields by fostering innovation, simplifying processes, and improving convenience. Its applications have already been implemented in various domains, including smart homes, intelligent spaces, energy management systems, and smart agriculture [1–3]. Additionally, it has been utilized in critical applications such as remote surgery and connected vehicles [4].

The primary factor favoring the pervasive integration of IoT in everyday life is the rapid advancement of technology, particularly in pivotal domains such as the miniaturization of low-power electronics and the enhancement of wireless connectivity [5]. These developments have facilitated a notable increase in the number of interconnected devices. Recent surveys indicate that while there were 13.9 billion connected devices worldwide in 2022, by 2025, this number will exceed 20 billion [6]. This rapid growth is also anticipated to persist, potentially reaching nearly 40 billion IoT devices by 2033. Other studies indicate that this number could reach 75 billion as early as 2025 [7]. The trend is on the rise, and while it is fostered by technology, it is important to recognize that it is also largely encouraged by companies that identify the potential of such technology and invest in it [8].

As the number of devices in operation continues to proliferate, so do the challenges associated with managing them effectively. These challenges include ensuring sufficient coverage in diverse environments, maintaining uninterrupted connectivity, ensuring reliability, and facilitating scalability. To date, no single communication infrastructure has been able to seamlessly connect all IoT devices globally, while simultaneously resolving the aforementioned issues [1]. A paradigm shift in infrastructure has the potential to optimize the widespread adoption of IoT technologies.

In response to these demands, the space industry has long been integrating satellites into the IoT ecosystem. This strategic move is fueled by the unique attributes of satellites, particularly those orbiting in Low Earth Orbit (LEO) and in Very Low Earth Orbit (VLEO), which effectively tackle the challenges at hand [9,10]. Indeed, LEO satellites provide global coverage, overcoming the challenges of connectivity in remote areas and rugged terrain. Furthermore, constellation architectures ensure reliability and scalability, accommodating the proliferation of IoT devices across vast geographical areas.

This process has led to the birth of the *Space Internet of Things (Space-IoT)* paradigm, which consists of using a network of satellites to solve the main challenges in the development of IoT networks on Earth [11]. Other times, the same concept has been expressed as *satellite-based IoT* [12] or simply *satellite IoT* [13]. These expressions highlight the growing trend of using satellite technology to facilitate IoT communication and connectivity on a global scale. In essence, the Space-IoT market is poised to significantly shape the future IoT landscape.

Edge Computing Onboard Satellites for Space-IoT

Numerous companies are already leveraging satellite constellations to address the needs of IoT applications. For example, the Iridium NEXT network by Iridium [14] comprises 66 interconnected satellites and is tailored to support IoT devices in remote and maritime environments. Similarly, the constellation of 24 operational satellites by Globalstar [15] provides voice and data services, bridging the gap between satellite and terrestrial networks across various sectors, including maritime, aviation, and emergency services. Orbcomm [16], with a focus on Machine-to-Machine (M2M) and IoT communication, ensures reliable two-way connectivity for asset tracking and remote monitoring in numerous industries.

As more companies explore the Space-IoT market [17–20], there is an increasing drive to reduce satellite size and operational costs while maintaining reliable communication capabilities. This demand has fueled the adoption of pico-satellites, such as *CubeSats*, which typically weigh between 0.1 and 1 kg [21]. These small, lightweight satellites offer significant advantages in terms of cost and energy efficiency. However, their constrained size and power resources necessitate downsizing and optimizing onboard electronics to enable the development of a global Space-IoT infrastructure.

A promising solution to these challenges is the deployment of computing power at the *edge* of the network, i.e., directly on satellites, through the *edge computing* paradigm. This approach can address many limitations associated with traditional *cloud computing* [22]. By moving computation closer to the data source, edge computing can reduce latency, optimize bandwidth use, enhance reliability and scalability, lower costs, and even improve security [22,23]. These advantages can be attained through the augmentation of computational capabilities in both *platform computers*—essential for controlling critical functions like power distribution, navigation, and spacecraft guidance—and *payload computers*, enabling them to handle larger volumes of data. Other studies even suggest rethinking the canonical structure of a satellite to make it more suitable as a node in an IoT network [24]. These benefits make edge computing an increasingly attractive strategy for Space-IoT systems.

Unsurprisingly, research efforts are more and more focused on advancing edge computing for satellites [25–29]. Many of these studies underscore the benefits of onboard processing, while also acknowledging the significant challenges posed by the limited computational resources and power constraints inherent to satellite systems [30]. A particularly promising avenue of research is the acceleration of Machine Learning (ML) and Deep Learning (DL) algorithms onboard satellites [31–36]. This approach opens new possibilities for real-time processing and decision-making in space-based IoT applications. For instance, ref. [36] proposes a novel application scheme for satellite IoT edge computing based on DL. In particular, the authors perform simulations to highlight the significant role played by this approach in image detection. Specifically, they demonstrate how combining edge computing and DL could be beneficial in reducing acquisition time and bandwidth utilization. On the other hand, ref. [34] outlines use cases where onboard Artificial Intelligence (AI) enhances satellite autonomy and control, thereby improving overall Quality of Service (QoS). Nonetheless, the study also highlights the substantial computing power required, which current space-grade processors struggle to meet.

Selecting the appropriate hardware for deploying these algorithms is crucial. Commercial Off-The-Shelf (COTS) components are often used to reduce costs and accelerate development timelines. However, these components are frequently unsuitable for the harsh space environment, where resistance to radiation, vibration, and extreme temperatures is critical. The constant exposure to cosmic rays and high-energy protons significantly increases the risk of Single Event Effects (SEEs) in these devices [37]. While some of these effects, such as Single Event Upsets (SEUs), are non-destructive soft errors, others, like Single Event Latchups (SELs), can have catastrophic consequences and are therefore unacceptable. Moreover, radiation exerts a long-term detrimental effect on hardware, notably reducing the overall lifespan of the mission.

Field Programmable Gate Arrays (FPGAs) offer a viable alternative. One of their key advantages is their availability in a wide range of silicon-grade technologies, making them well suited for space applications. For instance, radiation-hardened FPGAs like the Xilinx Virtex-5QV are immune to SELs and highly resistant to SEUs [38]. Due to their robustness, space-qualified FPGAs have been a reliable choice for space missions for many years [39], establishing themselves as a proven technology in the industry. Although they may involve higher initial costs and longer development times compared to COTS solutions, their customizable architecture allows for optimized performance and energy efficiency, making them ideal for accelerating compute-intensive tasks, such as DL algorithms [40]. In contrast, hardware platforms like Graphic Processing Units (GPUs), which dominate commercial applications for executing ML tasks, are rarely used in space due to their higher power consumption and susceptibility to SELs. Indeed, to the best of our knowledge, no space-grade GPU has yet been introduced to the market. Consequently, FPGAs are the preferred platform in space for onboard parallel computation [41].

To advance the deployment of FPGA-based technologies for edge computing algorithms on satellites, and to foster the development of the Space-IoT paradigm, this paper introduces the GPU@SAT DevKit. This innovative ecosystem is tailored to assess a versatile, high-performance accelerator (GPU@SAT) [42,43] specifically designed for edge computing tasks, i.e., image processing, which requires handling computationally intensive operations like convolutional layers. The accelerator features a parallel architecture, comprising multiple scalable cores and compatibility with the open-source OpenCL 1.2 programming framework, resembling the functionality of a GPU. The design is well suited for executing DL algorithms through the General Purpose computing on GPU (GPGPU) paradigm, leveraging GPU processing power for versatile computations beyond traditional graphics rendering. Additionally, it is entirely VHDL-coded and optimized for FPGAs, free from third-party Intellectual Property (IP) dependencies.

The remainder of this paper proceeds as follows: Section 2 explores FPGA-based solutions for accelerating DL algorithms, with a focus on the GPU@SAT architecture and its rationale for adoption. Section 3 provides an in-depth technical overview of the DevKit and illustrates the execution flow of a kernel. Section 4 presents the benchmarking utilized to validate the functionality of the DevKit, alongside initial results from the accelerator, demonstrating its promise in edge computing applications. Finally, Section 5 provides a summary of the key findings and outlines future research directions.

2. Background and Related Work

Research has long invested in the use of FPGA-based hardware for accelerating DL algorithms onboard satellites.

The work of [44] presents an embedded accelerator for a hybrid Neural Network (NN). This hybrid model integrates a classical Convolutional Neural Network (CNN) for feature extraction with a *spiking fully connected* stage for classification. The architecture optimizes resource utilization without significantly sacrificing accuracy. Notably, this proposed architecture has been integrated into the European Space Agency (ESA) OPS-SAT experimental satellite, which was launched on 18 December 2019 [45].

In [46], the authors present a novel approach to AI edge processing on Earth Observation (EO) satellites. Unlike the traditional bent pipe model, where images captured by the satellite are transmitted to ground stations for processing when the satellite is in the optimal position, this approach involves storing each image directly in mass memory and processing it on board using an AI engine. The authors also propose potential solutions for deploying Deep Neural Networks (DNNs) on the edge. Specifically, for missions with low to medium dependability requirements, they suggest the use of COTS components like Xilinx Ultrascale+ FPGAs.

In [47], the authors propose a hybrid FPGA and System-on-a-Chip (SoC) system for CNN acceleration in the context of spacecraft pose estimation tasks. They use the Xilinx Zynq Ultrascale+ Multi-Processor System-on-a-Chip (MPSoC) as the evaluation platform, exploiting the FPGA's Programmable Logic (PL) to perform inference. The presented implementation runs on two different CNN models and is shown to achieve an accuracy level comparable to a PC-based inference.

Research has also been conducted on NanoXplore European radiation-hardened FPGA devices. In [48], the NanoXplore NG-LARGE device is evaluated against Digital Signal Processing (DSP) and Computer Vision (CV) benchmarks. Specifically, the authors present a successful implementation of the full HW/SW processing pipeline for Vision-Based Navigation (VBN), including high-speed data link communication through the SpaceWire protocol.

In another paper [49], the researchers propose an FPGA-based accelerator tailored for the inference of the CloudScout CNN [50] and compare its performance to the Movidius Intel Myriad 2 Vision Processing Unit (VPU). The results suggest that while FPGA-based approaches may have higher power consumption and longer time-to-market compared to COTS solutions, they significantly improve inference speed and adaptability.

More recently, in [51], the researchers propose CloudSatNet-1, an FPGA-based hardware-accelerated quantized CNN tailored for onboard satellite cloud coverage classification. This model achieves up to 94.4% accuracy and also reduces memory consumption. However, further refinement and testing are needed to improve the classification accuracy on different datasets.

Undoubtedly, the growing interest in using FPGAs in space for DL algorithm inference takes advantage of the benefits demonstrated by these and other works [52]. Research indicates that, in certain aspects, FPGAs offer a more cost-effective and versatile solution compared to more powerful alternatives like GPUs [53]. However, the latter typically have higher throughput and peak speed, at the cost of higher power consumption.

Despite the significant potential of deploying GPUs in space, particularly for DL algorithms, to our knowledge, there have been no suitable solutions specifically designed for satellites. Previous research [54–57] has primarily focused on assessing the viability of COTS GPUs, despite their inherent challenges in the space environment.

Neuromorphic processors are also emerging as a promising technology for space applications [58,59]. These processors, designed to mimic the brain's neural architecture, offer ultra-low power consumption and real-time processing capabilities, making them ideal for resource-constrained environments like space. However, they are mostly suited for event-based applications, such as anomaly detection, where processing is triggered by specific events rather than continuous data streams. In this regard, they may complement traditional ML accelerators rather than replace them across the broader range of space applications.

An alternative approach involves using an embedded GPU implemented on FPGA fabric, which could overcome these obstacles. While this solution may not match the efficiency and performance of a custom design, it offers greater flexibility, extensibility, and customization. It can be developed across various silicon technologies and supports parallel computation at the edge of applications. This method allows combining both the high-performance capabilities of a GPU-like architecture and the reduced power consumption of FPGA technology, making it well suited for ML edge applications that require both computational power and low energy consumption. GPU@SAT exemplifies this approach.

GPU@SAT Design

GPU@SAT is an IP core designed for FPGAs that is coded in VHDL and does not incorporate third-party IP components. The design operates on a 32-bit architecture and follows the Single Instruction Multiple Thread (SIMT) execution model of a theoretical OpenCL device. To date, the level of compatibility remains incomplete, with the current version of the standard being the OpenCL 1.2 specification.

Figure 1 depicts the structure of an OpenCL device. In schematic form, such a device comprises a multitude of cores that operate in parallel to execute a program—which is referred to as a *kernel*—for a host processor that drives it. In this context, the OpenCL device can be considered a hardware accelerator that serves the processor for offloading tasks. A kernel is composed of multiple threads, designated as Work-Items (WIs), which run in parallel. To facilitate the division of workload among the cores in the architecture, the WIs are subsequently grouped into larger sets, identified as Work-Groups (WGs) [60].

Similarly, the GPU@SAT IP core is divided into several cores, called Compute Units (CUs), with each core containing up to eight Processing Elements (PEs). In accordance with the standard, the WGs are executed on the cores, while the WIs are run on the PEs. The sole distinction between the architecture and the standard lies in the further subdivision of the WGs into Wavefronts (WFs). Each WF is responsible for managing one Program Counter (PC) for 64 WIs. This approach reduces the number of PCs and the complexity of the hardware without sacrificing full-thread divergence. At each clock cycle, provided that there are no pipeline stalls, each PE executes an instruction for a single WI.

The GPU@SAT IP core can be managed by the host processor through an AXI4-Lite bus control interface. It supports the loading of up to 16 OpenCL kernels, accommodating a total of 64 KB of program code. The programmability of the core is enhanced by a comprehensive software ecosystem that includes an LLVM-based compiler. This compiler translates high-level OpenCL code into assembly code executable by the accelerator, enabling high-level programmability. This design allows users to repurpose the hardware without modifying the architecture (only the software needs to be updated, making the accelerator versatile and general-purpose). Moreover, the compiler is fine-tuned to optimize the binary code specifically for the GPU@SAT IP core. This optimization is achieved through a custom LLVM backend that efficiently manages resources, such as registers, to maximize the performance of the accelerator.

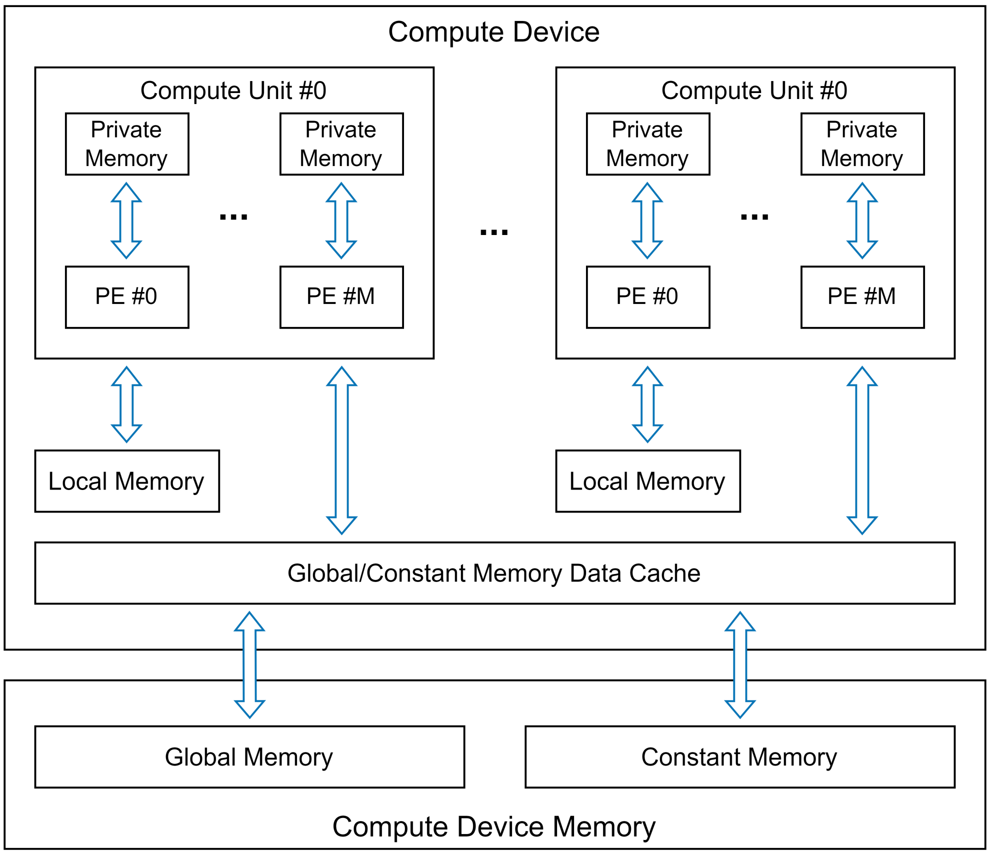

Figure 1. Structure of OpenCL device. Arrows show data exchange links between blocks.

To execute a kernel, the processor needs two types of information:

- The OpenCL properties of the kernel (such as the number of Work-Items in the kernel context).
- The executable binary code of the kernel itself.

These details must be pre-loaded by the processor into a dedicated memory space within the accelerator. Typically, this task falls upon the user, who must possess proficiency in C and OpenCL, as well as familiarity with the architecture, to instruct the host processor. Consequently, application development, particularly during the testing phase, demands a significant investment of engineering time.

3. GPU@SAT DevKit

The GPU@SAT DevKit was created to simplify and accelerate the development of applications that leverage the computational capabilities of the GPU@SAT IP core. This comprehensive hardware and software framework functions as a client–server application and consists of two key parts, as shown in Figure 2:

1. An evaluation board (*DevBoard*) that includes the GPU@SAT IP core and embedded software, which operates as the server.
2. A Python-based application running on a host PC, which operates as the client.

These modules communicate via Ethernet and work together to deliver the full capabilities of the GPU@SAT IP core while significantly reducing development time and streamlining the overall workflow. The following paragraphs provide a detailed overview of these two modules and outline the updated workflow for application execution by the user.

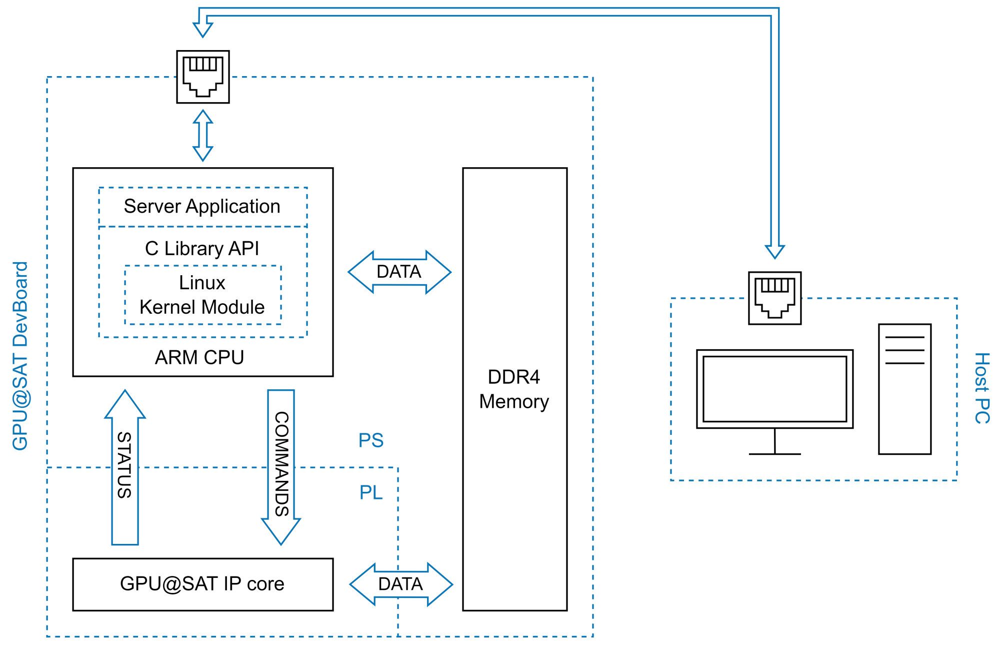

Figure 2. Overview of GPU@SAT DevKit.

3.1. GPU@SAT DevBoard and Embedded Software

The first part of the GPU@SAT DevKit is the integrated hardware and software system that composes the GPU@SAT DevBoard. Figure 2 illustrates the key elements of the system and their interconnections.

The DevBoard is based on the Zynq UltraScale+ ZCU104 evaluation platform [61], which is well suited for high-performance computing tasks. Indeed, the ZCU104 platform features a quad-core ARM Cortex-A53 Central Processing Unit (CPU), providing the necessary processing power, and a 64-bit wide Double Data Rate 4 (DDR4) memory integrated within the FPGA's Processing System (PS), ensuring efficient data handling and storage. It also includes the GPU@SAT IP core with eight CUs operating at 250 MHz, within the FPGA's PL.

The three components collaboratively form the hardware infrastructure for server operations. The CPU executes the embedded software and manages external communications to the client application via an Ethernet connection. It also coordinates the other components of the board, overseeing the accelerator and accessing the DDR4 memory. The GPU@SAT IP core handles computationally intensive tasks and can directly access memory through Direct Memory Access (DMA). Meanwhile, the memory serves as a shared data space between the CPU and the accelerator, enabling fast and efficient data exchange.

The embedded software, responsible for managing the system and providing an abstract representation of its architectural functionality, runs on the CPU and consists of three distinct parts. First, a Linux kernel module offers a unified interface for configuring the GPU@SAT IP core and managing dedicated memory spaces (*buffers*) in the DDR4 memory. Second, a high-level C library provides Application Programming Interfaces (APIs) that simplify interaction with the kernel module. Lastly, the server application handles Transmission Control Protocol (TCP) communication, allowing the DevBoard to interface with the host PC via Ethernet.

Overall, the system abstracts the functionality of the server and allows the client to perform the following tasks:

1. **Configure the GPU@SAT IP core with OpenCL properties:** this involves setting up the necessary parameters for the kernel to be executed, ensuring that the hardware is prepared to handle the specific computational tasks.
2. **Load the binary code of the kernel:** the client can upload the executable binary code corresponding to the kernel, which is then run on the GPU@SAT IP core.

3. **Allocate memory buffers in DDR4:** the client can allocate shared memory spaces in the DDR4, which are used by both the CPU and the GPU@SAT IP core.
4. **Schedule and execute multiple kernels:** the client has the capability to queue multiple kernels for execution, managing the sequence in which they are run and ensuring that dependencies between kernels are handled efficiently.
5. **Measure and report statistics and performance metrics:** the system can collect and report various statistics related to the execution of the kernels, providing valuable insights into performance and helping to identify areas for optimization.

Each of these mechanisms plays a crucial role in streamlining application execution. By configuring OpenCL properties (1) separately from the binary code (2), the same code can be reused across kernels with different configurations, minimizing memory usage and avoiding unnecessary duplication. The ability to allocate shared buffers (3) reduces the overhead associated with memory management and speeds up the execution of kernels that need to share data. Furthermore, by allowing multiple kernels to be scheduled and executed in succession (4), the system minimizes the need for continuous data transfer between the DevBoard and the host PC, significantly speeding up the processing of successive data-sharing kernels, such as those that constitute DL algorithms. Lastly, the capability to measure and report performance metrics (5) simplifies the testing and optimization of applications, making the development process more efficient and effective.

3.2. Client Application and XML Files

The second component of the GPU@SAT DevKit is the client application, a Python-based software designed to run on the host PC and facilitate communication with the GPU@SAT DevBoard via an Ethernet connection. Python was chosen for its portability, as it requires only an interpreter, making the selection of the host PC more flexible and less restrictive. This ensures that the client application can be easily deployed across various systems without the need for specialized hardware or software environments.

The client application was developed using a modular architecture, with a Python API specifically created by IngeniArs S.r.l. This API grants users comprehensive access to all the functionalities of the GPU@SAT DevBoard, allowing them to develop custom client applications tailored to their specific requirements while maintaining compatibility with the DevBoard. The modular design empowers users to adapt the client software to different use cases, enhancing the versatility of the DevKit in various development environments.

One of the primary functions of the client application is to parse eXtensible Markup Language (XML) files, which are used to extract configuration details for the DevBoard and to generate the necessary commands for TCP communication between the client and the DevBoard. To facilitate this process, the user is required to provide two specific XML files:

1. The *configuration file*: This file contains detailed information about the configuration of the DevBoard, including the setup of its components. It serves as a blueprint that guides the initialization and management of the hardware during the execution of tasks.
2. The *scheduler file*: This file defines the order in which the kernels will be executed and includes various execution-related parameters. It plays a crucial role in managing the workflow, ensuring that the kernels are run in the correct sequence and according to the specified settings.

The configuration file is divided into three sections, each defined by a specific tag. The *dataMemory* section describes all data memory buffers shared among the kernels and is structured as an array of *memRegion* objects. The *instructionMemory* section contains the code for the various kernels to be loaded, organized as an array of *codeRegion* objects. Lastly, the *descriptors* section lists the kernel descriptors to be loaded onto the GPU@SAT IP core, arranged as an array of *kernel* objects. Each element in the configuration file is assigned a unique identifier, known as *id*. These identifiers enhance code readability and allow the client to provide useful debugging information before the data are transferred to the DevBoard.

The scheduler file, on the other hand, contains the list of kernels to be executed, identified by their *id* as specified in the configuration file. Additionally, it includes tags that can enable various benchmarking features. Currently, only the *EnableTimeRecording* tag, which must be set to *true* to activate time recording for applications, is supported within the DevKit. However, the flexible structure of the client application allows for the seamless integration of additional metrics, which may offer valuable insights from the user's perspective. These metrics will be introduced in future updates.

3.3. Application Development Workflow

Figure 3 illustrates a comparison between the application development process before and after the introduction of the DevKit.

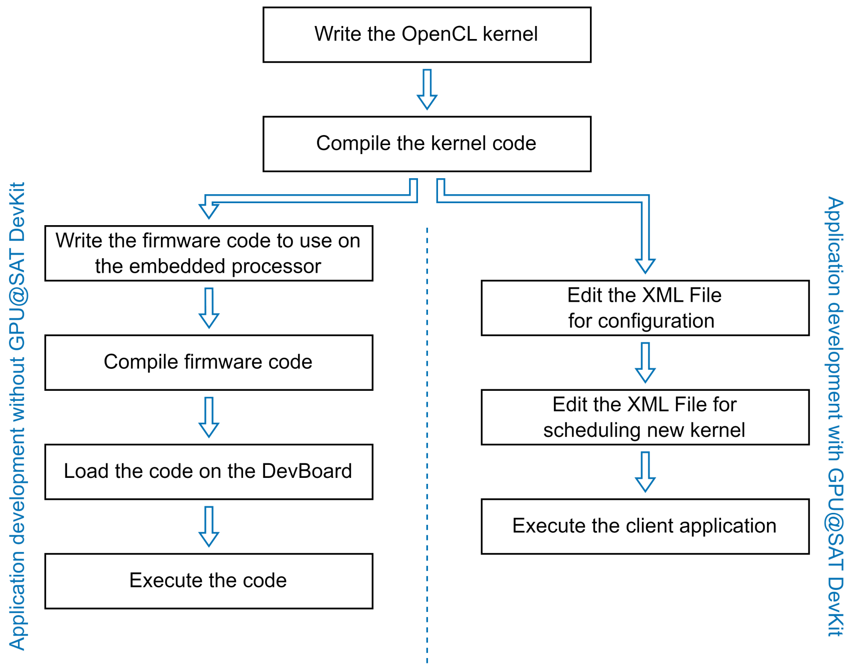
Write the OpenCL kernel
Compile the kernel code
Application development without GPU@SAT DevKit
Write the firmware code to use on the embedded processor
Compile firmware code
Load the code on the DevBoard
Execute the code
Application development with GPU@SAT DevKit
Edit the XML File for configuration
Edit the XML File for scheduling new kernel
Execute the client application

Figure 3. Application development with and without GPU@SAT DevKit.

In both scenarios, the workflow begins with writing an OpenCL kernel and compiling it to generate the binary code that aligns with the OpenCL properties of the kernel itself. Before the introduction of the DevKit, users had to manually rewrite the C firmware of the embedded processor to add commands for loading kernel information onto the accelerator. This process involved compiling the firmware, uploading it to the DevBoard, and then executing it. Any errors required restarting the entire process from the firmware rewrite, assuming the kernel itself was error-free.

The DevKit streamlines these operations significantly. Now, the user simply edits the configuration and scheduled XML files and runs the client application, which translates the files into commands for the server. This approach removes the need to manually interact with the embedded processor firmware, greatly reducing the overall application development time.

4. Results

Evaluating the benefits of the GPU@SAT DevKit in terms of development time presents a complex challenge. To effectively measure these advantages, we developed a benchmark derived from a subset of the GPU4S benchmark developed by the ESA [55]. This benchmark includes real-world applications commonly used in CV tasks and DL algorithms on resource-constrained systems in orbit, making it an ideal choice for validating the DevKit and collecting performance data for the GPU@SAT IP core. We analyzed both the time required to develop this benchmark and the expertise needed to execute the process. This approach allowed us to concretely assess the impact of the DevKit on simplifying and accelerating development workflows.

Table 1 provides a summary of the collected results. The results not only confirm the full functionality of the DevKit but also highlight its advantages for application development. The presented benchmark, consisting of six kernels across 18 configurations, required just one man-day of development time, whereas the previous workflow would have taken several man-days to complete.

Table 1. Execution times for various kernels on the GPU@SAT DevKit, measured in microseconds using the clock of the embedded processor. In these implementations, each output image pixel is consistently mapped to a single WI of the OpenCL kernel.

Name	Input Size [# Pixels]	Duration [µs]
$\text{Conv}_{3\times3}$ [1]	64×64	1125
	128×128	2243
	256×256	6172
$\text{Conv}_{5\times5}$ [2]	64×64	2804
	128×128	5051
	256×256	12,892
$\text{Conv}_{7\times7}$ [3]	64×64	3378
	128×128	6172
	256×256	23,374
Average pool	64×64	1686
	128×128	3367
	256×256	11,209
Max pool	64×64	1681
	128×128	3364
	256×256	11,212
Matrix Multiplication	64×64	2803
	128×128	15131
	256×256	118,420

[1] Convolution with a 3×3 filter. [2] Convolution with a 5×5 filter. [3] Convolution with a 7×7 filter.

The key takeaway lies in the comparison between the old and new development processes described in Section 3.3. Previously, even simple applications required hours to develop, while more complex ones demanded several man-days of work. With the DevKit, simple applications now take only a few minutes to develop, and complex ones can be completed within a few hours. Moreover, in the new workflow, the user does not need to be proficient in C coding, making application development even more user-friendly. This helps the development process, enabling a broader range of users, including those with less specialized programming skills, to contribute to the innovation and deployment of the applications.

In addition to benchmarking individual kernels, we also evaluated the DevKit's performance on a full NN inference task. Specifically, we considered the network included within the GPU4S benchmark repository, which is based on the CIFAR-10 dataset. This evaluation is crucial for assessing how the DevKit handles more complex, real-world scenarios like running entire DL models, typically used in satellite-based image processing or object detection tasks. Notably, these results align closely with those obtained for the inference of the same network on the METASAT platform [62], as detailed in [57]. While the FPGA implementation platform for METASAT differs, this comparison remains highly relevant, as both are soft-core solutions specifically designed for onboard processing in space applications. Table 2 presents the results obtained in terms of Frames Per Second (FPS) and compares them to the results of METASAT. Overall, the execution of entire networks benefits significantly from the DevKit's streamlined workflow. It is important to highlight that GPU@SAT is an IP core that can be implemented on space-grade FPGAs due to its low resource utilization, as shown in Table 3. Notably, end-to-end network inference—previously requiring considerable manual optimization and coding effort—was greatly simplified, with notable reductions in both setup time and the need for low-level firmware adjustments.

Table 2. Performance evaluation on the CIFAR-10 Neural Network implemented within the GPU4S benchmark for the GPU@SAT DevKit and METASAT.

Platform	Device	FPS
GPU@SAT DevKit	Xilinx ZCU104	35
METASAT	Xilinx VCU118	32

Table 3. Resource utilization of the GPU@SAT core, implemented with 8 CUs, on the ZCU104 board PL.

Resource	Utilization	Available	Percentage (%)
LUT	140,721	230,400	61.08
LUTRAM	10,201	101,760	10.02
FF	220,804	460,800	47.92
BRAM	312	312	100
URAM	1	96	1.04
DSP	256	1728	14.81

This does not even account for the significant time savings in debugging, which was particularly time-consuming in the old workflow due to necessary changes to the embedded processor firmware. Such a time-consuming nature of the previous process was largely due to the need for extensive manual coding and debugging, which often involved precise and detailed modifications to the embedded software. Additionally, the use of the *EnableTimeRecording* tag streamlines the collection of execution times, a process that previously, again, had to be manually embedded in the firmware code.

5. Conclusions and Future Work

The GPU@SAT DevKit represents an advanced tool in the field of edge computing for satellites, offering a robust and user-friendly ecosystem for accelerating the development of space-based IoT applications. By leveraging the high-performance GPU@SAT IP core, developers can now efficiently implement and test DL algorithms on FPGA platforms, thereby enhancing the computational capabilities onboard satellites.

The ability of the DevKit to streamline application development, reduce debugging time, and provide precise execution time measurements highlights its practical value in the Space-IoT domain.

Future development of the GPU@SAT DevKit will focus on addressing the current limitations related to the execution of kernel sequences. Currently, the DevBoard supports the execution of a fixed sequence of 16 kernels, a constraint imposed by the hardware of the GPU@SAT IP core. To fully unlock the potential of the DevKit, upcoming enhancements will focus on overcoming this limitation through targeted software improvements. These enhancements will unfold in two steps. The first step involves expanding the functionality of the embedded software, enabling the runtime allocation of descriptors and memory buffers, as well as expanding the set of metrics available to the user. In particular, activating additional profiling features would provide insights into metrics such as cache hits, misses, and other relevant data for the execution of specific kernels. This improvement will offer greater flexibility in managing computational tasks, allowing for more dynamic and complex workflows.

The second step will focus on enhancing the descriptive capabilities of the XML scheduler file. By enabling the file to accommodate a larger number of kernels, developers will have the ability to execute more intricate sequences of operations, further pushing the boundaries of what can be achieved with the tool. These developments will further extend the utility and adaptability of the GPU@SAT DevKit, making it a more versatile tool for edge computing in space.

In summary, while the GPU@SAT DevKit already offers significant benefits for satellite-based IoT applications, the planned enhancements aim to address existing limitations and further improve its utility. These developments will support more complex and flexible computational tasks onboard satellites, contributing to the advancement of edge computing in space.

Author Contributions: Conceptualization, G.B., G.G., M.D. and L.F.; methodology, G.B.; software, G.B. and M.M.; validation, G.B. and G.T.; investigation, G.B., G.T. and G.G.; writing—original draft preparation, G.B. and G.T.; writing—review and editing, G.B., G.T., G.G., M.M. and M.D.; visualization, G.T.; supervision, L.F.; project administration, M.D. All authors have read and agreed to the published version of this manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data are contained within the article.

Conflicts of Interest: Authors Gionata Benelli, Giovanni Todaro, and Gianluca Giuffrida were employed by IngeniArs S.r.l. The remaining authors declare that the research was conducted without any commercial or financial relationships that could be perceived as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
CNN	Convolutional Neural Network
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CU	Compute Unit
CV	Computer Vision
DDR4	Double Data Rate 4
DL	Deep Learning
DMA	Direct Memory Access
DNN	Deep Neural Network
DSP	Digital Signal Processing
EO	Earth Observation
ESA	European Space Agency
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GOPS	Giga Operations per Second
GPGPU	General Purpose computing on GPU
GPU	Graphic Processing Unit
IoT	Internet of Things
IP	Intellectual Property
ISL	Inter-Satellite Link
KPI	Key Performance Indicator
LEO	Low Earth Orbit
M2M	Machine-to-Machine
ML	Machine Learning
MPSoC	Multi-Processor System-on-a-Chip
NN	Neural Network
PC	Program Counter
PE	Processing Element
PL	Programmable Logic
PS	Processing System
QoS	Quality of Service
SEE	Single Event Effect
SEL	Single Event Latchup
SEU	Single Event Upset

SIMT	Single Instruction Multiple Thread
SoC	System-on-a-Chip
TCP	Transmission Control Protocol
VBN	Vision-Based Navigation
VLEO	Very Low Earth Orbit
VPU	Vision Processing Unit
WF	Wavefront
WG	Work-Group
WI	Work-Item
XML	eXtensible Markup Language

References

1. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
2. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
3. Ahmed, N.; De, D.; Hussain, I. Internet of Things (IoT) for Smart Precision Agriculture and Farming in Rural Areas. *IEEE Internet Things J.* **2018**, *5*, 4890–4899. [CrossRef]
4. Islam, S.R.; Kwak, D.; Kabir, M.H.; Hossain, M.; Kwak, K.S. The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access* **2015**, *3*, 678–708. [CrossRef]
5. Chen, X.; Ng, D.W.K.; Yu, W.; Larsson, E.G.; Al-Dhahir, N.; Schober, R. Massive Access for 5G and beyond. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 615–637. [CrossRef]
6. Transforma Insights. Current IoT Forecast Highlights. 2024. Available online: https://transformainsights.com/research/forecast/highlights (accessed on 2 May 2024).
7. Statista Research Department. Internet of Things (IoT) Connected Devices Installed Base Worldwide from 2015 to 2025. 2016. Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 2 May 2024).
8. Vailshery, L.S. Industrial Internet of Things (IIoT) Market Size Worldwide from 2020 to 2030. 2024. Available online: https://www.statista.com/statistics/611004/global-industrial-internet-of-things-market-size/ (accessed on 3 May 2024).
9. Qu, Z.; Zhang, G.; Cao, H.; Xie, J. LEO Satellite Constellation for Internet of Things. *IEEE Access* **2017**, *5*, 18391–18401. [CrossRef]
10. Kodheli, O.; Maturo, N.; Chatzinotas, S.; Andrenacci, S.; Zimmer, F. NB-IoT via LEO Satellites: An Efficient Resource Allocation Strategy for Uplink Data Transmission. *IEEE Internet Things J.* **2022**, *9*, 5094–5107. [CrossRef]
11. Narayana, S. Space Internet of Things (Space-IoT). Ph.D. Thesis, TU Delft, Delft University of Technology, Delft, The Netherlands, 2023.
12. Routray, S.K.; Tengshe, R.; Javali, A.; Sarkar, S.; Sharma, L.; Ghosh, A.D. Satellite Based IoT for Mission Critical Applications. In Proceedings of the 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, 1–2 March 2019; pp. 1–6.
13. Centenaro, M.; Costa, C.E.; Granelli, F.; Sacchi, C.; Vangelista, L. A Survey on Technologies, Standards and Open Challenges in Satellite IoT. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1693–1720. [CrossRef]
14. Iridium Communications Website. Available Online: https://www.iridium.com/ (accessed on 9 May 2024).
15. Globalstar Website. Available Online: https://www.globalstar.com/en-us/ (accessed on 9 May 2024).
16. Orbcomm Website. Available Online: https://www.orbcomm.com/ (accessed on 9 May 2024).
17. Astrocast Website. Available Online: https://www.astrocast.com/ (accessed on 10 May 2024).
18. Lacuna Space Website. Available Online: https://lacuna.space/ (accessed on 10 May 2024).
19. FOSSA Systems Website. Available Online: https://fossa.systems/ (accessed on 10 May 2024).
20. Myriota Website. Available Online: https://myriota.com/ (accessed on 10 May 2024).
21. Saeed, N.; Elzanaty, A.; Almorad, H.; Dahrouj, H.; Al-Naffouri, T.Y.; Alouini, M.S. CubeSat Communications: Recent Advances and Future Challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1839–1862. [CrossRef]
22. Hamdan, S.; Ayyash, M.; Almajali, S. Edge-Computing Architectures for Internet of Things Applications: A Survey. *Sensors* **2020**, *20*, 6441. [CrossRef]
23. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A Survey on the Edge Computing for the Internet of Things. *IEEE Access* **2018**, *6*, 6900–6919. [CrossRef]
24. Wang, Y.; Yang, J.; Guo, X.; Qu, Z. Satellite Edge Computing for the Internet of Things in Aerospace. *Sensors* **2019**, *19*, 4375. [CrossRef]
25. Wei, J.; Han, J.; Cao, S. Satellite IoT Edge Intelligent Computing: A Research on Architecture. *Electronics* **2019**, *8*, 1247. [CrossRef]
26. Song, Z.; Hao, Y.; Liu, Y.; Sun, X. Energy-Efficient Multiaccess Edge Computing for Terrestrial-Satellite Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 14202–14218. [CrossRef]
27. Li, C.; Zhang, Y.; Xie, R.; Hao, X.; Huang, T. Integrating Edge Computing into Low Earth Orbit Satellite Networks: Architecture and Prototype. *IEEE Access* **2021**, *9*, 39126–39137. [CrossRef]
28. Kim, T.; Kwak, J.; Choi, J.P. Satellite Edge Computing Architecture and Network Slice Scheduling for IoT Support. *IEEE Internet Things J.* **2022**, *9*, 14938–14951. [CrossRef]
29. Kua, J.; Loke, S.W.; Arora, C.; Fernando, N.; Ranaweera, C. Internet of Things in Space: A Review of Opportunities and Challenges from Satellite-Aided Computing to Digitally-Enhanced Space Living. *Sensors* **2021**, *21*, 8117. [CrossRef] [PubMed]

30. Garcia, L.P.; Furano, G.; Ghiglione, M.; Zancan, V.; Imbembo, E.; Ilioudis, C.; Clemente, C.; Trucco, P. Advancements in On-Board Processing of Synthetic Aperture Radar (SAR) Data: Enhancing Efficiency and Real-Time Capabilities. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2024**, *17*, 16625–16645. [CrossRef]
31. Kothari, V.; Liberis, E.; Lane, N.D. The Final Frontier: Deep Learning in Space. In Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications, New York, NY, USA, 3–4 March 2020; pp. 45–49.
32. Furano, G.; Meoni, G.; Dunne, A.; Moloney, D.; Ferlet-Cavrois, V.; Tavoularis, A.; Byrne, J.; Buckley, L.; Psarakis, M.; Voss, K.O.; et al. Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities. *IEEE Aerosp. Electron. Syst. Mag.* **2020**, *35*, 44–56. [CrossRef]
33. Fourati, F.; Alouini, M.S. Artificial intelligence for satellite communication: A review. *Intell. Converg. Netw.* **2021**, *2*, 213–243. [CrossRef]
34. Ortiz, F.; Monzon Baeza, V.; Garces-Socarras, L.M.; Vásquez-Peralvo, J.A.; Gonzalez, J.L.; Fontanesi, G.; Lagunas, E.; Querol, J.; Chatzinotas, S. Onboard Processing in Satellite Communications Using AI Accelerators. *Aerospace* **2023**, *10*, 101. [CrossRef]
35. Chintalapati, B.; Precht, A.; Hanra, S.; Laufer, R.; Liwicki, M.; Eickhoff, J. Opportunities and challenges of on-board AI-based image recognition for small satellite Earth observation missions. *Adv. Space Res.* **2024**. [CrossRef]
36. Wei, J.; Cao, S. Application of Edge Intelligent Computing in Satellite Internet of Things. In Proceedings of the 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), Tianjin, China, 9–11 August 2019; pp. 85–91.
37. Gaillard, R. Single Event Effects: Mechanisms and Classification. In *Soft Errors in Modern Electronic Systems*; Nicolaidis, M., Ed.; Springer: Boston, MA, USA, 2011; pp. 27–54.
38. Quinn, H. Radiation effects in reconfigurable FPGAs. *Semicond. Sci. Technol.* **2017**, *32*, 044001. [CrossRef]
39. Rockett, L.; Patel, D.; Danziger, S.; Cronquist, B.; Wang, J. Radiation Hardened FPGA Technology for Space Applications. In Proceedings of the 2007 IEEE Aerospace Conference, Big Sky, MT, USA, 3–10 March 2007; pp. 1–7.
40. Cong, J.; Sarkar, V.; Reinman, G.; Bui, A. Customizable domain-specific computing. *IEEE Des. Test Comput.* **2010**, *28*, 6–15. [CrossRef]
41. Ghiglione, M.; Serra, V. Opportunities and challenges of AI on satellite processing units. In Proceedings of the 19th ACM International Conference on Computing Frontiers (CF'22) , New York, NY, USA, 17–22 May 2022; pp. 221–224.
42. Todaro, G.; Monopoli, M.; Benelli, G.; Zulberti, L.; Pacini, T. Enhanced Soft GPU Architecture for FPGAs. In Proceedings of the 2023 18th Conference on Ph.D Research in Microelectronics and Electronics (PRIME), Valencia, Spain, 18–21 June 2023; pp. 177–180.
43. Monopoli, M.; Zulberti, L.; Todaro, G.; Nannipieri, P.; Fanucci, L. Exploiting FPGA Dynamic Partial Reconfiguration for a Soft GPU-based System-on-Chip. In Proceedings of the 2023 18th Conference on Ph.D Research in Microelectronics and Electronics (PRIME), Valencia, Spain, 18–21 June 2023, pp. 181–184.
44. Lemaire, E.; Moretti, M.; Daniel, L.; Miramond, B.; Millet, P.; Feresin, F.; Bilavarn, S. An FPGA-Based Hybrid Neural Network Accelerator for Embedded Satellite Image Classification. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
45. OPS-SAT Satelite Website. Available Online: https://www.eoportal.org/satellite-missions/ops-sat (accessed on 10 May 2024).
46. Furano, G.; Tavoularis, A.; Rovatti, M. AI in space: Applications examples and challenges. In Proceedings of the 2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Frascati, Italy, 19–21 October 2020; pp. 1–6.
47. Cosmas, K.; Kenichi, A. Utilization of FPGA for Onboard Inference of Landmark Localization in CNN-Based Spacecraft Pose Estimation. *Aerospace* **2020**, *7*, 159. [CrossRef]
48. Leon, V.; Stamoulias, I.; Lentaris, G.; Soudris, D.; Gonzalez-Arjona, D.; Domingo, R.; Codinachs, D.M.; Conway, I. Development and Testing on the European Space-Grade BRAVE FPGAs: Evaluation of NG-Large Using High-Performance DSP Benchmarks. *IEEE Access* **2021**, *9*, 131877–131892. [CrossRef]
49. Rapuano, E.; Meoni, G.; Pacini, T.; Dinelli, G.; Furano, G.; Giuffrida, G.; Fanucci, L. An FPGA-Based Hardware Accelerator for CNNs Inference on Board Satellites: Benchmarking with Myriad 2-Based Solution for the CloudScout Case Study. *Remote Sens.* **2021**, *13*, 1518. [CrossRef]
50. Giuffrida, G.; Diana, L.; de Gioia, F.; Benelli, G.; Meoni, G.; Donati, M.; Fanucci, L. CloudScout: A deep neural network for on-board cloud detection on hyperspectral images. *Remote Sens.* **2020**, *12*, 2205. [CrossRef]
51. Pitonak, R.; Mucha, J.; Dobis, L.; Javorka, M.; Marusin, M. CloudSatNet-1: FPGA-Based Hardware-Accelerated Quantized CNN for Satellite On-Board Cloud Coverage Classification. *Remote Sens.* **2022**, *14*, 3180. [CrossRef]
52. Hu, Y.; Liu, Y.; Liu, Z. A Survey on Convolutional Neural Network Accelerators: GPU, FPGA and ASIC. In Proceedings of the 2022 14th International Conference on Computer Research and Development (ICCRD), Shenzhen, China, 7–9 January 2022; pp. 100–107.
53. Xu, C.; Jiang, S.; Luo, G.; Sun, G.; An, N.; Huang, G.; Liu, X. The Case for FPGA-Based Edge Computing. *IEEE Trans. Mob. Comput.* **2022**, *21*, 2610–2619. [CrossRef]
54. Powell, W.A.; Campola, M.J.; Sheets, T.; Davidson, A.; Welsh, S. *Commercial Off-the-Shelf GPU Qualification for Space Applications*; National Aeronautics and Space Administration: Washington, DC, USA, 2018.
55. Kosmidis, L.; Lachaize, J.; Abella, J.; Notebaert, O.; Cazorla, F.J.; Steenari, D. GPU4S: Embedded GPUs in Space. In Proceedings of the 2019 22nd Euromicro Conference on Digital System Design (DSD), Kallithea, Greece, 28–30 August 2019; pp. 399–405.
56. Rodriguez, I.; Kosmidis, L.; Notebaert, O.; Cazorla, F.J.; Steenari, D. An On-board Algorithm Implementation on an Embedded GPU: A Space Case Study. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 1718–1719.

57. Solé, M.; Wolf, J.; Rodriguez, I.; Jover, A.; Trompouki, M.M.; Kosmidis, L.; Steenari, D. Evaluation of the Multicore Performance Capabilities of the Next Generation Flight Computers. In Proceedings of the 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), Barcelona, Spain, 1–5 October 2023; pp. 1–10.
58. Bersuker, G.; Mason, M.; Jones, K.L. Neuromorphic Computing: The Potential for High-Performance Processing in Space. The Aerospace Corporation's Center for Space Policy and Strategy. 2018. Available Online: https://csps.aerospace.org (accessed on 1 October 2024).
59. Izzo, D.; Hadjiivanov, A.; Dold, D.; Meoni, G.; Blazquez, E. Neuromorphic computing and sensing in space. In *Artificial Intelligence for Space: AI4SPACE*; CRC Press: Boca Raton, FL, USA, 2022; pp. 107–159.
60. Munshi, A.; Gaster, B.; Mattson, T.G.; Ginsburg, D. In *OpenCL Programming Guide*; Pearson Education: London, UK, 2011.
61. AMD. ZCU104 Evaluation Board-User Guide v1.1. Available online: https://docs.amd.com/v/u/en-US/ug1267-zcu104-eval-bd (accessed on accessed on 1 October 2024).
62. Kosmidis, L.; Solé, M.; Rodriguez, I.; Wolf, J.; Trompouki, M.M. The METASAT Hardware Platform: A High-Performance Multicore, AI SIMD and GPU RISC-V Platform for On-board Processing. In Proceedings of the 2023 European Data Handling & Data Processing Conference (EDHPC), Juan Les Pins, France, 2–6 October 2023; pp. 1–6.