Intelligent Data Processing using In-Orbit Advanced Algorithms on Heterogeneous System Architecture

Nandinbaatar Tsog, Moris Behnam, Mikael Sjödin, Fredrik Bruhn
Mälardalen University
Box 883, 721 23
Västerås, Sweden
firstname.lastname@mdh.se

Abstract— **In recent years, commercial exploitation of small satellites and CubeSats has rapidly increased. Time to market of processed customer data products is becoming an important differentiator between solution providers and satellite constellation operators. Timely and accurate data dissemination is the key to success in the commercial usage of small satellite constellations which is ultimately dependent on a high degree of autonomous fleet management and automated decision support. The traditional way for disseminating data is limited by on the communication capability of the satellite and the ground terminal availability. Even though cloud computing solutions on the ground offer high analytical performance, getting the data from the space infrastructure to the ground servers poses a bottleneck of data analysis and distribution. On the other hand, adopting advanced and intelligent algorithms onboard offers the ability of autonomy, tasking of operations, and fast customer generation of low latency conclusions, or even real-time communication with assets on the ground or other sensors in a multi-sensor configuration.**

In this paper, the advantages of intelligent onboard processing using advanced algorithms for Heterogeneous System Architecture (HSA) compliant onboard data processing systems are explored. The onboard data processing architecture is designed to handle a large amount of high-speed streaming data and provides hardware redundancy to be qualified for the space mission application domain. We conduct an experimental study to evaluate the performance analysis by using image recognition algorithms based on an open source intelligent machine library "MIOpen" and an open standard "OpenVX". OpenVX is a cross-platform computer vision library.

TABLE OF CONTENTS

Swedish special characters (å, ä, ö) in the email addresses are replaced by a and o.

978-1-5386-2014-4/18/$31.00

1. INTRODUCTION

Small satellites (i.e., satellites defined weighing less than 100 kg) all the way down to nanosatellites (i.e. satellites defined weighing less than 10 kg) are rapidly attracting interest in many areas including the commercial telecommunication, Earth Observation (EO) markets, and the intelligence and defense community [9]. EO satellites are experiencing rapid advancements in optical imaging payload technologies and onboard processing, leading to significantly improved quality and resolution of imagery gathered from spaceborne platforms. The smaller size of satellites together with a lower cost has allowed the use of high performing Commercial-Off-The-Shelf (COTS) electronic parts to be harnessed for image compression and cloud removal using both Graphical Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs) [7], [17]. The improvements in sensor technology have not been matched with equivalent developments in satellite downlink technologies, and hence the exponential increases in the generated data volume are forming a significant bottleneck onboard the platform. Optical communication holds promise to enable gigabit per second telemetry data transfer for downlinks and intersatellite links [13]. This would decrease the difference between sensor advances and the communication bottleneck. However, latency and storage capacity will still be big challenge since the number of places on Earth with suitable optical stations are limited.

Many emerging missions use constellations of many (e.g., over 100) small satellites to enable rapid revisit times and global coverage[5]. Small satellites are being deployed for many different applications, e.g., communications, space situational awareness, and Intelligence, Surveillance, and Reconnaissance (ISR), precision agriculture (PA), machine to machine communication, and air traffic management.

In order to address the latency issues and communication bottlenecks, more onboard data analytics is required to enable small satellites to accommodate the flexibility needed for autonomous constellation management, information extraction, compression and sensor fusioning with low latency. However, it is important to find data processing solutions that fit the Size, Weight, and Power (SWaP) constraints while collecting mission-critical sensor data. This paper further explores the use, efficiency and performance of HSA capabilities of modern System-on-Chips (SoC) for ISR sensors (e.g., EO/Infrared/Hyperspectral cameras) using the GIMME-series architectures [18].

GIMME-3 is an architecture developed at Mälardalen University to pursue a SWaP optimized onboard computing solution that enables Deep Learning on massively parallel units with advanced Error Code Correction (ECC) for aerospace application. GIMME-3 has been expanded to GIMME-4 which introduces HSA capability through the AMD® R-series SoC

(f.m. named Merlin Falcon) [18].

Radar is an another important sensor which usually requires massive processing. New methods suitable for onboard processing are being developed. Single-frequency transmitted wave-forms with high Doppler resolution nature called Doppler synthetic aperture radar (D-SAR) is one interesting approach for bistatic radar [19].

Contributions

The main contribution of this paper is to investigate the performance of onboard data processing on the heterogeneous architecture by running image processing algorithms which use both MIOpen framework of high performing machine learning primitives and OpenVX vision library. We focus on the fact that the concurrent executions of multiple advanced algorithms affect the worst-case execution time (WCET) of other parallel running tasks which of expresses the quality of the onboard heterogeneous system. Since less energy consumption is an another key factor to increase the quality of the onboard architecture, we have focused on the energy consumption of GPU based heterogeneous computing. We confirmed that an HSA compliant GPU based heterogeneous computing improves the quality of the onboard architecture for intelligent data processing since it either uses less energy consumption and performs better than CPU for the feature tracking with the different workloads.

Organization

In Section 2, we discuss an importance of onboard processing, usage of in-orbit heterogeneous architectures and energy consumption of parallel architectures running advanced algorithms. The architectures and specifications are introduced in Section 3. We describe our benchmark suites in Section4 and the evaluation of the experiments are described in Section 5. Section 6 concludes the paper and discusses future work.

2. RELATED WORK

In this work, we consider the contribution of heterogeneous architecture in mission critical applications such as onboard processing. Advanced and resource-intensive computing in new science-mission applications brings a new challenge to the space-computing community as these needs require next-generation systems that should support a broad potential of processing with low power consumption and high reliability [21]. Certain numbers of the heterogeneous multicore SoC platforms are introduced as a promising architecture for space-computing. In order to increase the reliability of such platforms, Wilson et al. consider a multifaceted strategy (HARFT strategy) for fault-tolerant computing, targeting SoC platforms consists of multicore CPUs and FPGA fabric. The HARFT strategy introduces fault-tolerant schemes by using both compute nodes to achieve a robust, hybrid, hardware redundant and fault-tolerant theme for a hybrid device.

Heterogeneous architectures including FPGA are not only the hybrid solution in space-computing, but also several architectures including GPUs exist. However, fault masking and tolerance on GPUs is less investigated for harsh environments [16]. Milluzzi and George discuss GPU protection on Tegra X1 SoC for space usage, i.e., how to avoid vulnerability of Tegra SoCs against a wide range of single-event upsets (SEUs) since it has complex caching structure which consists of a number of the GPU cores and a custom task scheduler. As a GPU protection, they consider a persistent thread method with triple-modular redundancy (TMR) which provides a strong basis for fault masking on a wide range of platforms. They have succeeded to remove the vulnerability of scheduler faults even when GPUs pose a unique challenge to a general TMR implementation. The paper reports that the NVIDIA Tegra™ K1 and X1 perform over 500 GFLOPS of peak performance at just 10 Watts Thermal Design Power (TDP). Hence, the SoC considered in our paper possesses sufficient computational performance that it employees Radeon™ R6 GPU which has the computational potential up to 614GFLOPS and less than 10Watt in peak performance.

Persistent threads style programming model/method is well-known to protect GPU from the interference of host CPU since it performs the direct communication with the different GPU kernels instead of unnecessary round-trip communications through host CPU. Moreover, this method is useful for FPGA protection as well. Khan et al. present a complete networking switch designed in OpenCL that consists of several high-level constructs which create the building blocks of any network application for FPGAs [14]. In this work, persistent kernel method is used to avoid the intervention of the host to provide the kernels with data processing constantly. Measuring the intervention of the tasks to one another and between the different compute nodes is one of the main challenges in our work to assess the quality of the concurrent executions.

Measuring the energy consumption of the advanced algorithms while running on the onboard computer is another challenge in our work. Liu et al. tackle with an advanced algorithm using the aerosol optical depth (AOD) properties from the performance and energy efficiency perspective [15]. As a result of a large number of remote sensing data and compute-intensive algorithms, the AOD retrieval is computationally expensive. Two different kinds of parallel architectures, multicore processors and GPU accelerators, are used to run the time-consuming SRAP-MODIS algorithm for the AOD retrieve. This algorithm includes not only a set of nonlinear equations but also requires a large number of input images. In this paper, the performance of parallel computations on both of the architectures and energy consumptions are analyzed in the context of a quantitative remote sensing retrieval application. The difference of the power consumptions between the idle and load conditions [10] is used as the power consumption of the applications for the multicore and GPU in order to evaluate the power consumption. We use this measurement method to determine the real power consumption of the application running.

In order to adapt to unexpected situations, acquirement of cognitive capabilities is important to autonomous control systems in space [8]. Q-learning is a model-free reinforcement learning technique and it is efficient in solving some classes of learning problems. Due to the constraints, SWaP, convergence rate and costs, learning algorithms are rarely implemented in onboard embedded systems in space. Similarly to exploring Convolution Neural Network in our paper, Gankidi and Thangavelautham present Q-learning with Aritificial Neural Network. This method fits well with the parallel computing and it achieved a great reducing processing time by using the fine-grain parallelism of an FPGA hardware. The result shows 43x speed up by Virtex 7 FPGAs compared to Intel i5 2.3 GHz CPUs. They emphasize that the fine grained parallel architectures are competitive considering power consumption.

3. BACKGROUND

AMD A-series A10-8700P APU

As illustrated in Figure 1, the AMD A10-8700P APU maintained in a SoC comprises a quad core 1.8GHz 64bit A10 CPU and Radeon™ R6 GPU. The CPUs consist of 2 compute units (CUs) each of which has 2 cores and shares 1MB L2 cache. The cores integrated into the same CU share 96KB L1 instruction cache, and 32KB L1 data caches are unique to each core. The GPU consists of 6 CUs with 64 cores each, and totally performs up to 614GFLOPS. The APU shares 8GB DDR3 handled by memory controllers with a full hardware cache coherence at 128 bit-wide memory bandwidth between GPU and CPU caches. An Address Translation Cache (ATC) hierarchy brings a fine grained shared virtual memory, i.e., the same virtual address space to all devices. This feature is known as a foundational aspect of HSA that is merely passing the pointer to data between all the devices and no memory copying required for the different CUs. The AMD Embedded G- and R-Series SoCs are used with the A-series APUs and there exist COTS products such as conga-TR3[2] with the size of 95mm by 125 mm. The AMD Embedded R-series SoC is based on a 28 nm process and compliant with HSA 1.0 thus meaning it supports fine-grain full cache coherency. The power consumption of this APU ranges between from 12 Watt up to 35 Watt while the thermal design power (TDP) is 15Watt.

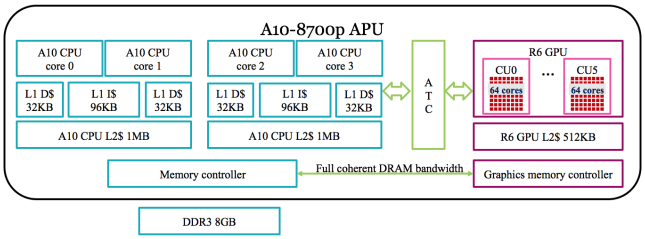


Figure 1. AMD A-series A10 APU's architecture

GIMME3 and GIMME4

A fault tolerant heterogeneous architecture, GIMME3, has been designed for high performance computing in mission critical applications [6]. GIMME3 architecture was productized and commercialized by Unibap AB (publ) and was flown into space on May 30th 2016 on the Satellogic NuSat-1 and NuSat-2 [18]. GIMME3 employees the AMD Embedded G-series SoC based on FT3 and FT3(b) footprints (formerly known as Kabini and Steppe Eagle, respectively) that support up to quad core CPU with 2 GPU CUs [1]. Each AMD GPU CU has 64 Arithmetic Logic Units (ALU) and GIMME3 can deliver 77 GFLOPS of GPU performance.

As an expansion of GIMME3, Tsog et al. introduce the next generation heterogeneous computing architecture GIMME4 using HSA for higher computing performance and better redundancy [18]. Similarly to the AMD A10-series APU, GIMME4 architecture is based on the AMD Embedded 2nd Gen R-series SoC with 8 GPU CUs and FP4 footprint, formerly known as Merlin Falcon. Major differences between the FT3(b) and FP4 footprint based products are the shift in CPU architecture from 'Bulldozer' to 'Excavator', updated GPU design, memory controller, and the official support for HSA [18]. The Unibap e2200 family development board based on GIMME4 architecture is 82 mm by 110 mm and 85g, and provides 819GFLOPS of GPU performance. Currently, GIMME4 has not tested yet in radiation environments, however, the previous version GIMME3 is fully confirmed [6] that it operates in radiation environments.

Heterogeneous System Architecture

In modern trend in industrial applications, the role of heterogeneous computing has been increasing dramatically. Employing multiple types of compute nodes, CPU, GPU, FPGA, DSP and so on, according to their strengths makes the embedded systems as robust as much. However, the different types of specifications and designs of the compute nodes bring difficulties for the developing process from cost and timing perspective. To overcome these problems, multiple leading hardware vendors have established HSA Foundation[3] to develop the Heterogeneous System Architecture (HSA) specification for reducing heterogeneous computing complexity and providing the developer friendly environments. The HSA aims to ease the process of developing the heterogeneous platform by providing the similar environment for the developers such as they used for the legacy systems, i.e., homogeneous systems. For example, providing the open-source well-known compilers, LLVM and GCC, and using only pointers in a virtual memory space gives access to the memory spaces of all the compute nodes. The virtual memory space in the HSA provides no memory copying between different physical memories, i.e., the HSA provides unified coherent memory that saves a lot of computation time for transferring data between different physical memories. As a part of HSA, AMD contributes by introducing an initiative GPUOpen[4], an open-source software stack, including, but not limited to, kernel level driver, runtime, tools and libraries such as ROCm, MIOpen, AMD OpenVX and CodeXL.

ROCm—ROCm is an open source software stack and consists of multiple modules which support GPU computing [18].

MIOpen—MIOpen, an alternative to CuDNN, is an open-source machine learning library that developed to exert full potential of ROCm software stack as well as heterogeneous computing. In the current release (version 1.0), MIOpen supports Convolution Neural Network (CNN), Pooling, Softmax, Activations, Gradient Algorithms Batch Normalization, and LR Normalization[20] with data described in 4-D tensors. Both OpenCL and HIP frameworks are enabled in MIOpen that HIP includes a tool "hipify" which ports CUDA code into C++.

OpenVX—The Khronos Group has designed an open and royalty-free standard OpenVX that is portable across different vendors and hardware types, and optimized and power-efficient image processing for computer vision applications. OpenVX enables the following use cases; face, body and gesture tracking, smart video surveillance, advanced driver assistance systems (ADAS), object and scene reconstruction, augmented reality, visual inspection, robotics and more [23]. Moreover, OpenVX and OpenCV complement each other to perform as perfect computer vision library since OpenVX has to be implemented by hardware vendors and OpenCV has a strong open source community. Not only, OpenVX is the computer vision library, but also it has great potential to being as a machine learning library in its Neural Network Extension. There are multiple vendors implement

[2]conga-TR3: http://www.congatec.com/en/products/com-express-type6/conga-tr3.html

[3]HSA Foundation: http://www.hsafoundation.com

[4]GPUOpen: https://gpuopen.com

their OpenVX libraries for both computer vision and neural network libraries. However, we focus only on computer vision library and use AMD OpenVX (AMDOVX) [3] in this paper. Currently, the released version of the AMDOVX is 0.9.6 and it includes feature tracking "Optical flow" algorithm as well as feature detection algorithms, e.g. Harris, FAST, Canny. Furthermore, the current version interoperates with OpenCV as well.

CodeXL—CodeXL[5] is an open-source development tool suite, debugging and profiling, for the different processors such as CPU, GPU, and APU. Using CodeXL facilitates the HSA development process as it provides debugging functionality for OpenGL, OpenCL and HSA, and profiling functionality for both OpenCL and HSA kernels. CodeXL works on both Windows™ and Linux™ as a Visual Studio™ extension and a standalone user interface with both graphical and command line. In this paper, we deal with CodeXL mainly for power profiling purpose, however, it is used for debugging purpose as well.

4. Experiment Setup

As illustrated in Figure 2, we consider a comparison of the process of a reporting system using satellite data. The traditional reporting system is depicted with solid lines and it consists of storing raw data on the onboard computer of satellite, downlinking to the ground station, processing data on the ground station and creating the report. Meanwhile, the report creating system using onboard processing is illustrated with dash lines. The system begins with onboard processing and storing analyzed data on the onboard computer. The report will be ready once the analyzed data is downlinked to the ground station. In this paper, we consider the onboard processing only.

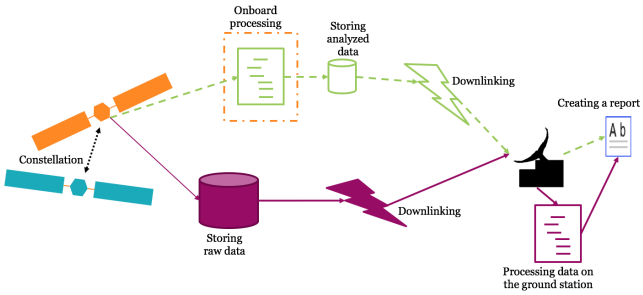


Figure 2. A reporting system using satellite data

The SWaP is the key to evaluate advancing onboard processing while assessing the quality of the onboard processing. The size constraint is satisfied as both the Unibap e2200 family product and the conga-TR3 fit in the 1.5U CubeSats or more. From the weight perspective, the Unibap e2200 family product takes less than 10% of the entire weight of 1U CubeSat as specified by the CubeSat standard and less than 3% of 3U CubeSat which is the preferred size of CubeSats for the advanced missions. By the specification of the board/system, the power consumption is known as between 15-35Watt that fits for the 3U CubeSats [4]. However, we investigate the detail of power consumption to find the real power consumption of the advanced onboard processing while accessing the quality of the onboard processing.

[5]CodeXL: https://gpuopen.com/compute-product/codexl/

Benchmark suites

We design the following experiment scenarios by using MIOpen and OpenVX with the CodeXL in order to assess the HSA compliant onboard computer for advanced processing.

- *ExpA - An investigation of the computational performance and power consumption in CPU and GPU.* The goal of this experiment is to investigate the computational performance and power consumption of advanced algorithms on CPU and GPU devices of the HSA compliant onboard computer. We use the CodeXL profiler to measure the power consumption of the advanced algorithms. To the best of our knowledge, CodeXL is an optimal tool to measure the power consumption since it omits to equip physical measuring tools.
- *ExpB - Assessing a quality of the concurrent executions of advanced tasks.* In this experiment, we aim to investigate the quality of the HSA compliant onboard computer by executing the multiple concurrent advanced algorithms. We consider a comparison of the measurement-based worst case execution time (WCET) of a task with different workloads on the CPU and GPU.

Configuration of test scenarios

In the experiment ExpA, we consider the following 7 tasks (in Table 1) on both CPU and GPU computations; OVX1, OVX2, ML1-1, ML1-2, ML1-3, ML1-4 and ML1-5. OVX1 and OVX2 are based on a tracking algorithm with the different test data [11] and [12], respectively. ML1-1, ML1-2, ML1-3, ML1-4 and ML1-5 are the following machine learning applications of MIOpen with the default configurations, respectively; Activations, Batch Normalization, CNN, LR Normalization and Pooling.

Shortened name	Detailed name
OVX1	Tracking algorithm with test data 1 [11]
OVX2	Tracking algorithm with test data 2 [12]
ML1-1	Activations
ML1-2	Batch Normalization
ML1-3	CNN
ML1-4	LR Normalization
ML1-5	Pooling

Table 1. Tasks' shortened and detailed name list

The combinations of the following 5 tasks are used in the experiment ExpB; OVX1, OVX2, ML10, ML100 and ML1000. OVX1 and OVX2 are the same as we defined for the experiment ExpA. ML10, ML100 and ML1000 are the concurrent executions of the machine learning applications Activations, Batch Normalization, CNN, LR Normalization and Pooling with a custom configurations such as 10, 100 and 1000 iterations, respectively. We measure the WCET of the task OVX1 while we consider the following tasks as the workloads; OVX2, ML10, ML100 and ML1000. The task OVX1 converts a 4K image to 1280x720 RGB image for every frame and tracks the features.

Test data

In this paper, we use the International Space Station (ISS) Expedition 42's time lapse videos of earth as test data. The resolution of the videos is 4K 3840x2160 and the size of each frame is around 6MB. The test data 1 and 2 have 180 and 600 frames, respectively. The frame rate of both the data is 60fps.

Evaluation environment

The experiments are performed on A10-8700P APU employed Acer E15 E5-552-T99R model notebook. The follow-

Tasks	Computation time			Energy consumption		
	GPU [ms]	CPU [ms]	Ratio=CPU/GPU	GPU [Joules]	CPU [Joules]	Ratio=CPU/GPU
OVX1	79.33	137.35	1.73	4.41	4.78	1.08
OVX2	31.18	93.62	3.00	3.92	4.34	1.11
ML1-1	1.12	0.66	0.58	1.09	1.14	1.05
ML1-2	0.19	22.34	119.67	0.73	0.87	1.19
ML1-3	12.06	2873.56	238.20	1.63	22.01	13.52
ML1-4	0.57	86.82	153.23	0.75	1.43	1.89
ML1-5	1.73	29.65	17.16	0.76	0.99	1.31

Table 2. The computation time and energy consumption of the tasks per frame or iteration in the ExpA.

ing software stacks are installed in the environment; Ubuntu 16.04, Linux Kernel 4.14.rc3 , ROCm 1.6.3, CodeXL 2.5-25, MIOpen 1.0 and OpenVX 0.9.6. To reproduce the experiments, we have implemented the patches[6] to the CodeXL, since the current version of CodeXL is not suitable for the newer versions of the Linux Kernel than 4.10.

5. Experiment Results

The accuracy of the tasks (algorithms) are confirmed by the comparison of calculations of both CPU and GPU. We have confirmed the optical flow and Harris feature algorithms of OpenVX as shown in Figure 6 in Appendix A. The results of the ExpA are shown in Table 2. Both the results of computation time and energy consumption are measured per frame and iteration for OpenVX and MIOpen applications, respectively. We can see that GPU computes faster than CPU except in case of the ML1-1 task. The speed up ratio reaches up to 238 times in the ML1-3 task which is based on one the most well-known algorithm CNN. From the energy consumption perspective, GPU leads CPU for all the cases. Moreover, GPU consumes surprisingly 13.52 times less energy in case of CNN algorithm as GPU reaches 6 Watts for the full performance in contrast with the usage of CPU reaches 3 Watts. As we know the results are per unit (frame and iteration), this experiment shows that GPU is a potential candidate for the onboard processing.

Processor	Task / Workloads				
	OVX1	OVX2	ML10	ML100	ML1000
GPU [s]	7.11	9.34	0.25	1.92	21.51
CPU [s]	11.52	24.32	27.57	320.05	5406.53
CPU/GPU	1.62	2.60	111.94	166.82	251.30

Table 3. The computation time of each workload running alone (no concurrent executions of any other tasks as well as the workloads).

The aim of the expB is to investigate the measurement-based WCET of the task (OVX1) while running with the different workloads. The computation time of the workloads in the expB is the total computation time of the tasks, meanwhile we considered the computation time per unit (frame or iteration) similar to the case in the ExpA. In Table 3, we can see that the computation time of the workloads are different. Moreover, the computation time of the workloads is measured when each workload runs alone, i.e., no concurrent executions of any other tasks at the same time. We can see that it is better to run the machine learning algorithms on GPU than CPU since the ratio of CPU/GPU increases when the iteration number of a running algorithm increases.

[6]The patches to CodeXL 2.5-25:
https://github.com/GPUOpen-Tools/CodeXL/issues/161#issuecomment-337128790

Figure 3 shows the variation of the computation time of OVX1 while running together with the different workloads. Detail information are shown in Table 4. We can see that the computation time of the OVX1 slows down to 2.82 times from 11.52s (no workload) to 32.56s (OVX1+OVX2+ML1000) on CPU. Meanwhile, the increase of the computation time stays 2.23 times from 7.11s (no workload) to 15.86s (OVX1+OVX2+ML1000) on GPU. Therefore, GPU takes from 40ms (7.11s/179) to 89ms for the calculation of each frame, and CPU takes from 64ms to 182ms for each frame. As shown in Figure 3, we confirm that the computation time of the GPU is more stable than CPU. The task OVX1 uses the data1 which is 180frames, 4K resolution and 60fps. In other words, it takes 1/60s=17ms per frame. Therefore, the computation time of CPU is approximately 10.71 times more time compare to the frame rate time of the video. However, GPU consumes approximately 5.24 times more time of the frame rate time of the video with 4K resolution, and it could be considered reasonable with the lower frame rate (20-30fps) and the lower resolution of the images.

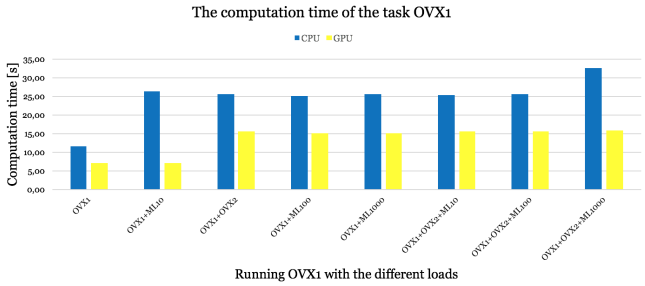


Figure 3. The variation of the computation time of OVX1 while running together with the different workloads.

Concurrent executions	The computation time of the task OVX1	
	CPU [s]	GPU [s]
OVX1 + no workload	11.52	7.11
OVX1 + ML10	26.36	7.24
OVX1 + OVX2	25.50	15.59
OVX1 + ML100	25.17	15.17
OVX1 + ML1000	25.64	15.17
OVX1 + OVX2 + ML10	25.39	15.55
OVX1 + OVX2 + ML100	25.74	15.66
OVX1 + OVX2 + ML1000	32.56	15.86

Table 4. The computation time of the task OVX1 while running together with the different workloads in the ExpB.

6. Conclusion / Future Work

First of all, we have come to conclusion that GPU is a potential candidate for the onboard computer processing

of the CubeSat as we performed two experiments over 7 different machine learning and computer vision algorithms. From our experimental study, we have confirmed that the HSA compliant GPU computes up to 238 times faster and consumes between 13.5 times less energy, compared to the CPU calculation. Moreover, we have confirmed that the computation time of GPU is more stable than CPU while running together with the different workloads. Therefore, we conclude that GPU can be a highly potential candidate in the onboard computer processing of the CubeSat. For future work, we would like to continue developing combined usage of the intelligent applications for the onboard computer of the CubeSat that allows more usability and reliability.

APPENDICES

A. TEST DATA

Source of the test data

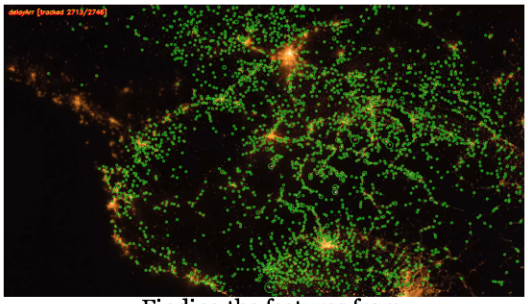
The first frames of ISS Expedition 42's time lapse videos with the id number "jsc2015m000221" and "jsc2015m000226" are shown in Figures 4, and 5, respectively. These time lapse videos are assembled from JSC still photo collections (still photos iss042e255412 - iss042e255592 and iss042e283240 - iss042e283840).

Figure 4. The first frame image from ISS Expedition 42 Time Lapse Video of Earth with the id "jsc2015m000221"

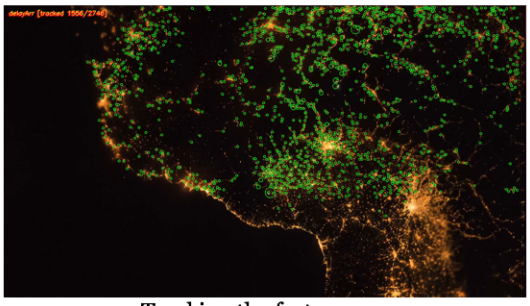
Figure 5. The first frame image from ISS Expedition 42 Time Lapse Video of Earth with the id "jsc2015m000226"

Tracking results

The results of the tasks OVX1 and OVX2 are shown in Figure 6.

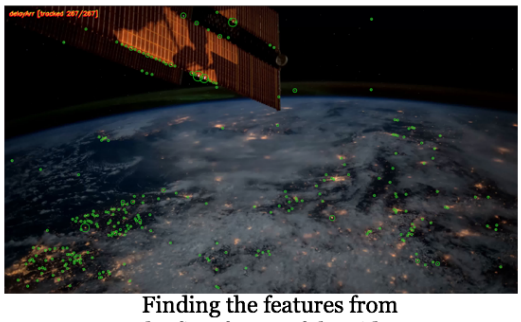


Finding the features from the first frame of the video

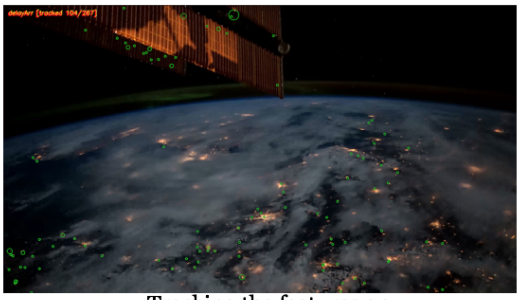


Tracking the features on the coming frames

(a)Applying OVX1 to the data1



Finding the features from the first frame of the video



Tracking the features on the coming frames

(b)Applying OVX2 to the data2

Figure 6. Applying OVX1 and OVX2 to the data1 and data2, respectively

B. PSEUDO CODE FOR THE MEASUREMENTS OF THE COMPUTATION TIME

A guide to measure the computation time is shown in the following pseudo code.

Algorithm 1 How to set a timestamp to the MIOpen code

```
* Include header files located in MIOpen/driver/.
- #include "InputFlags.hpp"
- #include "timer.hpp"

* Declare a timestamp variable
- Timer t;

* Set the timestamp for START
- START_TIME;

* Set the timestamp for END and calculate the computation
time
if inflags.GetValueInt("time") == 1 then
STOP_TIME;
if WALL_CLOCK then
print t.gettime_ms();
end if
end if
```

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to Dr Harris Gasparakis, an AMD GPGPU, Computer Vision and Machine Learning technical expert and project manager, for his great knowledge in computer vision, machine learning and HSA related areas. He has helped us a great deal by providing an extensive amount of support whenever necessary.

REFERENCES

[1] AMD. 2017, Embedded G-Series SoC Processors — AMD, http://www.amd.com/en-us/products/embedded/processors/g-series, (accessed 2017-10-12).

[2] AMD. 2017, Application Brief of AMD Embedded R-Series SoC, http://www.amd.com/Documents/merlin-falcon-product-brief.pdf, (accessed 2017-10-12).

[3] AMD OpenVX: https://gpuopen.com/compute-product/amd-openvx/, (accessed 2017-10-09).

[4] S. S. Arnold, R. Nuzzaci and A. Gordon-Ross, "Energy Budgeting for CubeSats with an Integrated FPGA,"

[5] C. R. Boshuizen, J. Mason, P. Klupar and S. Spanhake, "Results from the Planet Labs Flock Constellation," in Small Satellite Conference 2014.

[6] F. Bruhn, K. Bruhnberg, J. Hines, K. Asplund and M. Norgren, "Introducing radiation tolerant heterogeneous computers for small satellites," 2015 IEEE Aerospace Conference, Big Sky, MT, 2015, pp. 1-10. doi: 10.1109/AERO.2015.7119158

[7] R. L. Davidson and C. P. Bridges, "GPU accelerated multispectral EO imagery optimised CCSDS-123 lossless compression implementation," 2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1-12. doi: 10.1109/AERO.2017.7943817

[8] P. R. Gankidi and J. Thangavelautham, "FPGA architecture for deep learning and its application to planetary robotics," 2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1-9. doi: 10.1109/AERO.2017.7943929

[9] P. Hershey, B. Wolpe, J. Klein and C. Dekeyrel, "System for small satellite onboard processing," 2017 Annual IEEE International Systems Conference (SysCon), Montreal, QC, 2017, pp. 1-6. doi: 10.1109/SYSCON.2017.7934705

[10] S. Hong and H. Kim, An integrated GPU power and performance model, in Proc. ACM SIGARCH Comput. Archit. News, 2010, pp. 280289. doi: 10.1145/1815961.1815998

[11] ISS Expedition 42 Time Lapse Video: https://images.nasa.gov/#/details-jsc2015m000221.html, (accessed 2017-10-19).

[12] ISS Expedition 42 Time Lapse Video: https://images.nasa.gov/#/details-jsc2015m000226.html, (accessed 2017-10-19).

[13] S. Janson, D. Rowen, T. Rose and R. Welle, "CubeSat Scale Laser Communication," 31st Space Symposium, Colorado Springs, Colorado, April 13-16, 2015.

[14] J. Khan, P. Athanas, S. Booth and J. Marshall, "OpenCL-based design pattern for line rate packet processing," 2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Seattle, WA, 2017, pp. 190-194. doi: 10.1109/ASAP.2017.7995278

[15] J. Liu, D. Feld, Y. Xue, J. Garcke and T. Soddemann, "Multicore Processors and Graphics Processing Unit Accelerators for Parallel Retrieval of Aerosol Optical Depth From Satellite Data: Implementation, Performance, and Energy Efficiency," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, no. 5, pp. 2306-2317, May 2015. doi: 10.1109/JSTARS.2015.2438893

[16] A. Milluzzi and A. George, "Exploration of TMR fault masking with persistent threads on Tegra GPU SoCs," 2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1-7. doi: 10.1109/AERO.2017.7943882

[17] L. Santos, L. Berrojo, J. Moreno, J. F. López and R. Sarmiento, "Multispectral and Hyperspectral Lossless Compressor for Space Applications (HyLoC): A Low-Complexity FPGA Implementation of the CCSDS 123 Standard," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 9, no. 2, pp. 757-770, Feb. 2016. doi: 10.1109/JSTARS.2015.2497163

[18] N. Tsog, H. Gasparakis, M. Behnam, M. Sjödin and F. Bruhn, "Technical Report: Advancing Onboard Computer Data Processing in CubeSats," Technical Report, Mälardalen University, 2017.

[19] L. Wang and B. Yazici, "Bistatic Synthetic Aperture Radar Imaging Using UltraNarrowband Continuous Waveforms," in IEEE Transactions on Image Processing, vol. 21, no. 8, pp. 3673-3686, Aug. 2012. doi: 10.1109/TIP.2012.2193134

[20] MIOpen: https://github.com/ROCmSoftwarePlatform/MIOpen, (accessed 2017-10-05).

[21] C. Wilson, S. Sabogal, A. George and A. Gordon-Ross, "Hybrid, adaptive, and reconfigurable fault tolerance,"

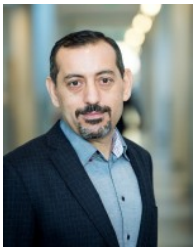
2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1-11. doi: 10.1109/AERO.2017.7943867

[22] GPUOpen: https://gpuopen.com/, (accessed 2017-10-09).

[23] OpenVX: https://www.khronos.org/openvx/, (accessed 2017-10-09).

BIOGRAPHY

Nandinbaatar Tsog *is a PhD student at Mälardalen University, Sweden. He received his B.S. in computer science from University of Electro-Communications, Japan in 2006. After few years of industrial experience, he finished M.S. in computer science specialized in intelligent embedded systems from Mälardalen University in 2016. The aim of his research is to investigate predictability in heterogeneous architectures such as HSA.*



Moris Behnam *has awarded a B.Eng., and M.Sc. in Computer and Control Engineering at the University of Technology, Iraq, and also M.Sc., Licentiate, and PhD in Computer Science and Engineering at Mälardalen University, Sweden, in 1995, 1998, 2005, 2008 and 2010 respectively. Moris has been a visiting researcher at Wayne State University, USA in 2009 and he has been a Postdoctoral Researcher at University of Porto in 2011. He is the Head of embedded systems division at Mälardalen University. His research interests include real-time scheduling, synchronization protocols, multicore/multiprocessor systems, distributed embedded real-time sysems, using control theories in real-time scheduling, indutrial cloud computing and internet of things.*

Mikael Sjödin *received his PhD in computer systems 2000 from Uppsala University (Sweden). Since then he has been working in both academia and in industry with embedded systems, real-time systems, and embedded communications. The current research goal is to find methods that will make software development cheaper, faster and yield software with higher quality. Concurrently, Mikael is also been pursuing research in analysis of real-time systems, where the goal is to find theoretical models for real-time systems that will allow their timing behavior and memory consumption to be calculated.*

Fredrik Bruhn *received a Ph.D. in Microsystems Technologies from Uppsala University, Uppsala, Sweden in 2005 and a Masters of Science in Atomic and Molecular Physics from Uppsala University in 2000. He graduated from International Space University Summer Session Program in 2001. He has been with Mälardalen University since 2013 as Adjunct Professor in Robotics and Avionics. He has been a guest researcher at JPL and entrepreneur starting several high technology companies in robotics and space applications.*