IAC–24–B.4.6B.8

Reconfiguration of FPGA during operation of small satellite for flexible hyperspectral data compression

Simen Løcka Eine,[a*], Dennis D. Langer[b], Roger Birkeland and Milica Orlandic[c]

[a] *Department of Electronic Systems (IES), Norwegian University of Science and Technology (NTNU), O.S. Bragstads Plass 2B, 7034 Trondheim, Norway,* einesimen@gmail.com
[b] *Department of Marine technology (IMT), NTNU,* dennis.d.langer@ntnu.no
[c] *IES, NTNU,*
[*] *Corresponding author*

Abstract

Reconfigurable hardware can be used to increase processing efficiency in an environment limited by size, time and power, such as in small satellites. In this paper we demonstrate how the reconfiguration of Field Programmable Gate Arrays (FPGAs) can be used to utilize the same hardware for different programmable architectures, increasing flexibility and efficiency on an onboard processing system that has limited resources. The small satellite HYPSO-1 (HYPerspetral Smallsat for Ocean observation) is used as a case study, for which FPGA reconfiguration during satellite operations is possible. HYPSO-1 is a 6U CubeSat with a hyperspectral camera as its payload, which features a hardware accelerator for hyperspectral data cube compression based on CCSDS123. The hardware accelerator can compress a hyperspectral data cube of nominal dimensions in less than a second of processing time. In this paper, we present 1) a method for testing and verifying new FPGA designs before permanent integration into satellite operation, and 2) adding support for FPGA compression of a multitude of cube dimensions, as a static hardware implementation can compress only a specific cube dimension. On-board reconfiguration of the FPGA is performed automatically depending on the specified cube dimensions in a capture command. The FPGA accelerated compression decreases the compression time for hyperspectral data cubes from 5-6 minutes to less than a second, as the software implementation of CCSDS123 compression can now be omitted for non-nominal cube dimensions. The time saved per capture reduces energy usage and time until the hyperspectral camera instrument can collect data again. These factors impact the total imaging capacity of the system and are especially useful when scheduling the imaging of geographically close targets, that are to be imaged during the same satellite pass.

Keywords: Cubesat, hyperspectral remote sensing, compression, FPGA, reconfiguration.

1. Introduction

FPGAs are programmable logic devices, often used to run optimized algorithms faster and more efficiently than CPUs and GPUs. These devices are now often used in small satellite systems, such as CubeSats, as processing tasks for satellite data are strictly defined and therefore can often be optimized for deployment on FPGAs [1]. By using the reconfiguration of FPGAs to run different algorithms, the devices become more flexible and can speed up multiple processes or applications significantly.

The HYPSO mission was developed at the Norwegian University of Science and Technology (NTNU) and the mission aims to research and observe the ocean by capturing hyperspectral image data from a constellation of small satellites. The hyperspectral data can e.g. be used to detect Harmful Algal Blooms or for water quality monitoring. The HYPSO-1 CubeSat is the first satellite in the HYPSO constellation, and it was launched in January 2022 [2]. HYPSO-2 was launched in August 2024. The OPU (Onboard Processing Unit) for the payload on both HYPSO-1 and HYPSO-2 contains a Zynq System on Chip (SoC) from AMD Xilinx. The SoC has an ARM processor that runs Linux and uses custom software to interact with the rest of the satellite. A Kintex-7 FPGA is available on the SoC, and the software interacts with this FPGA via memory [3].

HYPSO-1 can acquire orders of magnitude more data than is possible to downlink, using its 1 Mbit/s S-band radio. Hence, HYPSO-1 does not observe continuously, but divides data acquisition up into small packets of data from specific targets that are called hyperspectral data cubes or captures. The hyperspectral data cubes are compressed to reduce the needed downlink time. The lossless CCSDS123 issue 1 hyperspectral image data compression algorithm [4] is used, and it can reduce the size of a typical capture on HYPSO-1 from 153 MB to 80 MB or less [5]. This algorithm is implemented both in software and on the FPGA, typically taking about 398 s on the CPU and 186 ms on the FPGA. [3]. The hyperspectral data cube dimensions supported for compression in the FPGA implementation are fixed to a single configuration, called the *nominal dimensions*. The more flexible software algorithm that runs on CPU is used to compress captures of any non-nominal dimension. Software updates post-launch are used to enable compression in the FPGA of dimensions other than the nominal dimensions using dynamic FPGA reconfiguration, as well as enabling the support to implement and accelerate more algorithms than CCSDS123 issue 1 compression.

The contribution of this paper is to showcase how on-demand reprogramming of FPGA hardware during satellite operation can be beneficial in certain situations, specifically making the use of the FPGA for compression of more capture configurations. We present results from performing FPGA reconfiguration on-board the HYPSO-1 satellite that showcases the benefit this brings.

The rest of the paper is outlined as follows.: Section 2 discusses the proposed design and necessary development for support of FPGA reconfiguration during satellite operations on the HYPSO satellites. Section 3 presents and discusses the results of the implementation. Section 4 concludes the discussion of the results.

2. Implementation

The master thesis [6] covers the development and necessary implementation for FPGA reconfiguration on HYPSO-1 in detail. The following section provides an overview of the motivation and design choices relevant when reconfiguring during operation.

2.1 Motivation

On-board FPGA reconfiguration has the potential to accelerate a wide variety of algorithms and enable operational flexibility. These algorithms can serve different purposes from configurable compression to on-board operational autonomy and decision making. The process of developing an FPGA implementation of such an algorithm and integrating it into the satellite operational procedures is challenging. For this reason, a development and testing workflow has been defined that focuses on simplicity and modularity.

The intended flow for testing new designs on HYPSO-1 is illustrated in Figure 1. It is a collaboration between FPGA developers and satellite operators to make sure a design works as expected on the platform before it is integrated into day-to-day satellite operations.

The flow consists of a developer who creates an FPGA design and integrates it into the satellite project. The *FlatSat* is a functional replica of the satellite, set up at a table. If the design works on target hardware, a satellite operator collaborates with the developer to create a test and run this

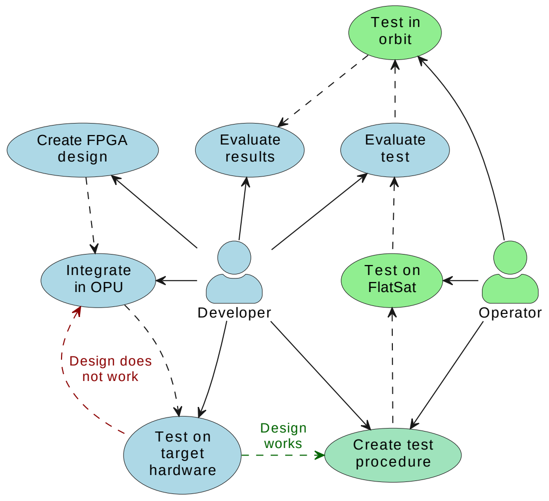

Fig. 1. Suggested flow for testing new FPGA designs

on the FlatSat. If the test is successful, the test can be run on the satellite, and the results evaluated by the developer.

2.2 Reconfiguration methods

Partial dynamic reconfiguration and full static reconfiguration are different methods used to reprogram an FPGA. Partial reconfiguration only rewrites a specified partition on the FPGA and can be done while the rest of the device is still operating. On the other hand, full reconfiguration rewrites the whole FPGA, and designs on the device can not be used during the reconfiguration process. Partial reconfiguration is faster and allows for smaller configuration files, as less of the configuration data is rewritten compared to full reconfiguration [7]. However, partial reconfiguration requires more effort to set up, and operators must be aware of what designs exist outside the partial configuration.

The FPGA used on the OPU supports both partial and full reconfiguration, but full static reconfiguration was chosen as the default method to make the development process and satellite operation simpler compared to partial reconfiguration.

The OPU on HYPSO-1 runs an embedded Linux system, created with the PetaLinux tools provided by AMD Xilinx. By default, a static FPGA configuration is embedded into the operating system image on the OPU, and an operating system image update is necessary to change the default configuration. To facilitate for reconfiguration of the FPGA during runtime, the static configuration is separated from the rest of the operating system and loaded during system startup. This also requires the software interacting with the FPGA to take into account the possibility that a design may not be present.

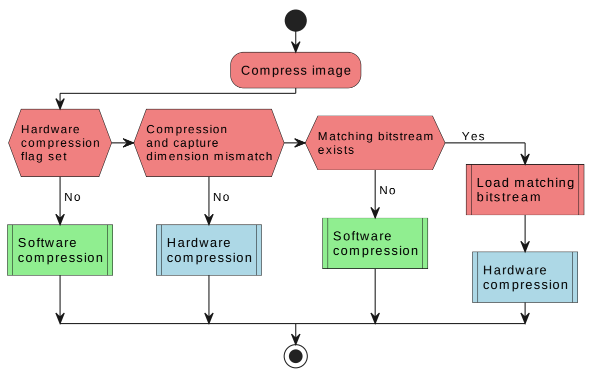

Fig. 2. Flow diagram of hardware compression reconfiguration

2.3 Implementation into operations and processing pipeline

The reconfiguration is integrated into satellite operations by expanding the set of commands received and handled by the OPU. The commands provide satellite operators with a simple interface for operations such as uploading test files, listing available files, reconfiguring the FPGA, and also checking the currently loaded design. The FPGA is reconfigured via an AMD Xilinx application called `fpgautil`, which in turn uses the Linux *FPGA Manager* backend.

Open Firmware Devicetree is a data structure and language used to describe hardware. Devicetree files can be used to describe the hardware configuration in Linux systems, and even change it during runtime [8]. It is possible to load devicetree overlays when the FPGA is reconfigured, describing the new design to the operating system. This is supported and recommended when reconfiguring the FPGA on HYPSO-1, as it makes it easier for software to interact with the FPGA designs via Linux device drivers. An example of this is the Linux *Userspace IO* driver, which can be used with memory-mapped interfaces such as Advanced eXtensible Interface (AXI).

2.4 Usecases

A relevant use case for the FPGA reconfiguration feature is to automatically load a specific FPGA design as part of a processing pipeline. As the non-flexible FPGA algorithm for hyperspectral image compression on HYPSO-1 is much faster than its flexible software equivalent, this is a relevant case for automatic reconfiguration of the FPGA. Automatic reconfiguration is integrated into the code responsible for hyperspectral image compression, following the flow described in Figure 2. The software checks if hardware compression should be used, and if this is the case it checks the devicetree for an existing hardware compression algorithm with the correct dimensions. If this does not exist in the current devicetree, the software searches for a bitstream that matches the required dimensions and loads this one to the FPGA before running the hardware compression. If no acceptable bitstream is found, the software compression is used. Reconfiguring the FPGA before compression of captures with non-standard dimensions utilizes the faster hardware while keeping the compression flexible.

Category	$X \times Y \times Z$
Nominal	$684 \times 956 \times 120$
No binning	$684 \times 106 \times 1080$
Full sensor	$1216 \times 33 \times 1936$
Wider spatial	$1092 \times 598 \times 120$ or $1216 \times 537 \times 120$

Table 1. Categories of dimensions, from [3]

3. Results and Discussion

FPGA reconfiguration during operation was implemented as part of [6], and since 9th March, 2024, it has been used on the HYPSO-1 satellite. An anomaly detection algorithm [9] and a hyperspectral image compression algorithm [10] were tested on HYPSO-1 with FPGA reconfiguration.

The HYPSO-1 satellite operators use different categories for the dimension of the hyperspectral cubes.The most common of these are the *nominal* and the *wider spatial* dimensions. Nominal dimensions are the default dimensions intended for the camera at launch and were initially the only dimensions supported by hardware compression. Wide dimensions is a different set of dimensions that captures a wider spatial area, and before the introduction of reconfiguration on HYPSO-1, these images have been compressed with the software implementation on the CPU. The hardware compression on nominal images takes about 184 ms and the software compression about 398 s [3]. Therefore, the hardware compression saves more than 6 minutes of processing time after an image capture. A long processing (compression) time after each capture is a challenge for satellite operations, as it makes captures of targets that are located geographically close to each other difficult.

With the operating system image update on HYPSO-1 in March 2024, FPGA configurations with hardware compression for the most used image dimensions have been uploaded to the satellite. These are automatically reloaded when the image dimensions of a new recording do not match the dimensions of the currently loaded FPGA con-

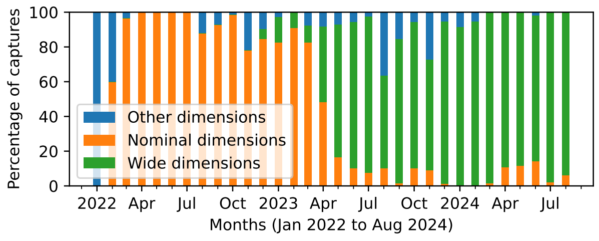

Fig. 3. Percentage of dimensions captured on HYPSO-1

Method	Size [KB]	Mean time [ms]	σ (SD) [ms]
Partial	2852	61.16	1.05
Full	5840	111.62	1.16
Full + Overlay	5840	252.50	4.00

Table 2. Mean time and standard deviation for different reconfiguration methods on OPU

figuration. An analysis of a set of 2316 captures from HYPSO-1 is presented in Figure 3. It shows that between the launch of HYPSO-1 in January 2022 and the image update in March 2024, roughly 54% of the captures were of nominal dimensions, and the rest of these captures used software image compression. After the image update in March, alternative dimensions such as the wide dimensions could also use hardware compression. In the period between 9th March and 31st July 2024, wide dimensions make up more than 90% of the image captures, and dimensions that are not supported by hardware compression make up less than 0.5% of the captures.

The reconfiguration process itself adds some time and will affect the performance. To evaluate the impact of this time, partial reconfiguration, full reconfiguration, and full reconfiguration with devicetree overlay were tested 50 consecutive times each, and the results are shown in Table 2. The time spent will vary based on the size of the configuration files and the speed at which they are loaded. The partial reconfiguration file for this test contained 2852 KB, while the full configuration files contained 5840 KB. The partial reconfiguration method in this test took almost half the time of the full reconfiguration, and the devicetree overlay more than doubled the time used in full reconfiguration. This time must be considered when reconfiguring new FPGA designs for increased performance. For the case of the reconfiguration of the hyperspectral compression algorithm, the hardware algorithm gains more than 6 minutes compared to the software algorithm on nominal dimensions, and this method will gain performance even with the slowest reconfiguration method. The extreme performance increase, in this case, makes the reconfiguration time insignificant in comparison, and choosing the full reconfiguration method with the devicetree overlay for simpler operation is a reasonable trade-off.

4. Conclusion

Supporting reconfiguration during regular satellite operations makes it easier for researchers associated with the HYPSO project to test new FPGA implementations on the satellite. Using devicetree overlays with the FPGA configuration introduced longer configuration times, but also made software interaction with the designs easier due to the use of native device drivers.

As a concrete example, reconfiguration of the FPGA made hardware compression of hyperspectral images more flexible for HYPSO-1, utilizing reconfiguration of the on-board FPGA to compress a larger number of hyperspectral images than before. This saved processing time after image captures and made satellite operations easier while requiring only a fraction of a second more setup time to perform the FPGA reconfiguration. This reconfiguration framework will also be included in the new HYPSO-2 satellite, that was launched in August 2024, depending on a system image update.

References

[1] N. Montealegre, D. Merodio, A. Fernández, and P. Armbruster, "In-flight reconfigurable FPGA-based space systems," in *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2015, pp. 1–8. DOI: 10.1109/AHS.2015.7231177.

[2] S. Bakken *et al.*, "HYPSO-1 CubeSat: First Images and In-Orbit Characterization," *Remote Sensing*, vol. 15, no. 3, 2023, ISSN: 2072-4292. DOI: 10.3390/rs15030755.

[3] D. D. Langer *et al.*, "Robust and Reconfigurable On-Board Processing for a Hyperspectral Imaging Small Satellite," *Remote Sensing*, vol. 15, no. 15, 2023, ISSN: 2072-4292. DOI: 10.3390/rs15153756.

[4] M. Orlandić, J. Fjeldtvedt, and T. A. Johansen, "A parallel fpga implementation of the ccsds-123 compression algorithm," *Remote Sensing*, vol. 11, no. 6, 2019, ISSN: 2072-4292. DOI: 10.3390/rs11060673.

[5] R. Birkeland, S. Berg, M. Orlandic, and J. L. Garrett, "On-Board Characterization Of Hyperspectral Image Exposure And Cloud Coverage By Compression Ratio," in *2022 12th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in*

Remote Sensing (WHISPERS), 2022, pp. 1–5. DOI: 10.1109/WHISPERS56178.2022.9955117.

[6] S. L. Eine, "In-orbit testing of FPGA designs on HYPSO-1," Master thesis, Norwegian University of Science and Technology, 2024.

[7] D. G. Perera, "Analysis of FPGA-Based Reconfiguration Methods for Mobile and Embedded Applications," in *Proceedings of the 12th FPGAworld Conference 2015*, ser. FPGAworld '15, event-place: Stockholm, Sweden, New York, NY, USA: Association for Computing Machinery, 2015, pp. 15–20, ISBN: 978-1-4503-3737-3. DOI: 10.1145/2889287.2889297.

[8] Grant Likely, *Linux and the Devicetree*, 2012. [Online]. Available: https://www.kernel.org/doc/html/latest/devicetree/usage-model.html (visited on 03/28/2024).

[9] S. Boyle, "Design Space Exploration of FPGA Accelerators for Hyperspectral Anomaly Detection," English, M.S. thesis, Norwegian University of Science and Technology, 2023. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3093196.

[10] D. Vorhaug, "Hyperspectral Image Compression Accelerator on FPGA Using CCSDS 123.0-B-2," Master thesis, Norwegian University of Science and Technology, 2024.