IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 19, 2018, accepted January 7, 2019, date of publication January 11, 2019, date of current version January 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2892308

Scalable Hardware-Based On-Board Processing for Run-Time Adaptive Lossless Hyperspectral Compression

ALFONSO RODRÍGUEZ[1], (Student Member, IEEE), LUCANA SANTOS[2], ROBERTO SARMIENTO[3], AND EDUARDO DE LA TORRE[1]
[1]Centro de Electrónica Industrial, Universidad Politécnica de Madrid, 28006 Madrid, Spain
[2]Moltek Ltd. for European Space Agency, 2201 AZ Noordwijk, The Netherlands
[3]Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, 35001 Las Palmas, Spain

Corresponding author: Alfonso Rodríguez (alfonso.rodriguezm@upm.es)

This work has been conducted within the ENABLE-S3 project that has received funding from the ECSEL Joint Undertaking under grant agreement No 692455. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway. This work has also been supported by the Spanish R&D programme under grant PCIN-2015-254. The authors would like to thank the Spanish Ministry of Education, Culture and Sport for its support under the FPU grant program.

ABSTRACT Hyperspectral data processing is a computationally intensive task that is usually performed in high-performance computing clusters. However, in remote sensing scenarios, where communications are expensive, a compression stage is required at the edge of data acquisition before transmitting information to ground stations for further processing. Moreover, hyperspectral image compressors need to meet minimum performance and energy-efficiency levels to cope with the real-time requirements imposed by the sensors and the available power budget. Hence, they are usually implemented as dedicated hardware accelerators in expensive space-grade electronic devices. In recent years though, these devices have started to coexist with low-cost commercial alternatives in which unconventional techniques, such as run-time hardware reconfiguration are evaluated within research-oriented space missions (e.g., CubeSats). In this paper, a run-time reconfigurable implementation of a low-complexity lossless hyperspectral compressor (i.e., CCSDS 123) on a commercial off-the-shelf device is presented. The proposed approach leverages an FPGA-based on-board processing architecture with a data-parallel execution model to transparently manage a configurable number of resource-efficient hardware cores, dynamically adapting both throughput and energy efficiency. The experimental results show that this solution is competitive when compared with the current state-of-the-art hyperspectral compressors and that the impact of the parallelization scheme on the compression rate is acceptable when considering the improvements in terms of performance and energy consumption. Moreover, scalability tests prove that run-time adaptation of the compression throughput and energy efficiency can be achieved by modifying the number of hardware accelerators, a feature that can be useful in space scenarios, where requirements change over time (e.g., communication bandwidth or power budget).

INDEX TERMS Data compression, dynamic and partial reconfiguration, FPGAs, high-performance embedded computing, hyperspectral images, on-board processing.

I. INTRODUCTION

The use of hyperspectral imaging is already consolidated as a key enabling technology in remote sensing applications. However, the continuous improvements in the field still pose several challenges at multiple levels. For instance, enhanced sensor resolution means more data to be processed and/or stored, which in turn increases both computing and memory requirements in the systems. This, together with the limited communication bandwidth between satellites and ground stations, motivates the use of high-efficiency on-board data compression.

Hyperspectral compression algorithms are usually implemented in hardware fabrics (e.g., ASICs or radiation-hardened FPGAs), since they provide high-performance and low-energy solutions [1]. However, these approaches present several drawbacks: on the one hand, they lack flexibility once deployed, since it is impossible to make modifications in the implemented circuit; on the other hand, they rely on expensive space-qualified devices to ensure correct behavior during mission time.

Recently, the appearance of CubeSats has enabled the use of low-cost technology, usually based upon commercial off-the-shelf components, in small satellite deployments [2]. One of the main benefits of this is that CubeSats can be used as testbeds for non-conventional developments targeting space applications. For instance, they allow the evaluation of dynamically and partially reconfigurable FPGAs as a possible replacement for traditional radiation-hardened alternatives [3]. This evaluation includes the use of Dynamic and Partial Reconfiguration (DPR) of SRAM-based fabrics with a twofold objective: on the one hand, to support functional adaptation with software-like flexibility but hardware-like performance; on the other hand, to implement run-time fault mitigation mechanisms such as module relocation or configuration memory scrubbing [4].

In this paper, a run-time scalable hardware-based implementation of a lossless hyperspectral data compression algorithm is presented. The proposed approach relies on a configurable number of low-complexity compressor cores (HyLoC) managed by a DPR-enabled hardware processing architecture (ARTICo^3) operating in the reconfigurable part of a commercial System on Programmable Chip (SoPC).

HyLoC [5] is a hardware-based implementation of the Consultative Committee for Space Data Systems (CCSDS) 123 standard that targets resource-constrained systems, where limiting area overhead is more important than achieving high compression throughput. As such, it has a reduced footprint and features low-power execution at the expense of limited performance. ARTICo^3 [6], on the other hand, is a hardware-based processing architecture that benefits from a multi-accelerator computing scheme to create a run-time adaptive solution space for data-parallel algorithms. This solution space is defined by a tradeoff between computing performance, energy consumption and fault tolerance.

The HyLoC compressor is sequential in nature. In order to make it compatible with the ARTICo^3 execution model, input images are split in fixed-size subimages that are compressed independently. Hence, the implementation is provided with data-level parallelism and can fully benefit from a configurable number of accelerators to achieve better performance (one order of magnitude) at a very low cost in terms of power consumption, logic resources and design/verification times.

To the best of the authors' knowledge, this is the first implementation of a lossless hyperspectral compressor with multiple SIMD-like hardware accelerators that renders run-time adaptive throughput and energy consumption, while still being competitive with alternative state-of-the-art solutions. This novel approach allows the system to compress faster or slower depending on the availability of the downlink bandwidth or the urgency to receive, in the ground station, the information contained in the image. In addition, the proposed image partitioning scheme can complement the inherent redundant operation available in ARTICo^3 to increase the fault tolerance of the compressor (i.e., a single error in one of the compressed bitstreams only prevents that subimage from decoding from that point onwards, instead of compromising the whole input image).

The ARTICo^3-based HyLoC compressor differs from all hardware-based alternatives available in the literature, which typically rely on the use of a single and highly optimized accelerator. These implementations can achieve high throughput (ideally, 1 compressed sample per clock cycle) in BIP mode with an optimized pipeline, although data dependencies force the compressor to run slower in BSQ or BIL modes. However, resource utilization grows significantly when optimizing the cores, due to the amount of internal storage required. Moreover, the throughput may be severely affected if internal resources are not enough and storage needs to be outsourced (e.g., to an external memory attached to the FPGA). Although this last problem cannot be avoided, the parallel execution of several resource-constrained hardware accelerators mitigates the impact on execution performance without increasing area overhead drastically.

The rest of this paper is organized as follows. Section II presents the building blocks used in this work. A discussion on the parallelization approach used to deploy the hyperspectral image compressor on the hardware-based processing architecture is presented in Section III. The proposed implementation is assessed in Section IV, and the conclusions drawn from the results are presented in Section V.

II. TECHNOLOGY BACKGROUND

A. HYLOC: A LOW-COMPLEXITY IMPLEMENTATION OF THE CCSDS 123 STANDARD

The CCSDS 123 standard [7] describes a compression algorithm divided in two stages: a predictor and an entropy coder. The algorithm takes a multispectral or hyperspectral image (i.e., a three-dimensional array of integer samples) as input, and exploits redundancies in both spatial and spectral domains to reduce data volume without compromising its integrity (i.e., the compression is lossless). As a result, the original input samples can be completely recovered from the encoded bitstream that is produced as output.

A simplified flowchart of the algorithm is presented in Figure 1. The predicted sample $\hat{s}_{z,y,x}$, which corresponds to the actual sample $s_{z,y,x}$, is computed using the previously processed samples in a three-dimensional neighborhood that spans the current spectral band as well as the P previous bands (see Figure 2). P is a user-defined parameter that can range between 0 (i.e., no information from previous bands is used for prediction) and 15.

The steps of the algorithm are as follows. First, a local sum (i.e., $\sigma_{z,y,x}$) of the neighboring samples of $s_{z,y,x}$ in the current band is computed. Users can select how this local sum is computed from two alternatives: neighbor-oriented and column-oriented (see Table 1). Then, the local sums are

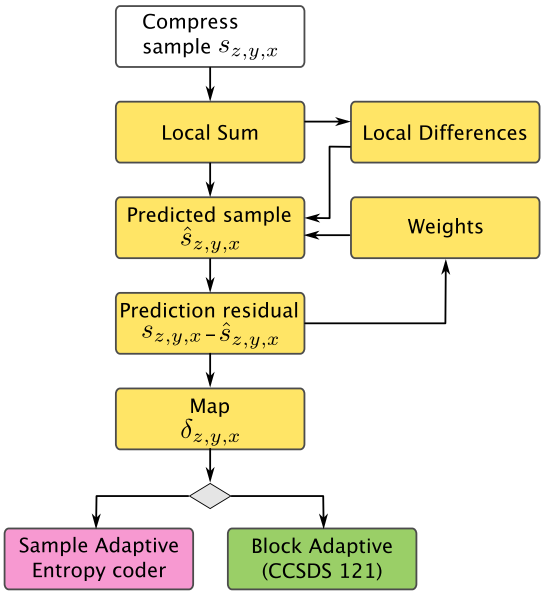

Compress sample $s_{z,y,x}$
Local Sum
Local Differences
Predicted sample $\hat{s}_{z,y,x}$
Weights
Prediction residual $s_{z,y,x} - \hat{s}_{z,y,x}$
Map $\delta_{z,y,x}$
Sample Adaptive Entropy coder
Block Adaptive (CCSDS 121)

FIGURE 1. Flowchart of the CCSDS 123 standard algorithm for multispectral and hyperspectral data compression.

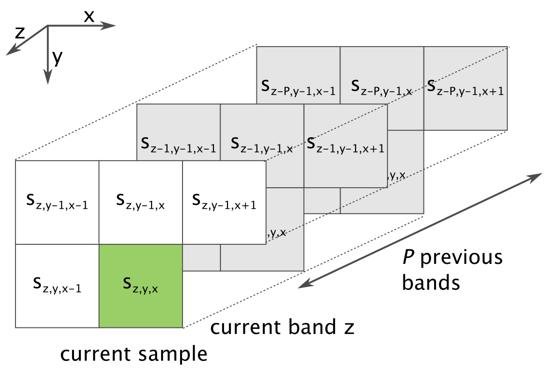


FIGURE 2. Three-dimensional neighborhood used for prediction in the CCSDS 123 standard.

TABLE 1. Equations to calculate the local sum $\sigma_{z,y,x}$.

Local sum $\sigma_{z,y,x}$ of sample $s_{z,y,x}$
Neighbor-oriented
$s_{z,y-1,x-1} + s_{z,y,x-1} + s_{z,y-1,x} + s_{z,y-1,x+1}$
Column-oriented
$4s_{z,y-1,x}$

used to calculate the local differences vector (i.e., $U_{z,y,x}$). The elements of this vector depend on the prediction mode, which is also a user-defined parameter. Hence, for reduced prediction, only central local differences (i.e., $d_{z,y,x}$) are used; for full prediction, on the other hand, both central and directional local differences (i.e., $d^N_{z,y,x}$, $d^W_{z,y,x}$ and $d^{NW}_{z,y,x}$) are required. The exact equations to obtain all values are shown in Table 2.

The predicted sample $\hat{s}_{z,y,x}$ is calculated by performing the dot product of the local differences vector $U_{z,y,x}$ and a weight vector $W_{z,y,x}$ that is updated according to the resulting prediction error (i.e., the difference between prediction and actual sample). Finally, the prediction residuals are mapped to positive integer values $\delta_{z,y,x}$ that are subsequently entropy coded.

TABLE 2. Equations to calculate the elements of the local differences vector $U_{z,y,x}$. For reduced prediction, only central local differences are used; for full prediction, both central and directional local differences are used.

Local differences vector $U_{z,y,x}$
Central local differences
$d_{z-1,y,x} = 4\,s_{z-1,y,x} - \sigma_{z-1,y,x}$
$d_{z-2,y,x} = 4\,s_{z-2,y,x} - \sigma_{z-2,y,x}$
$\dots$
$d_{z-P,y,x} = 4\,s_{z-P,y,x} - \sigma_{z-P,y,x}$
Directional local differences
$d^N_{z,y,x} = 4\,s_{z,y-1,x} - \sigma_{z,y,x}$
$d^W_{z,y,x} = 4\,s_{z,y,x-1} - \sigma_{z,y,x}$
$d^{NW}_{z,y,x} = 4\,s_{z,y-1,x-1} - \sigma_{z,y,x}$

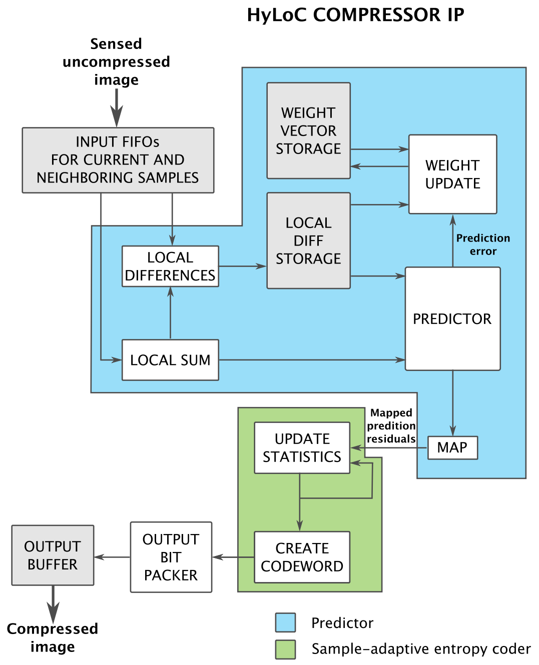


FIGURE 3. Block diagram of the HyLoC IP Core.

The CCSDS 123 standard offers two options for entropy coding: a sample-adaptive and a block-adaptive encoder.

HyLoC [5] is a low complexity VHDL implementation of the CCSDS 123 lossless compressor for multispectral and hyperspectral images that fits in a small space-qualified FPGA. It performs the compression in BSQ order with sample-adaptive entropy coding only, and serializes the computation of the local differences and weight vectors to reduce complexity as much as possible. It is fully compliant with the standard and allows for tuning the the CCSDS 123 parameters at compile time. A simplified block diagram of the HyLoC compressor IP core is presented in Figure 3.

During the HyLoC development, the tradeoffs between compression efficiency and hardware complexity of the implementation were analyzed under several compression scenarios (i.e., exploring using different user-defined parameters). The final hardware architecture designed for HyLoC makes it possible to compress any image in BSQ order without requiring an additional external data memory to store the local differences vector, and it also features the lowest complexity in terms of necessary logic to compute the mathematical operations at the prediction stage. The price to be paid in return for the reduced complexity is an increased latency, as the computation of the dot product between the local differences and the weight vectors is serialized. Nevertheless, the low complexity of the IP core allows for several instances to co-exist in the same FPGA and consequently scale up the throughput. As it will be shown in the following sections, this is one of the main contributions of this paper.

The VHDL description of HyLoC was validated against reference software implementations from ESA [8] and UAB [9]. Moreover, it was synthesized for anti-fuse and SRAM FPGA technologies, including space-qualified devices such as the RTAX1000S. When implemented on a Virtex-5 FX130, the maximum occupancy of resources was 10%, demonstrating its low-complexity nature, and the maximum operating frequency was 134 MHz, providing a throughput of 11.3 MSamples/s.

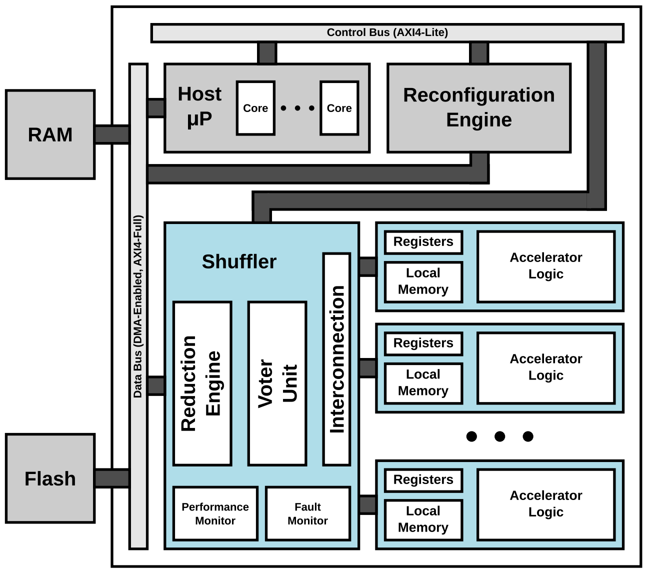


FIGURE 4. Top-level block diagram of the ARTICo^3 architecture.

B. ARTICo^3: A FRAMEWORK FOR HARDWARE-BASED HIGH-PERFORMANCE EMBEDDED COMPUTING

ARTICo^3 [6] is a DPR-enabled hardware-based processing architecture for high-performance embedded systems (see Figure 4 for a top-level block diagram). Its execution model follows a processor-coprocessor approach, where computationally intensive tasks with explicit data-level parallelism (called kernels) are offloaded to hardware accelerators. Moreover, one of the main features of ARTICo^3-based processing is that the pool of hardware accelerators is dynamic: DPR can be used to either achieve flexible functional adaptation (i.e., time-multiplexing application-specific logic in the FPGA), or to enable module replication (i.e., loading one or more copies of the same accelerator).

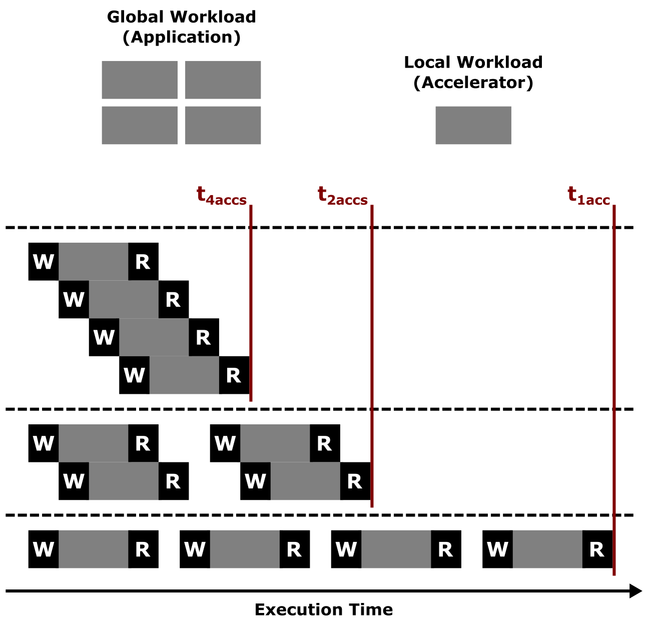


FIGURE 5. ARTICo^3 execution model. User applications offload a global workload to a reconfigurable number of hardware accelerators, each one being capable of processing a local workload. Black boxes represent DMA transfers to (Write) or from (Read) accelerators.

In addition, the internal datapath of the architecture can be configured at run time to support different operation modes. Hence, several instances of the same hardware accelerator can be used to execute a kernel in SIMD-like fashion, in a fault-tolerant manner using module redundancy and a voter unit, or in a combination of both modes. This configurability creates, for each kernel, a dynamic solution space defined by run-time tradeoffs between not only computing performance and fault tolerance, but also energy consumption. Furthermore, and given the data-parallel nature of the kernels, the architecture provides transparent performance scalability when dynamically changing the number of effective kernel-specific accelerator instances operating in SIMD-like fashion. An effective accelerator instance is either a single accelerator operating in non-redundant mode or a group of accelerators operating in redundant mode (i.e., 2 in DMR, 3 in TMR). Figure 5 illustrates performance scalability in the ARTICo^3 execution model: given a global workload, which is defined in the user application running in the host microprocessor, the time it takes for the offloaded computation to finish depends on the number of effective accelerator instances and their processing capabilities (i.e., the local workload they can accept). In the example, a global workload of 4 units requires 1 execution round with 4 accelerators, 2 execution rounds with 2 accelerators, or 4 execution rounds with 1 accelerator.

The ARTICo^3 framework uses the architecture as its core element, but provides additional components at both design and run time. At design time, the ARTICo^3 toolchain eases the integration of user-defined accelerator logic into a run-time reconfigurable computing system. An automated flow builds the multi-accelerator setup and generates the required FPGA configuration files from the kernel specifications provided by the developer. These descriptions can be done either in low-level RTL code (e.g., VHDL, Verilog) or in high-level C/C++ code. Hence, hardware engineers can work with performance-oriented and resource-efficient accelerators, whereas software engineers can also work with ARTICo^3 by leveraging High-Level Synthesis (HLS) engines. At run time, the ARTICo^3 runtime library makes both reconfiguration and execution management transparent for developers, relying on a user-friendly API called from the code running on the host processor. The application executable is also built by the ARTICo^3 toolchain from a C/C++ specification, making it mandatory to provide an already partitioned hardware/software application as input.

III. PARALLELIZATION APPROACH

The original HyLoC compressor was conceived to have a small footprint in terms of logic resources and, as a result, its normal operation is eminently sequential. Moreover, and due to the adaptive nature of the compression algorithm itself, it was meant to compress relatively large hyperspectral images. This approach, although reasonable in some contexts (e.g., small rad-hard FPGAs or expensive ASICs), is no longer valid in an ARTICo^3-based implementation, because no data-level parallelism can be extracted and exploited.

According to Section 2.1 of the CCSDS 123.0 Recommended Standard [7]: *A user may choose to partition the output of an imaging instrument into smaller images that are separately compressed (...). This Recommended Standard does not address such partitioning or the tradeoffs associated with selecting the size of images produced under such partitioning.* This is further described in Section 5.3.2 of the CCSDS 120.2 Informational Report [10]: *The Recommended Standard does not directly address such image segmentation; the term "segment" is not part of the standard. In the view of the Recommended Standard, each such image segment is simply a separate image.*

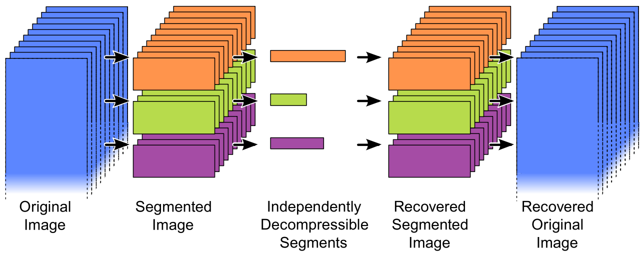

Original Image
Segmented Image
Independently Decompressible Segments
Recovered Segmented Image
Recovered Original Image

FIGURE 6. **Extracted data-level parallelism (from CCSDS 120.2 Informational Report [10]). In this example, *image segments are produced by partitioning a larger image along the y-axis. This is a natural approach for imagers that produce data in BIP or BIL order, but other partitioning approaches could also be used.***

Hence, extracting data-level parallelism by splitting the image in several segments (as shown in Figure 6) is compliant with the reference standard, as long as each partition is handled as an independent image. Notice, however, that the document mentions certain tradeoffs that need to be taken into account when applying partitioning. These tradeoffs involve sacrificing the compression rate to, for instance, achieve fault tolerance (e.g., minimize the effects of data corruption in communication links), or to limit the maximum size of each compressed image [7]. By using ARTICo^3, computing performance (algorithm speedup) and energy efficiency become part of these tradeoffs, since they are also considered when deciding the size and number of segments that are to be compressed independently.

In any case, hyperspectral image partitioning usually leads to a reduction in the compression rate for several reasons:

- Each segment requires a header, since it is handled as an independent image.
- Prediction is limited at the boundaries between segments (less neighbors).
- Small segments mean less data to "learn" from and therefore, less adaptivity in the compressor.

Although the limits might change depending on the application scenario, variations in the compression rate need to be bounded. It is important to analyze the whole constraint set in order to assess the feasibility of any given partitioning. This is even more important in the proposed approach, where low-level, physical restrictions derived from the DPR-based design flow are present. For this reason, an in-depth analysis on the impact of different partitioning strategies on the compression rate is presented below. It is important to highlight that, although this analysis is particularized throughout the discussion to the data-parallel implementation of HyLoC in ARTICo^3, the proposed partitioning strategies are platform-agnostic and hold valid for CPUs, GPUs and hardware accelerators on FPGAs.

TABLE 3. **CCSDS 123 compressor configuration.**

Parameter	Value
Encoding Order	BSQ
Bands for Prediction (P)	3
Local Sum Mode	Neighbor-Oriented
Prediction Mode	Full Prediction

The configuration parameters used in the CCSDS 123 compressor for all tests can be found in Table 3. Although the HyLoC core supports more customization parameters, only these have an actual impact on compression performance and area overhead [5]. With this setup, two schemes have been evaluated using a calibrated AVIRIS [11] image with 224 spectral bands and different spatial dimensions: on the one hand, a strip-based partitioning over the y-axis; on the other hand, a square-based partitioning over both x-axis and y-axis.

The strip-based partitioning (see Figure 7), which is the same one presented in [10], shows convergence in the compression rate for subimages that are fairly large. Small images render worse compression rates, especially when partitioned, since the compression algorithm does not have enough input samples to fully exploit its adaptive core. This partitioning

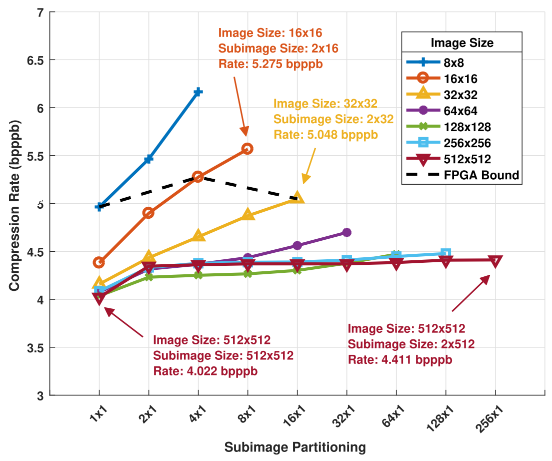


FIGURE 7. Compression Rate: evaluation when dividing the original image (calibrated AVIRIS image with 224 bands, different spatial dimensions) in strips.

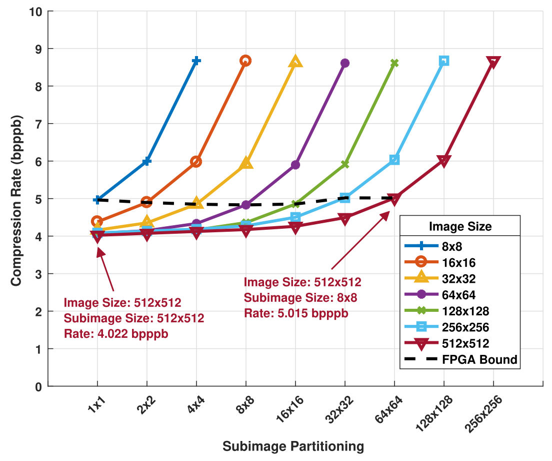


FIGURE 8. Compression Rate: evaluation when dividing the original image (calibrated AVIRIS image with 224 bands, different spatial dimensions) in squares.

scheme is not uniform, since the subimage size changes from smaller images to larger ones.

The square-based partitioning (see Figure 8), on the other hand, shows strong deviations in the compression effectiveness due to the partitioning being uniform (i.e., subimages have fixed size regardless of the actual input image size). In this scenario, the impact of small subimages is larger, since there are even less input samples for the compression algorithm to reach efficient results. However, the deviation in the compression rate remains within an acceptable range when the size of the subimages is increased (e.g., from 4.022 to 5.015, around 25%, for a spatial resolution of 8 × 8 pixels).

Although the strip-based partitioning is better than the square-based one when compressing large input images, its implementation leads to increased memory footprints that are incompatible with ARTICo^3 accelerators. Before building the reconfigurable system, the ARTICo^3 toolchain takes the input kernel specifications and places them in a parametric wrapper that provides seamless integration with the rest of the hardware infrastructure. Within this wrapper, the user logic is attached to a local memory with a size of up to 64 KiB. The dashed line in both Figure 7 and Figure 8 represents this resource limitation for both partitioning schemes (points above the line denote realizable solutions). Note that, due to its lack of scalability, the strip-based approach becomes no longer feasible in the region where better compression results are achieved. As a result, the square-based partitioning scheme, which provides bounded results regardless of the input image and thus favors scalability, has been selected for implementation in an ARTICo^3-based deployment.

TABLE 4. Compression Rate: single image versus parallel partitioning. All reference images are calibrated AVIRIS samples. The size of all subimage blocks is 224 × 8 × 8.

Image	Nz×Ny×Nx	Rate (bpppb) Single	Rate (bpppb) Parallel
Indian Pines	220×152×152	6.3	7.07
Yellowstone SCN 0	224×512×512	3.98	4.98
Yellowstone SCN 3		3.87	4.86
Yellowstone SCN 10		3.37	4.28
Yellowstone SCN 11		3.68	4.61
Yellowstone SCN 18		3.97	5.09

At this point, it is important to highlight that compression effectiveness is data-dependent (i.e., is not only affected by the specific partitioning of the hyperspectral image, but also by the image itself). This is due to the adaptive nature of the compression algorithm, and further discussion in this regard can be found in [10], where it is also possible to see certain convergence in the compression effectiveness provided that subimages are large enough. Hence, additional tests have been carried out using well-known datasets as input images (i.e., Indian Pines [12], Yellowstone Scenes [13]). Table 4 presents the compression rates achieved using subimage blocks of 224 × 8 × 8, which is the maximum size for ARTICo^3-based implementations. Please note that spatial dimensions have been extended (Indian Pines) or reduced (Yellowstone Scenes) to achieve correct partitioning with an integer number of blocks. Results show that the worst case only incurs in 28% overhead in the compression rate for a spatial resolution of 8 × 8 pixels, a value that is consistent with the previous tests.

IV. EXPERIMENTAL RESULTS

Once the feasibility of the data-parallel approach has been assessed, the next step is to evaluate the run-time adaptation and scalability features of the ARTICo^3-based HyLoC implementation. To this end, a Zynq MMP development board featuring an XC7Z100-2FFG900 Zynq-7000 device has been used. This decision has been motivated by two key factors: on the one hand, Zynq-7000 devices provide better embedded processor performance than any other Xilinx FPGA (dual-core ARM Cortex-A9 versus single-core MicroBlaze); on the other hand, the selected device features the largest number of logic resources in the family, which allows for better accelerator scalability. A hardware template has been

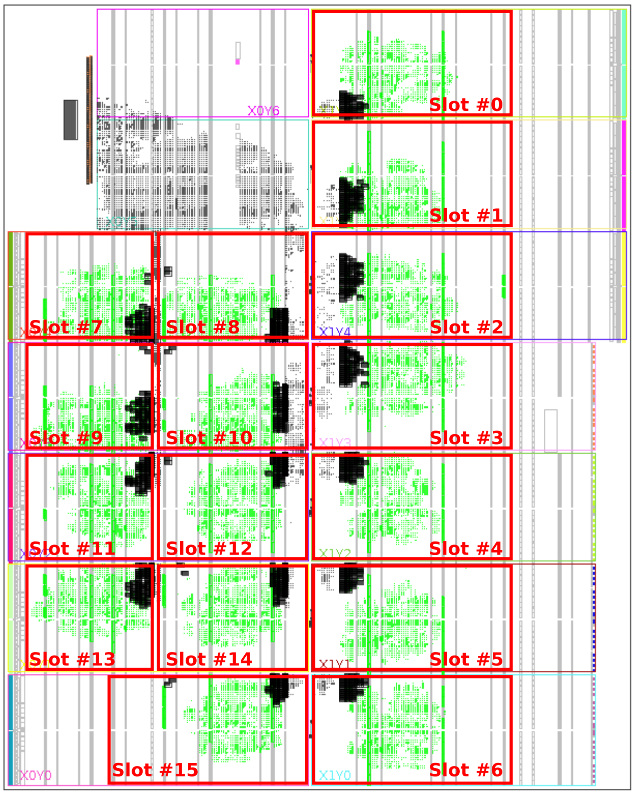


FIGURE 9. Experimental setup: 16 HyLoC accelerators (green) connected to the ARTICo^3 infrastructure (black). Reconfigurable slots have been highlighted in red for clarification.

TABLE 5. Resource Utilization: ARTICo^3 infrastructure, HyLoC kernel, and reference implementation.

Component	ARTICo^3	HyLoC Kernel	Baseline HyLoC [5]
Info	–	64 KiB 0 registers VHDL (Vivado) Zynq-7000	– – VHDL (ISE) Virtex-5
LUTs	4158	2932	2342
FFs	2366	1529	1535
DSPs	–	1	1
BRAMs	–	16	–

defined to support this board in the ARTICo^3 toolchain, enabling up to 16 reconfigurable slots to load kernel-specific hardware accelerators. Figure 9 shows the FPGA layout after implementation, with the ARTICo^3 architecture and all HyLoC instances. Moreover, an ad-hoc port of the ARTICo^3 runtime library for bare-metal applications has been used instead of the standard Linux-based version, aiming at improving computing performance and execution scalability by reducing memory management overheads.

Table 5 shows the resource utilization of the ARTICo^3 infrastructure, one instance of a HyLoC kernel, and the reference implementation [5]. Notice that, although the synthesis tool (ISE versus Vivado) and the FPGA used (Virtex-5 versus Zynq-7000) are not the same, the area overhead introduced by the ARTICo^3 wrapper is almost negligible (except for the local memory storage). Also note that, when scaling up

TABLE 6. Estimated power consumption.

	Power (W)
FPGA Static	0.298
Processing System	1.578
HyLoC	0.04 ($\times$16)
ARTICo^3	0.08
Additional Logic	0.021
Total	**2.617**

the number of HyLoC accelerator instances, the ARTICo^3 infrastructure becomes a small fraction of the overall resource utilization (e.g., 8% with 16 accelerators).

The Zynq MMP development board does not have power measurement circuitry and therefore, only estimated values have been used to evaluate the energy efficiency of the proposed implementation. However, the values obtained using Vivado power estimator are coherent with actual power traces obtained in other ARTICo^3 deployments where on-board measuring capabilities were present [6]. Table 6 reports the estimated power consumption values for both individual components and complete system.

The evaluation of computing performance, execution scalability and energy efficiency of the ARTICo^3-based HyLoC implementation has been done using as input hyperspectral image a calibrated AVIRIS capture with 224 bands, 512 lines and 512 samples. Each of the hardware accelerators has been configured with the maximum subimage size (as discussed in Section III), which is 224 bands, 8 lines and 8 samples, and operates at a clock frequency of 100 MHz. Table 7 shows the theoretical results (i.e., assuming zero on-chip communication and memory access overheads, and knowing that one ARTICo^3-compliant HyLoC takes 185555 clock cycles to compress one subimage), and the values obtained when using DPR to dynamically change the pool of available processing cores in a real implementation. These results correspond to a scenario where no module redundancy is used, but the same values would be obtained when using the same number of effective accelerator instances. In this context, execution time shows almost linear scalability when using up to 8 accelerators, proving that they operate in a computing-bounded region (i.e., accelerator execution takes more time than data transfers between memories), which is the preferred scenario to fully benefit from ARTICo^3-based acceleration. However, using 16 accelerators only renders 9.7$\times$ speedup, which indicates that the system is entering the memory-bounded region (i.e., memory bandwidth reaches its saturation point). In any case, all configurations are compliant with the CCSDS specification in terms of energy efficiency, even though some of them are below the required threshold for compression throughput. Please bear in mind that the specification states that a compressor should reach 20 MSamples/s at less than 0.5 W/(MSamples/s) [10].

Finally, Table 8 shows a comparison of the proposed implementation with different alternatives from the state of the art. In addition, reference values for AVIRIS sensors and the CCSDS reference standard are also included in the

TABLE 7. Compression throughput and energy efficiency when compressing a calibrated AVIRIS image with 224 bands, 512 lines and 512 samples using ARTICo^3 and a variable number of HyLoC accelerators (configured to compress subimage blocks of 224 bands, 8 lines and 8 samples, FPGA fabric @ 100 MHz).

Accelerators	Rounds	Theoretical Throughput MSamples/s	Actual Throughput MSamples/s	Energy Efficiency mW/(MSamples/s)
1	4096	7.72	6.92	378.18
2	2048	15.45	13.28	197.1
4	1024	30.9	24.52	106.73
8	512	61.81	42.48	61.61
16	256	123.62	67.04	39.1

TABLE 8. Comparison of CCSDS 123 implementations according to the encoding order, bands used for prediction (P), target device, maximum operating frequency, throughput and energy efficiency.

Implementation	Order	P	Device	f_{max} MHz	Throughput MSamples/s	Energy Efficiency mW/(MSamples/s)
AVIRIS [11]	–	–	Sensor	–	1.7	–
AVIRIS-NG [14]	–	–	Sensor	–	42.29	–
CCSDS [10]	–	–	Reference standard	–	20	500
Emporda [9]	All	0 – 15	Software (i7-7500U)	–	4.93 [15]	–
HyLoC [5]	BSQ	3	Virtex-5	134	11.3	207.52
SHyLoC [16]	All	3	Virtex-5	–	140	–
Tsigkanos et al. [1]	BIP	3	Virtex-5	213	213	22.6
Báscones et al. [15]	All	0 – 15	Virtex-7	50	47.62	9.45
Fjeldtvedt et al. [17]	BIP	15	Zynq-7000	147	147	2
Davidson et al. [18]	BIP	3	Jetson TX1	–	116.2	86.06
This work	BSQ	3	Zynq-7000	100 (134)	67.04 (165.65)	39.1 (15.8)

list to highlight real-time processing conditions. Since some solutions do not consider the communication and memory access overheads to calculate the throughput, both actual and theoretical values are provided for the proposed approach (for the latter, the maximum operating frequency of a single HyLoC instance has been used). It is possible to see that the ARTICo^3-based HyLoC compressor renders results in the range of the rest of implementations, being only 3.2× slower (considering actual throughput values) than the fastest solution but flexible at run time, a unique feature of this work. The GPU-based implementation presented in [18] is the only one that also uses input image partitioning (referred to as tiling). However, the implemented partitioning scheme is the strip-based one, and the execution is performed using a fixed number of processing elements (GPUs cannot dynamically change its computing resources). Therefore, it cannot dynamically adapt its performance to changing requirements as the implementation proposed in this work. Regarding other hardware-based alternatives in the literature, the most common approach is to generate highly optimized application-specific accelerators, as opposed to ARTICo^3, which has a more general purpose (i.e., the same architecture can be used to accelerate different algorithms). For instance, in [1], a combination of task-level parallelism and a reconfigurable fine-grained pipeline is used to achieve one of the fastest implementations of the CCSDS 123 standard up to date. In addition, these implementations usually rely on design-time customization of the hardware accelerators, as opposed to ARTICo^3, where DPR allows dynamic changes at run time. This can be seen in [15], where the authors hint that several logic synthesis iterations can be run to fine-tune the parameters of the compressor to the hyperspectral sensor that is used.

V. CONCLUSIONS

In this paper, a novel run-time scalable implementation of the CCSDS 123 standard has been presented. As opposed to other alternatives from the literature, the proposed approach relies on a DPR-enabled multi-accelerator approach, where a configurable number of low-complexity and resource-efficient compressor cores operate in SIMD-like fashion, exploiting data-level parallelism to achieve competitive performance values.

The proposed parallelization approach is supported by a platform-agnostic hyperspectral image partitioning scheme that is compatible with the reference standard. Moreover, the impact of this partitioning on the compression rate is acceptable even if additional low-level constraints derived from the hardware implementation are taken into account (e.g., compression rate does not exceed a 30% overhead when using subimage sizes of 224 bands, 8 lines and 8 samples).

Experimental results show that adaptation is supported at run time, enabling different performance/energy levels by switching the number of hardware accelerators. Performance scalability is almost linear until memory bandwidth starts to saturate (e.g., using 16 accelerators reduces compression time around 10×), and although some configurations do not meet the reference standard requirements in terms of throughput, energy efficiency is always below the required threshold.

Furthermore, the competitiveness of the proposed implementation when compared with other alternatives from the literature encourages the increase of research activity around the use of commercial off-the-shelf devices with run-time reconfiguration capabilities in low-cost space applications.

REFERENCES

[1] A. Tsigkanos, N. Kranitis, G. A. Theodorou, and A. Paschalis, "A 3.3 Gbps CCSDS 123.0-B-1 multispectral & Hyperspectral image compression hardware accelerator on a space-grade SRAM FPGA," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2018.2854412.

[2] M. N. Sweeting, "Modern small satellites-changing the economics of space," *Proc. IEEE*, vol. 106, no. 3, pp. 343–361, Mar. 2018.

[3] A. D. George and C. M. Wilson, "Onboard processing with hybrid and reconfigurable computing on small satellites," *Proc. IEEE*, vol. 106, no. 3, pp. 458–470, Mar. 2018.

[4] A. Pérez, L. Suriano, A. Otero, and E. de la Torre, "Dynamic reconfiguration under RTEMS for fault mitigation and functional adaptation in SRAM-based SoPCs for space systems," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Jul. 2017, pp. 40–47.

[5] 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," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 757–770, Feb. 2016.

[6] A. Rodríguez, J. Valverde, J. Portilla, A. Otero, T. Riesgo, and E. de la Torre, "FPGA-based high-performance embedded systems for adaptive edge computing in cyber-physical systems: The ARTICo^3 framework," *Sensors*, vol. 18, no. 6, p. E1877, 2018.

[7] *Lossless Multispectral and Hyperspectral Image Compression*, Recommended Standard CCSDS 123.0-B-1, Consultative Committee for Space Data Systems (CCSDS), May 2012.

[8] European Space Agency. *Data Compression Tools*. Accessed: Nov. 14, 2018. [Online]. Available: https://www.esa.int

[9] Group on Interactive Coding of Images, Universitat Autònoma de Barcelona. *Emporda*. Accessed: Nov. 14, 2018. [Online]. Available: http://www.gici.uab.es

[10] "Lossless multispectral and hyperspectral image compression," Consultative Committee Space Data Syst., Inf. Rep. CCSDS 120.2-G-1, Dec. 2015.

[11] NASA. *Airborne Visible InfraRed Imaging Spectrometer*. Accessed: Nov. 14, 2018. [Online]. Available: https://aviris.jpl.nasa.gov

[12] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe. (Sep. 2015). *220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3*. [Online]. Available: https://purr.purdue.edu/publications/1947/1

[13] A. B. Kiely and M. A. Klimesh, "Exploiting calibration-induced artifacts in lossless compression of hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 8, pp. 2672–2678, Aug. 2009.

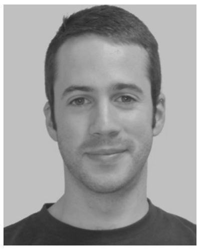
[14] NASA. *Airborne Visible InfraRed Imaging Spectrometer Next Generation*. Accessed: Nov. 14, 2018. [Online]. Available: https://aviris-ng.jpl.nasa.gov

[15] D. Báscones, C. González, and D. Mozos, "FPGA implementation of the CCSDS 1.2.3 standard for real-time hyperspectral lossless compression," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 4, pp. 1158–1165, Apr. 2018.

[16] Universidad de Las Palmas de Gran Canaria. (2017). *SHyLoC IP Core*. Accessed: Nov. 14, 2018. [Online]. Available: https://www.esa.int

[17] J. Fjeldtvedt, M. Orlandic, and T. A. Johansen, "An efficient real-time FPGA implementation of the CCSDS-123 compression standard for hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 10, pp. 3841–3852, Oct. 2018.

[18] R. L. Davidson and C. P. Bridges, "Error resilient GPU accelerated image processing for space applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 9, pp. 1990–2003, Sep. 2018.



ALFONSO RODRÍGUEZ (S'15) received the B.Sc. degree in industrial engineering and the M.Sc. degree in industrial electronics from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2012 and 2014, respectively.

He is currently pursuing the Ph.D. degree in industrial electronics with the Centro de Electrónica Industrial, UPM, where he is currently a Full-Time Researcher. Under a HiPEAC collaboration grant, he was a Visiting Researcher with the Computer Engineering Group, Universität Paderborn, where he worked on operating systems for reconfigurable computing. His current research interests include artificial intelligence, high-performance embedded systems, and reconfigurable computing.

LUCANA SANTOS received the telecommunication engineering degree from the University of Las Palmas de Gran Canaria, in 2008, and the Ph.D. degree from the Integrated System Design Division, IUMA, in 2014.

She was a Visiting Researcher with the European Space Research and Technology Centre, The Netherlands. Since 2018, she has been with the Data Systems and Microelectronics Division, European Space Agency. She is currently a member of the CCSDS Multispectral/Hyperspectral Data Compression working group. She has participated actively in industrial projects in the field of hardware architectures for hyperspectral and multispectral image compression on GPUs and FPGAs for Thales Alenia Space España and the European Space Agency. She has co-authored several scientific papers and has been a Reviewer of major international journals in her research areas. Her current research interests include hardware architectures for on-board data processing, reconfigurable architectures, and hardware/software co-design methodologies.

ROBERTO SARMIENTO is currently a Full Professor of electronic engineering with the Electronics and Telecommunication Engineering School, University of Las Palmas de Gran Canaria, Spain. He contributed to set the Electronics and Telecommunication Engineering School, where he was the Dean of the Faculty, from 1994 to 1998, and was the Vice Chancellor for Academic Affairs and Staff with ULPGC, from 1998 to 2003. He is a co-founder of the Research Institute for Applied Microelectronics, where he is also the Director of the Integrated Systems Design Division. He has published more than 70 journal papers and more than 160 conference papers. He has participated in more than 50 projects and research programmes funded by public and private organizations. His current research interest includes electronics system on-board satellites. He received five six-year research periods from the National Agency for the Research Activity Evaluation in Spain.

EDUARDO DE LA TORRE received the Ph.D. degree in electrical engineering from Universidad Politécnica de Madrid (UPM), in 2000. He is currently an Associate Professor in electronics with UPM, Spain, doing research with the Centre of Industrial Electronics. He has participated in more than 40 projects, eleven of them being EU funded projects and, overall, in nine funded projects related with reconfigurable systems. His main expertise is in FPGA design, embedded systems design, HW acceleration, signal processing, and partial and dynamic reconfiguration of digital systems. He has been the Program Chair of ReConFig and the General Chair of ReCoSoC, two conferences with strong interest in hardware reconfiguration.