MANAGING THE DATA FLOOD - ANALYZING IMAGES ONBOARD SMALL EARTH OBSERVATION SATELLITES

Moritz P. Heimbach*, Pawan Kumar*, Shreeja Shrijan Shrestha*, Patrick Glöckner*, Markus Plattner†, Chedi Fassi†, Marco Schmidt*

* Julius-Maximilians-Universität Würzburg, ESSEO, Würzburg, Germany
† Engineering Minds Munich GmbH, Munich, Germany

Abstract

Satellites as small as cubesats and other Nanosatellites can generate vast amounts of optical data using high-resolution and multispectral image sensors. Coupled with the growing popularity of proliferated earth observation constellations consisting of many small satellites instead of larger, more traditional earth observation satellites, this presents new challenges.
Firstly, this data has to be downlinked from the satellites to ground stations, a task which is made more complicated with increasing numbers of satellites in a constellation. Additionally, smaller satellites tend to have less power available for transmitting, as well as smaller antennas, reducing downlink speeds. Secondly, the data must be analyzed to extract mission-specific information depending on the current use-case: For example, wildfires have to be identified or the location and type of ships detected. For many applications, only these extracted meta-information are relevant, not the actual satellite image itself. These necessary steps of downlinking and image analysis can lead to large delays between image acquisition and actionable decisions being made on the basis of this data.
In this paper we describe the integration of an onboard image analysis payload into the UWE-5 satellite mission, an educational communications satellite mission consisting of two 3U CubeSats. This processing unit features two boards, each containing a Microchip PolarFire MPFS250T FPGA SoC. In this architecture, one FPGA SoC controls the camera payload, while the second one synthesizes a Machine Learning (ML) accelerator to analyze camera data onboard the satellite using Convolutional Neural Networks (CNNs). This approach enables the satellite to evaluate the usefulness of specific images by analyzing the degree of cloud cover or to detect features deemed important by the operator, accelerating the path from data acquisition to actionable insight and reducing data volume.

Keywords
Onboard Image Processing; Edge AI; Machine Learning in Space

Acronyms

- COTS Commercial Off The Shelf
- ESA European Space Agency
- ESSEO Embedded Systems and Sensors for Earth Observation
- FPGA Field-Programmable Gate Array
- HAL Hardware Abstraction Layer
- ISL Inter Satellite Link
- LE Logic Element
- ML Machine Learning
- MSS Microprocessor Subsystem
- OBC On Board Computer
- RTOS Real-Time Operating System
- SoC System-on-Chip
- SPPM Smart Processing and Power Module
- UWE Unversity of Würzburg Experimental Satellite

1. INTRODUCTION

In this paper we describe the integration of an onboard image analysis payload into a 3U CubeSat. The necessity for such processing platforms arises from the rapidly increasing numbers of earth observation platforms as small as nano- and microsatellites, each equipped with high resolution multispectral cameras and generating large amounts of data. Examples for this are the Planet Labs Dove constellation of 3U Cubesats, consisting of over 150 satellites[1], the Satellogic Aleph-1 constellation[2] consisting of 52 satellites measuring $51 \times 57 \times 82\,\text{cm}$ or a constellation of Lemur-Cubesats operated by Spire Global[3]

[1] https://www.planet.com/our-constellations/ (last accessed 2025/08/17)
[2] https://satellogic.com/technology/constellation/ (last accessed 2025/08/17)
[3] https://spire.com/space-services/lemur-space-platform/ (last accessed 2025/08/18)

doi: 10.25967/650390

With the traditional approach for Earth observation, the optical data generated by these satellites has to be downlinked to terrestrial base stations before being analyzed for mission-specific information such as the presence and location of wildfires, ships or planes. This poses multiple challenges: Small satellites tend to offer slower downlink speeds, increasing the time required to downlink high-resolution optical data. Additionally, with increasing numbers of satellites are deployed in a constellation, either additional groundstations are required, or the contact periods allocated to each satellite have to be reduced. In order to mitigate this, companies such as Spire Global started adding ISLs to their satellites, enabling the forwarding of data through the constellation between satellites until it can be sent to ground by a satellite passing over a base station.
Once received by the base station, the images are analyzed by humans or machines: In the first step, cloud cover and other weather conditions can render images useless for most purposes, leading to them being immediately discarded. Additionally, if the purpose is not general Earth observation such as in the Landsat- or Sentinel-series of satellites, only images containing specific features are relevant. For example an image not containing ships will be discarded, if the operator is focused on the tracking of ships. In the latter case, information like position, size, type and heading of vessels will be extracted, but it is not necessary to preserve the complete optical image.
Since in these cases mostly the meta-information extracted from the images are relevant and not every individual image, it can be beneficial to perform the image analysis already onboard the satellite, offering multiple advantages: First, images that are unusable due to cloud coverage or other adverse weather effects can be discarded immediately, freeing up limited downlink capacity. Second, images can be pre-selected or assigned downlink priority based on the satellite's mission or alternatively, the extracted meta-information such as positions of ships or wildfires can be downlinked directly, drastically reducing transmission requirements. Additionally, enhancing the onboard autonomy capabilities of spacecraft both in orbits and landed on planets is desirable, particularly for those with limited communication capabilities and for mobile systems such as rovers.
In this paper, we describe the design of a new edge AI processing unit, its expected performance its integration into a 3U Cubesat for validation. Section 2 gives an overview over the current state of the art in onboard processing of payload data aboard satellites. The physical architecture of the proposed processing unit and its integration into a satellite, as well as the software architecture and integration, are described in section 3. Section 4 and section 5 contain information on the structure of the ML architecture and the deployment of custom ML models, respectively. An outlook on future work is provided in section 6.

2. STATE OF THE ART

Onboard data processing in spacecraft is growing in importance, particularly focussing on processing visual image data. Here, ML techniques are used for various purposes, such as the detection of clouds in hyperspectral images [1] and the identification of wildfires [2]. Additional applications include calculating the electrical energy required by a rover to move to a target destination [3] or the localization of planetary rovers after landing [4]. In order to run these ML models newer, more powerful onboard processing platforms are required. These can either be COTS-systems or purpose-designed onboard processing units. Several companies started to either include onboard image analysis hardware and software in their existing product portfolios, or were founded with the specific purpose of providing such products and services.
One company specializing in Spacecraft onboard computing is Ubotica Technologies with their line of CogniSAT processing platforms. These platforms follow the PC104 form-factor and are built around the Intel Movidius Myriad Vision Processing Units [5], a line of dedicated processors for ML-based image processing. The complete boards have a typical power consumption of 2 W for image processing tasks, with peak power consumption below 5 W. Deployed in the CogniSat-6 satellite, the CogniSat-XE2 processing unit has a peak power consumption of about 3.5 W while detecting ships in multispectral images with a low-power standby consumption of 15 mW.
A CogniSat-XE1 processing unit is also deployed on the Φsat-2 mission [6] to deploy and test ML algorithms in space. Developed by ESA with the experience from Φsat-1, which validated onboard cloud detection, Φsat-2 aims to deploy various so-called AI-Apps in-orbit to verify the applicability and usefulness of ML applications onboard EO satellites[4]. These AI-Apps are detailed in [7]. One such application is the Sat2Map-App, developed by CGI, that uses a Cycle-Consistent Generative Adversarial Network to automatically generate maps of accessible roads from satellite images. Next is a Maritime Vessel Detection and Classification APP developed by CEiiA, as well as an application for the detection of anomalies in marine ecosystems by IRT Saint Exupéry Technological Research Institute. The latter detects incidents such as oil spills, algae blooms and sediment flooding in real time. GEO-K provides a Deep Image Compression App, which compresses images onboard the satellite using ML-techniques to reduce the size for downlink, while enabling reconstruction on the ground without the loss of relevant information. An application for the detection of wildfires - PhiFire AI developed by Thales Alenia Space - classifies regions into fire, safe, burnt and water zones. Lastly, an App for Cloud Detection as a service is provided by KP Labs. This can be used by other apps to automatically discard images with

[4] https://www.open-cosmos.com/news/phisat-2-launch (last accessed 2025/08/29)

high percentage of cloud cover in order to reduce the necessary downlink capacity.
KP Labs was founded with the specific goal of supporting onboard data processing and analysis on satellites. The company offers different onboard processing platforms for general data processing and for the execution of ML algorithms: The Antelope DPU has the PC104 form factor, 8 GiB of DDR4 RAM, 4 GB or 8 GB SLC NAND Flash and a Zynq UltraScale+ MPSoC containing Quad-Core ARM Cortex A53CPU and varying sizes of FPGAs. The supply voltage of the Antelope DPU can lie between 5 V and 14 V. The FPGA can be configured to perform data processing or run ML applications, with the ability for in-orbit FPGA reconfiguration.
The Leopard-DPU contains a Zynq UltraScale+ ZU6EG to ZU15EG MPSoC, containing the same CPU as the Antelope-DPU, but with a larger FPGA. DDR4 RAM and SLC Flash-based file system storage are both available from 4 GB to 16 GiB, with up to 512 GiB SLCC Flash for data storage. According to [8], up to 3 TOPS of ML inference performance can be achieved by the DPU at a power consumption between 7.5 W and 40 W, depending on the workload and processing speed.
The NVidia Jetson series of embedded computers is also used for onboard ML inference and training, such as the Jetson Xavier NX deployed on the Sonate-2 satellite [9] or the Jetson TX2i deployed on NewSat-satellites by Satellogic and Palantir Technologies[5]. Additionally, Jetsons in space-capable housings are being distrubuted by Aitech Systems[6], however the Nvidia Jetson modules themselves are not certified for the space environment and require radiation shielding.
SkyServe inc provides onboard data processing services for cloud segmentation, haze detection and selection of high-priority images utilizing the Unibap iX5 processing unit[7] built around the same Intel Movidius Myriad X VPU also deployed by Ubotica Technologies [10].

3. PAYLOAD ARCHITECTURE

This section details first the hardware architecture of the proposed processing unit, followed by the software architecture.

3.1. Hardware Architecture

The novel processing unit described in this paper - the Smart Processing and Power Module (SPPM) - consists of two stacked boards with the 70×70 mm UNISEC form factor[8], connected via a 160-pin application interface connector. The development of the individual board architecture has been described in [11], with the corresponding power supply being described in [12]. Each board contains a Microchip Technology PolarFire MPFS250T FPGA SoC, incorporating a quad-core RISC-V processor, as well as a reconfigurable FPGA. All products in the PolarFire SoC FPGA series contain the same Microprocessor and only differ in the size of the available LEs in the FPGA, counting between 23 k and 461 k LEs. The chosen MPFS250T contains 250 k LEs, being identical to the processor in the PolarFire Icicle Kit development board.
The maximum thickness of the SPPM stack is 27 mm. Figure 1 shows the stacked processing unit. To achieve modularity and fault isolation, each processing element is assigned a specific subset of functions. In this configuration, one board (SPPM1) provides basic functionality found also in traditional EO-satellites without onboard image processing, such as communication with the satellite bus, camera control and image storage. The processor on the second board (SPPM2) performs the onboard image analysis by synthesizing an ML accelerator in its FPGA. The reason for distributing these functionalities over two SoCs is to separate the more experimental image analysis step from the regular camera operation. Additionally, this way the FPGA-based ML accelerator in SPPM2 can be reconfigured in-orbit without interrupting the regular camera use.

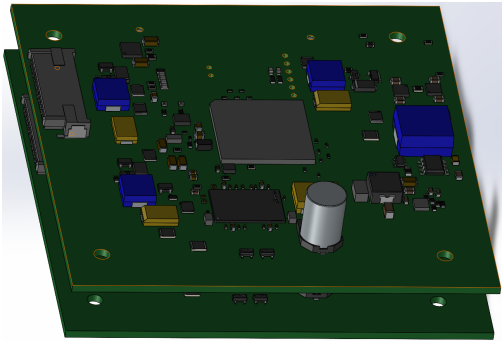

Fig. 1. Stacked Smart Processing and Power Module (SPPM). Both boards contain identical components and architectures.

To connect SPPM1 to the satellite bus and to attach instruments such as cameras, RS-485 and UART communication interfaces are available. Due to the flexibility of the FPGA fabric, additional interfaces required by the desired payload can be synthesized as needed. The maximum power consumption of a single MPFS250T is specified as 12 W[9], with the expected power consumption for SPPM1 being about 3 W for general functionality and below 5 W for SPPM2 during image processing. The input voltage can range from 5 V to 12 V and will be fixed at 5 V for the integration into the UWE-5 satellite[10] in development at University of Würzburg and BTU Cottbus.

[5] https://blog.palantir.com/edge-ai-in-space-93d793433a1e (last accessed 2025/08/25)
[6] https://aitechsystems.com/space-products/space-gpgpu/ (last accessed 2025/09/07)
[7] https://unibap.com/solutions/hardware/ix5/ (last accessed 2025/09/07)
[8] http://unisec-europe.eu/ (last accessed 2025/06/20)
[9] https://www.sundancedsp.com/in-design/polarberry-fpga-module-with-risc-processors/ (last accessed 2025/09/03)
[10] https://www.informatik.uni-wuerzburg.de/uwe5 (last accessed 2025/09/05)

The MVP Aerospace KissCAM series[11] of small satellite cameras is being considered for validation. The KissCAM Pro offers a resolution of up to 1920×1080 pixel with an M12 lens mount at a power consumption below 500 mW. Assuming an orbital height of 600 km and a telephoto lens with an opening angle of 22°, a ground sample distance of about 216 m can be achieved. This resolution is sufficient for testing cloud segmentation aboard the satellite on state- and country-scale or to detect features such as glaciers.

3.2. Software Architecture

The MSS of the PolarFire SoC allows the deployment of Linux, bare-metal C-code and/or RTOS-applications on the same chip, featuring an E51 monitoring core and four U54 application cores. Figure 2 shows the previously mentioned distribution of tasks between SPPM1 and SPPM2. To execute the respective functionalities in the MSS of both SPPM1 and SPPM2, the choice between using Linux, bare-metal C-code or an RTOS like FreeRTOS had to be made. Since the UWE-5 Satellite utilizes the KronOS middleware developed by the ESSEO working group at the University of Würzburg, it was decided to port KronOS to the PolarFire SOC. KronOS is a fault-resistant and real-time capable communications middleware utilizing the COMPASS protocol already deployed on the preceding UWE-3 and UWE-4 satellites [13]. It provides a streamlined API for onboard communication and process management, building atop the POSIX interface. KronOS allows easy cross-compilation and the adaptation of new HALs for different processors and compilers. The layered, HAL-based architecture of KronOS can be seen in Figure 3. Using POSIX allows embedded targets using the POSIX compatibility layer of FreeRTOS as well as any POSIX compliant operating system to join the KronOS network.
The four application cores of the MSS can either run Linux on all cores, or only execute Linux on one to three cores and use the remaining core(s) in different ways. These can either run an RTOS such as FreeRTOS or execute Bare Metal C-code[12]. Linux can utilize multiple cores with symmetric multiprocessing, while an RTOS can only be run as asymmetric multiprocessing. Communication between cores running Linux, RTOS and Bare Metal is possible using the Remote Processor Messaging protocol[13]. Since the EO applications running on the SPPM do not require predictable latency behaviour, no requirement for real-time capability exists. Should the requirements change at a later stage of the project, one of the cores running Linux can be changed to run different software. As shown in Figure 2, Linux will distribute tasks over all four cores to communicate with the satellite bus and the OBC, control the camera and perform data management. Additionally, if the SPPM is commanded to perform image analysis, SPPM1 will trigger SPPM2 to perform the analysis, after which SPPM1 will process the result. Depending on mission priorities, SPPM1 can then decide to downlink or discard an image or to downlink only sections containing geolocated points of interest.
To use Linux, a POSIX compliant implementation of the KronOS HAL was written, allowing the use of serial communication devices, file transfer and storage and the internal clock. This is not only useful onboard SPPM1, but allows any POSIX compliant hosts to join the KronOS network.
Commanding of the SPPM will be handled by the OBC, communicating via KronOS. KronOS implements a Command API which allows execution of commands on remote targets, identified through a unique ID. Routing through multiple subsystems, e.g. uplink by a telecommunications payload, routing through the OBC and reception by the camera subsystem is all handled by KronOS internally. Once received by the target system, the Command API forwards the command to an application specific service, in this case the Camera API. The Camera API on SPPM1 will handle the incoming command, and e.g. trigger the camera to take a picture or start the post-processing in coordination with SPPM2. Coordination between SPPM1 and SPPM2 will also be done through KronOS allowing other participants like ground stations to monitor or intervene, whereas the image data will be exchanged using a faster Serializer/Deserializer pair between both FPGAs.
The described command chain for the camera has already been tested between a ground station computer and the SPPM1 attached via a serial connection and worked as expected on development boards acquired from Microchip. This test setup will be integrated in a bench-top test with the OBC and additional components pending their readiness.
To change the architecture of the ML accelerator synthesized in SPPM2, bitstreams containing different architectures can be stored in non-volatile memory aboard SPPM2 either at launch or throughout the mission. This way, accelerators better suited for the respective mission priorities can be deployed, optimizing parameters such as inference time or power consumption, without affecting the regular camera operation and acquisition of images. Trained ML models for various applications can also be stored in the non-volatile memory and uploaded throughout the mission, before being loaded by the ML accelerator.

4. ML INFERENCE ARCHITECTURE

To perform inference, an ML accelerator must be synthesized in the FPGA of SPPM2. Microchip Technolo-

[11] https://mvpaerospace.com/products/ (last accessed 2025/08/26)
[12] https://github.com/polarfire-soc/polarfire-soc-documentation/blob/master/applications-and-demos/asymmetric-multiprocessing/amp.md (last accessed 2025/08/26)
[13] https://github.com/polarfire-soc/polarfire-soc-documentation/blob/master/applications-and-demos/asymmetric-multiprocessing/rpmsg.md (last accessed 2025/08/26)

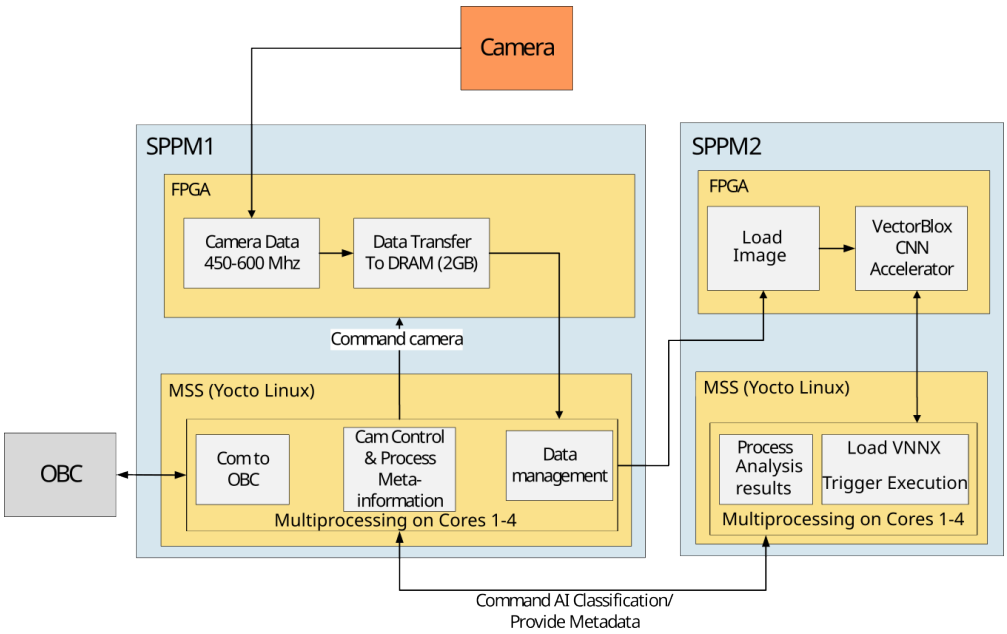

Fig. 2. High-Level architecture and dataflow inside the SPPM. In the Microprocessor Subsystems of both SPPMs, Linux performs multiprocessing on all cores.

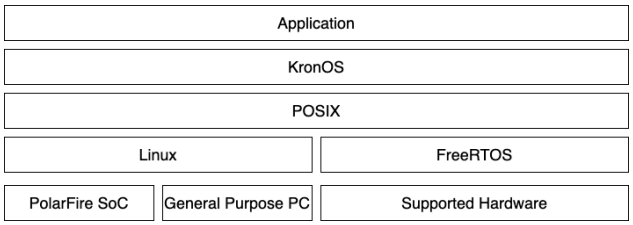

Fig. 3. Layered software architecture deployed on the SPPM. The application utilizes standardized communication interfaces provided by KronOS, which in turn uses the POSIX-interface to build upon FreeRTOS or Linux.

gies provides the VectorBlox Accelerator SDK[14] which will be used for image processing in a first step. The VectorBlox accelerator is a matrix processor supporting 8-bit integer (INT8) operations. The accelerator is available in three different sizes and performance classes: V250, V500 and V1000. The architecture consists of a parallel processor, called MXP Vector Processor, that executes data-parallel operations on vectors of data, as well as a CNN Accelerator containing a 2D-array of multiply-accumulate units in order to leverage the high level of parallelism in convolutional neural networks. The performance classes influence the size of the respective processing units, i.e. the the width of the vector processor and the side-lengths of the CNN Accelerator array. The expected inference performance levels of the VectorBlox accelerator will be discussed in Section 5. As mentioned in Section 3.2, in the future further accelerators will be synthesized, enabling to choose accelerators optimized for latency, power consumption or flexibility bsed on current priorities. Additionally, parameters such as the input image resolution can be dynamically adjusted to prioritize either accuracy or inference time.

[14] https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/vectorblox (last accessed 2025/06/20)

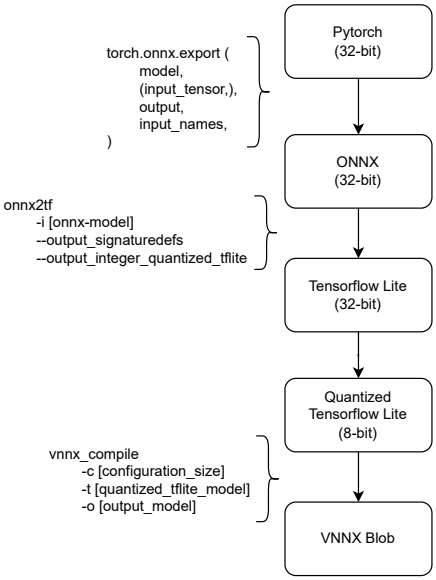

Fig. 4. Conversion of 32-bit floating point Pytorch Model to 8-bit integer quantized VNNX using Vectorblox SDK

To perform inference with trained ML models, these are required to have the form of binary VNNX blobs, a datatype created specifically to contain VectorBlox models. The VectorBlox SDK provides the necessary toolchain to generate these blobs, which can either be executed in a simulator on the development workstation or be deployed directly to VectorBlox hardware.

TAB 1. Technical specifications and peak performance for select inference processors

System	Operations Per Second	System information	Power
VectorBlox V250	78 int8 GOPs	154 MHz, 104 Math Blocks	<5 W
VectorBlox V500	152 int8 GOPs	148 MHz, 204 Math Blocks	<5 W
VectorBlox V1000	284 int8 GOPs	138 MHz, 332 math blocks	<5 W
Raspberry Pi4 [14]	6.9 int GOPs / 3.3 float GOPs	1 GB - 8 GB RAM	<15 W
Raspberry Pi5 [15]	14.6 int GOPs / 11.3 float GOPs	2 GB - 16 GB RAM	<25 W
Jetson TX2i [16]	1.33 FP16 TFLOPS	8 GB RAM	10 – 20 W
Jetson Xavier NX [17]	21 TOPS	8 GB RAM	10 – 20 W
Jetson Orin NX16 [18]	157 TOPS	16 GB RAM	10 – 40 W
Myriad X VPU [19]	>1TOPS inference		2.5 W

According to the VectorBlox Programmer's Guide[15], four conversion paths are supported, all of which utilize TensorFlow Lite (TFLite) as the intermediate format.
A graph showing out conversion pipeline can be seen in figure 4. In our experiments, the model was trained in Pytorch with 32-bit quantization, after which it was exported to ONNX (32-bit quantization) using the torch.onnx[16] module. The ONNX model was then converted to TFLite (32-bit quantization) via the onnx2tf[17] tool. Since the vnnx_compile utility of Vectorblox SDK requires an INT8-quantized TFLite model as input, the network was quantized to produce an 8-bit integer TFLite model. Finally, the quantized model was compiled into VNNX blobs, whereby each possible CoreVectorBlox configuration (V250 / V500 / V1000) requires a specific blob.
In order to be executed in the PolarFire accelerator, VNNX blobs compiled for the present configuration must be stored in the non-volatile menory of the SPPM. A device tree overlay is created to add a VectorBlox instance, enabling the Linux kernel to recognize the hardware and expose it as a device node. The VectorBlox SDK provides APIs for interacting with VectorBlox IP, enabling various tasks, such as allocating Direct Memory Access (DMA) buffers, initializing the VectorBlox IP Core, and running the model. Using these APIs, a script is developed to run in Linux on the MSS. The script initializes the VectorBlox instance and triggers the loading of the VNNX blob into the accelerator. It also ensures that input images for the network are placed in the correct address in the DDR memory buffer. Through DMA, the Accelerator retrieves the input images from these buffers, performs inference, and places the output image or classification vector at a predefined DDR address buffer without further interaction with the MSS.

[15] https://github.com/Microchip-Vectorblox/VectorBlox-SDK/blob/master/docs/VectorBloxPG.pdf (last accessed 2025/09/06)
[16] https://docs.pytorch.org/docs/stable/onnx.html (last accessed 2025/09/06)
[17] https://github.com/PINTO0309/onnx2tf (last accessed 2025/09/06)

5. DEPLOYMENT OF CLOUD SEGMENTATION MODEL

The expected ML inference performance, as well as the accompanying power consumption, depends on the specific architecture of the ML accelerator synthesized in the FPGA. As mentioned in Section 4, at first we only assume the implementation of Microchip's own VectorBlox accelerator. The Peak performance of the VectorBlox accelerators according to the HB0919 CoreVectorBlox Handbook Revision 2.0 can be seen in table 1 and have also been analyzed in [20]. In [21], the implementation and testing of a custom reconfigurable deep learning accelerator in the PolarFire SOC is described.
Table 1 also shows a selection of processors and computers used for inference and onboard processing in spacecraft. Here, the column denoting the operations per second corresponds to a theoretical, optimal case without any bottlenecks and with all computing units at maximum utilization. In addition, considering both regular compute units and specific SIMD-extensions to the CPU - such as NEON for ARM processors - can be difficult and individual multiply-accumulate operations are sometimes counted as one operation and sometimes as two separate operations. Thus, these theoretical values serve only to give a rough overview over the expected performance of a processor or system relative to the others.
In order to compare these theoretical performance values with experimental data, preliminary experiments for the real-world use-case of cloud segmentation in satellite images were performed. To this end, a U-Net like convolutional neural network was built in Pytorch, taking in images in the red-, blue-, green-, and near-infrared bands with a single-band cloud mask as the output. The network consists of four encoding layers that reduce the image resolution, while increasing the number of feature maps, and four decoding layers, that in turn increase the resolution and decrease the number of feature maps, until a single image with an identical resolution to the input image resolution is returned. Table 2 shows the necessary Giga-Floating point operations necessary to infer a single cloud mask with 192×192 or 384×384 pixel resolution, according to the fvcore- and THOP-libraries. Discrepancies

TAB 2. Necessary floating point operations for inference with the deployed cloud segmentation model, according to python fvcore and THOP libraries

Library	Image Resolution	GFLOPs
fvcore	192×192	6.884
fvcore	384×384	27.538
THOP	192×192	7.813
THOP	384×384	31.251

TAB 3. Average inference times and power consumption over 9201 inferences for resolution 192×192 on the 38-cloud dataset [22]. Idle power draw of Raspberry Pi 4 is 2.3 W, Pi 5 2.8 W, PolarFire Icicle Kit with Linux running and VectorBlox synthesized in the MPFS250T 4.6 W, Jetson Orin NX 16GB 8.3 W. The Nvidia Jetson was set to 10 W mode. Deployment on the Microchip Icicle kit development board is still work in progress.

System	Quantization	Inference	Power
R Pi4	Float32	0.64 s	6.3 W
R Pi4	Int8	2.89 s	5.2 W
R Pi5	Float32	0.40 s	10.2 W
R Pi5	Int8	0.84 s	9.6 W
Orin NX16	Float32 CPU	0.32 s	9.6 W
Orin NX16	INT8 CPU	1.16 s	9.8 W
Icicle Kit	INT8		

between the results of both libraries are due to their respective layer support and method of counting, however the difference between fvcore and THOP is only about 12 %.

Table 3 shows the measured average inference time and power consumption during inference for different combinations of system and quantization settings. On the Raspberry Pi 4 and 5, as well as the Nvidia Jetson Orin NX 16GB, the standard Linux distribution provided by the manufacturer for the specific hardware was installed. Inference was performed using the regular Pytorch-package, with inference being only performed in the CPU of the Nvidia Jetson.

When deploying the model to the VectorBlox accelerator, a limit of 32 MB for the size of .vnnx-models was encountered. This is due to the reserved portion in the DDR memory in the configuration for the Icicle kit. At the same time, the resulting .vnnx file size depends on the target architecture, with a blob for the V500 architecture being considerably larger than for the V1000 architecture. The .vnnx-file for the cloud segmentation model is about 50 MB in size for the V500 architecture and about 7 MB for the V1000 architecture, but the deployment of the V1000 accelerator on the development board is still ongoing.

The raspberry Pi 4 with its Cortex A72 processor, The Pi 5 with its Cortex A76 and the Jetson Orin NX16 with the Cortex A78 are all part of the same processor family, with the A72 being the least powerful and the A78 being the most powerful. Thus, their performance relative to each other seems reasonable. The slower inference for 8-bit integer quantization in the A78 of the Jetson Orin can be traced back to the processor being set to the 10 W power mode, disabling some processor cores and limiting clock speed. With all cores active and drawing close to 18 W of power, the inference time for 8-bit integer quantization in the A78 drops to about 0.42 s.

In general, it can be seen that contrary to expectations the inference time for 8-bit integer quantization is generally longer than for 32-bit floating point operations, even though the processors are generally capable of more integer operations per second than floating point operations. This is caused by the specific implementation of the quantized operations in Pytorch and the target of future research.

6. OUTLOOK

As a next step, we plan to finalize the deployment of all three VectorBlox accelerators in the Icicle kit, followed by the execution of the cloud segmentation model in the accelerator.

Next, we plan to set up tests resembling the final architecture of the payload processing unit in the satellite. This will consist of a camera connected to a PolarFire development board simulating SPPM1, while this board is also connected to the satellite's OBC and another PolarFire development board, standing in for SPPM2. This way, the complete pipeline from tasking the image acquisition over the image analysis using ML and the evaluation of the results can be tested in an environment close to final deployment in the satellite. Additionally, the networking functionality of the KronOS middleware can be further tested.

Additionally, we plan to look into benchmarking the ML inference capability of VectorBlox, as well as other inference accelerator designs on ML-models of different architectures and performance classes.

ACKNOWLEDGEMENTS

This work originated from the project "Verbundvorhaben UWE-5: Studentisches Ausbildungsprojekt zur Untersuchung der Einbindung von Nanosatelliten in 5G-Netze" which is funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under the funding labels 50RU2307A and 50RU2307B.

Contact Address:

moritz.heimbach@uni-wuerzburg.de

Literatur

[1] Gianluca Giuffrida, Lorenzo Diana, Francesco de Gioia, Gionata Benelli, Gabriele Meoni, Massimiliano Donati, and Luca Fanucci. Cloudscout: a deep neural network for on-board cloud detec-

tion on hyperspectral images. Remote Sensing, 12(14):2205, 2020.

[2] James MacKinnon, Troy Ames, Dan Mandl, Charles Ichoku, Luke Ellison, Jacob Manning, and Baram Sosis. Classification of wildfires from modis data using neural networks. In Machine Learning Workshop, number GSFC-E-DAA-TN46421, 2017.

[3] Shoya Higa, Yumi Iwashita, Kyohei Otsu, Masahiro Ono, Olivier Lamarre, Annie Didier, and Mark Hoffmann. Vision-based estimation of driving energy for planetary rovers using deep learning and terramechanics. IEEE Robotics and Automation Letters, 4(4):3876–3883, 2019.

[4] Larry Matthies, Shreyansh Daftry, Scott Tepsuporn, Yang Cheng, Deegan Atha, R Michael Swan, Sanjna Ravichandar, and Masahiro Ono. Lunar rover localization using craters as landmarks. In 2022 IEEE Aerospace Conference (AERO), pages 1–17. IEEE, 2022.

[5] David Rijlaarsdam, Tom Hendrix, Pablo T Toledano González, Alberto Velasco-Mata, Léonie Buckley, Juan Puig Miquel, Oriol Aragon Casaled, and Aubrey Dunne. The next era for earth observation spacecraft: An overview of cognisat-6. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2024.

[6] Nicola Melega, Nicolas Longepe, Agne Paskeviciute, Valentina Marchese, Oriol Aragon, Irina Babkina, Alessandro Marin, Jakub Nalepa, Leonie Buckley, Giorgia Guerrisi, et al. Development and implementation of the ϕsat-2 mission. In Small Satellites Systems and Services Symposium (4S 2024), volume 13546, pages 1244–1254. SPIE, 2025.

[7] Alessandro Marin, César Coelho, Florian Deconinck, Irina Babkina, Nicolas Longepe, and Massimiliano Pastena. Phi-sat-2: Onboard ai apps for earth observation. Proc. Space Artif. Intell, 6, 2021.

[8] Maciej Ziaja, Piotr Bosowski, Michal Myller, Grzegorz Gajoch, Michal Gumiela, Jennifer Protich, Katherine Borda, Dhivya Jayaraman, Renata Dividino, and Jakub Nalepa. Benchmarking deep learning for on-board space applications. Remote Sensing, 13(19):3981, 2021.

[9] Tobias Schwarz, Oleksii Balagurin, Tobias Greiner, Tobias Herbst, Tobias Kaiser, Hakan Kayal, and Andreas Maurer. Early results and inflight experience of the 6u-mission sonate-2. In Small Satellites Systems and Services Symposium (4S 2024), volume 13546, pages 350–365. SPIE, 2025.

[10] Vishesh Vatsal, Adithya Kothandhapani, Arvind Subramanian, Harmit Vyas, and Abhinav Jayaswal. Challenges in onboard processing: Learnings from mission matterhorn. 2024.

[11] Markus Plattner and Chedi Fassi. Payload computer based on polarfire soc. In 2023 European Data Handling & Data Processing Conference (EDHPC), pages 1–3. IEEE, 2023.

[12] Markus Plattner, Chedi Fassi, and Aron Berner. Smart power supply for fpga/soc. In 2023 13th European Space Power Conference (ESPC), pages 1–4. IEEE, 2023.

[13] Jesko Bahr. An advanced approach to satellite software and communication based on smartos and compass protocol: Design, implementation, and test on the picosatellite platform uwe, 2016.

[14] ARM Cortex A72 1500M Hz benchmark. https://www.cpubenchmark.net/cpu.php?cpu=ARM+Cortex-A72+4+Core+1500+MHz&id=3917. Accessed: 2025-09-16.

[15] ARM Cortex A76 2400M Hz benchmark. https://www.cpubenchmark.net/cpu.php?cpu=ARM+Cortex-A76+4+Core+2400+MHz&id=5743. Accessed: 2025-09-16.

[16] Nvidia Jetson TX2i technical specifications. https://developer.nvidia.com/embedded/jetson-tx2i. Accessed: 2025-09-16.

[17] Nvidia Jetson Xavier NX technical specifications. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/. Accessed: 2025-09-16.

[18] Nvidia Jetson Orin NX 16GB technical specifications. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/. Accessed: 2025-09-16.

[19] Intel Movidius Myriad X VPU technical specifications. https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/myriad-x-product-brief.pdf. Accessed: 2025-09-16.

[20] Matteo Dadà, Luca Zulberti, Pietro Nannipieri, Luca Fanucci, and Silvia Moranti. Inference and evaluation of deep convolutional neural networks on microchip's hardware accelerator vectorblox. In 2023 European Data Handling & Data Processing Conference (EDHPC), pages 1–8. IEEE, 2023.

[21] M Sudarshan Shenoy and M Ramesh Kini. Implementation of reconfigurable deep learning accelerator (rdla) on polarfire soc. In 2023 IEEE Asia Pacific Conference On Postgraduate Research In Microelectronics And Electronics (PRIMEAsia), pages 48–49. IEEE, 2023.

[22] S. Mohajerani, T. A. Krammer, and P. Saeedi. A cloud detection algorithm for remote sensing images using fully convolutional neural networks. In 2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP), pages 1–5, Aug 2018. DOI: 10.1109/MMSP.2018.8547095.