

Implementation of CCSDS Standards for Lossless Multispectral and Hyperspectral Satellite Image Compression

LUCANA SANTOS 
ANA GÓMEZ
ROBERTO SARMIENTO 

University of Las Palmas de Gran Canaria, Las Palmas, Spain

This paper presents the modeling, design, and implementation of two intellectual property (IP) cores that are compliant with the consultative committee for space data systems (CCSDS) 121.0-B-2 and CCSDS 123.0-B-1 lossless satellite image compression standards. The CCSDS 121.0-B-2 describes a lossless universal compressor based on a Rice adaptive encoding. The CCSDS 123.0-B-1 standard describes a lossless algorithm specifically designed for efficient on-board compression of hyperspectral and multispectral images, and it is based on a prediction and entropy-based encoding structure. Two options are offered for the latter: the sample-adaptive and the block-adaptive encoder, which corresponds to the CCSDS 121.0-B-2 algorithm. These IP cores have been designed as independent compressors, but they can be easily combined in a plug-and-play fashion to be used together thanks to a dedicated interface. Additionally, standard interfaces are provided for configuration and external memory access. The design process encompasses the consideration of several different hardware architectures in order to maximize throughput and optimize the requirements of on-board resources at the same time. Both IPs are compliant with the high degree of configurability considered in the standard. The obtained VHDL code is completely technology independent, so it can be used to target any field-programmable gate array (FPGA) or ASIC of interest in the space environment, aiming to perform efficiently compression in satellites despite the inherent

Manuscript received June 16, 2017; revised July 24, 2018, March 5, 2019, and May 20, 2019; released for publication June 18, 2019. Date of publication July 23, 2019; date of current version April 10, 2020.

DOI. No. 10.1109/TAES.2019.2929971

Refereeing of this contribution was handled by T. Vladimirova.

This work was supported by the European Space Agency under Project ESA ITT AO/1-8032/14/NL/AK “CCSDS Lossless Compressor for Space Applications”.

Authors’ address: L. Santos, A. Gómez, and R. Sarmiento are with the University of Las Palmas de Gran Canaria, 35001 Las Palmas, Spain, E-mail: (lucana.santos@esa.int; lsfalcon@iuma.ulpgc.es; roberto@iuma.ulpgc.es). (*Corresponding author: Ana Gómez.*)

0018-9251 © 2019 IEEE

constraints of these devices. Results are reported as implementation results for the FPGA devices that are currently considered most representative and suited for on-board use from Xilinx and Microsemi families. In particular, the CCSDS123 IP core reaches for the Virtex5 FX130 a throughput of 153.5 Msamples per second, comprising only 5% of the look-up table (LUTs) in the device.

I. INTRODUCTION

Hyperspectral images are formed by capturing light reflectance in hundreds of different wavelengths. When processed, the extracted information provides data that is essential in the remote sensing field. The amount of information hyperspectral images contain tends to increase, as the resolutions of the sensors placed in satellites grows, leading to an ever larger amount of data. This substantial load of information may be a problem when considering that the on-board storage and the downlink bandwidth are limited on a satellite. For these reasons, reducing the data volume by means of compression has become almost mandatory in many on-board missions, such as GAIA [1] or PROBA-V [2] among many others.

This is the reason for the ever increasing number of research activities in the area of multispectral and hyperspectral compression. As many applications are related to compression on-board satellites or for airborne sensors, in this research paper two main objectives have been pursued: high compression ratio in order to meet the available bandwidth and low implementation complexity. As the processing power on a satellite is limited (even more in the current trend to use small satellites for many missions [3]), the traditional data compression algorithms need to be simplified or otherwise they cannot be applied to space data systems.

Most of the hyperspectral compression algorithms comprises three stages: decorrelation, quantization, and entropy coding. In the state-of-the-art algorithms, decorrelation of information is mostly performed by either transforms or prediction techniques. Transform-based methods are the preferred methods for image [Two-dimensional (2-D)] compression (e.g., JPEG2000 [4]), as they get better compression ratios for lossless, near-lossless or lossy compression. Applying these methods to multispectral and hyperspectral images (3-D) requires the extension of the decorrelation procedure to the spatial-spectral domain. Applying the Karhunen–Loeve transform (KLT) in the spectral dimension has demonstrated to provide an increased rate-distortion performance [5], [6]. The pairwise orthogonal transform has been proposed [7] to overcome the limitations and complexity of the KLT. In addition, principal component analysis [8], and other orthogonal transforms, which are less hardware demanding, such as [9] have been proposed for spectral decorrelation.

Transform based as opposed to prediction-based decorrelation methods are preferred in general for lossy compression. Nevertheless, their application on-board satellites is highly complex mainly because they require significant memory and computational resources. Taking into account the increased use of field-programmable gate array (FPGA) technologies for space applications

[10]–[12], the predictive coding techniques represents the best way to achieve a good compromise between performance (compression ratio) and complexity. Generally, predictive methods are preferred for lossless compression on satellites. Prediction techniques are based on the possibility to make a prediction of the current sample using previous samples. This prediction result is finally encoded on the entropy encoder stage, which explores the probability of the symbols in such a way that long codewords are used to represent symbols with low probability, and short codewords are utilized to represent the most frequent symbols.

The consultative committee for space data systems (CCSDS) is a consortium made up of the world's leading space agencies. It has recommended three different standards for data compression in space: a universal lossless compression solution, consultative committee for space data systems (CCSDS) 121.0-B-2 standard [13]; a lossy or lossless compression standard for 2-D images, CCSDS 122.0-B-1 standard [14]; and finally, a lossless compression standard for multispectral and hyperspectral images, CCSDS 123.0-B-1 standard [15]. The CCSDS 121.0-B-2 and CCSDS 123.0-B-1 standards are the basis for the work hereof. The algorithms described by these standards are able to reduce the data volume by means of removing redundancies in the data source and encoding. Reduction does not compromise data integrity, hence the exact same information is recovered after the decompression. Detailed performance results and comparisons of the CCSDS123 algorithm with other algorithms can be found in the scientific literature (e.g., [16]–[18])

The CCSDS 121.0-B-2 standard defines an entropy encoder that utilizes the Rice encoding [19] technique and an optional prediction-based decorrelator to work over blocks of data. The number of possible encoding options is variable and depends on several configuration parameters, such as input data resolution. The code option that yields the shortest encoded length for the current block is selected for transmission. CCSDS 123.0-B-1 addresses 3-D images (multispectral or hyperspectral). It is based on a prediction and entropy-encoding scheme. Prediction deals with the spatial and spectral redundancy whereas entropy-encoding manages redundancy of the information itself. The CCSDS 123.0-B-1 standard offers two different options for the entropy encoding: a sample-adaptive encoder, which uses Golomb power-of-two codes; and the entropy encoder defined by the CCSDS 121.0-B-2 standard, the block-adaptive encoder.

Compression algorithms themselves are the data and control-flow intensive. Efficiently computing these algorithms on satellites is an important challenge considering also the specific requirements and technology limitations of the space environment. Efforts to optimize performance and to obtain low power consumption must be done, so usually algorithms in on-board systems are implemented on FPGAs or application-specific integrated circuit (ASIC). Low power consumption, tolerance to high energy radiation and high throughput are valued features for solutions in the space. Thus, ideal systems shall be based on low-

complexity and high-throughput hardware architectures efficiently implemented in the specific technologies suited for the space environment.

Implementing the CCSDS 121.0-B-2 and 123.0-B-1 standards in space-qualified FPGAs or ASIC is a challenging task, due to the complexity of the algorithms, and the inherent limitations of the available hardware devices for space. Nevertheless, the public availability of the CCSDS standards has motivated the implementation of reusable IP cores, such as [20], which presents a reconfigurable implementation of the first issue of the block-adaptive encoder, CCSDS 121.0-B-1, together with a predictor in a LEON-based system-on-chip (SoC). Another implementation of the CCSDS 121.0-B-1 can be found in [21].

An additional difficulty for a hardware implementation is that the algorithms are highly configurable, offering a fair amount of parameters that can be chosen by the users to achieve the best possible compression performance for their particular application. This high configurability often forces implementers to reduce the amount of selectable parameters, because the choice of specific parameters can lead to more complex hardware implementations or reduce the maximum achievable throughput. By limiting the amount of options it is possible to optimize the final hardware for specific features. In [22], the original fast lossless, precursor of the CCSDS 123.0-B-1 presents an implementation in Virtex 5, restricting the pixel order to band-interleaved per pixel only; the implementation presented in [23] is optimized for band-sequential order and a specific implementation in band-interleaved order only can be found in [24] for the Xilinx Zynq FPGA.

In this paper, two synthesizable reusable Intellectual Property (IP) cores for the data compression and hyperspectral and multispectral image compression, described in VHDL, are presented. The IPs are fully compliant with the CCSDS 121.0-B-2 standard and the CCSDS 123.0-B-1 standard. The IP cores are designed to achieve low-complexity and high throughput in space-qualified FPGA and ASIC technologies. The VHDL description of the IP cores is technology independent and, therefore, it may be mapped to a diversity of target devices. Besides, in contrast to existing implementations, they offer the possibility for users to select among all the configuration options present in the standards, hence, the IPs are highly parametrizable and configurable. An architectural study based on SystemC modeling is carried out to ensure the user is provided with all possible parameters without compromising hardware complexity or maximum throughput. As shown later in Section IV-C, the complexity and throughput optimization implies designing different hardware architectures for some parameters. A description of the SystemC models is presented in Section IV as a summary of the more detailed contribution presented in [25].

The rest of the paper is structured as follows. Section II presents a brief description of the CCSDS algorithms. Section III outlines the new compression system. Section IV details the compression system model and

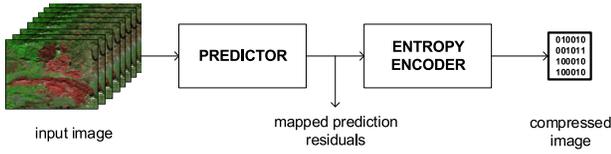


Fig. 1. CCSDS 123 standard: compressor schematic.

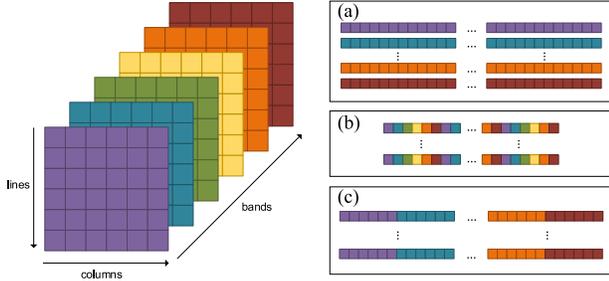


Fig. 2. Covered compression orders. (a) BSQ order (b) BIP order. (c) BIL order.

the outcomes of the model-based analysis. Section V discusses the developed IP core implementations; Section VI summarizes the results. Finally, Section VII presents the conclusions.

II. OVERVIEW OF THE CCSDS LOSSLESS COMPRESSION STANDARDS

The CCSDS 123.0-B-1 standard defines a payload lossless data compressor that has applicability to multispectral and hyperspectral imagers and sounders. Compression consists of two functional parts: a decorrelation followed by an entropy coding, as depicted in Fig. 1. The former is based on predictions, i.e., a linear adaptive prediction method is used for the estimation of each sample from neighboring samples. The latter is based on either a sample-adaptive encoder or a block-adaptive encoder. The block-adaptive option corresponds to the encoder described by the CCSDS 121.0-B-2 standard.

The CCSDS 123.0-B-1 standard also specifies the allowed formats for both input and output images. Input to the compressor must be a 3-D image with its samples arranged in band sequence (BSQ) or band interleaved order. The IP cores implemented in the scope of this paper consider three possible sample orders BSQ, band interleaved by pixel (BIP), and band interleaved by line (BIL), as it is illustrated in Fig. 2. It is assumed that the samples are compressed in the same order as they are stored or captured by the sensor, in order to allow for online compression. In BSQ the compression of all the samples in a band is computed before processing the next bands; in BIP a sample is compressed for all the bands before processing next samples; finally, in BIL each line of samples is compressed for all the bands before processing the next lines. The output from the compressor is an encoded bitstream. The content of this bitstream depends on the configuration parameters of the compression, and it is variable length. The structure of the compressed bitstream is a header containing the

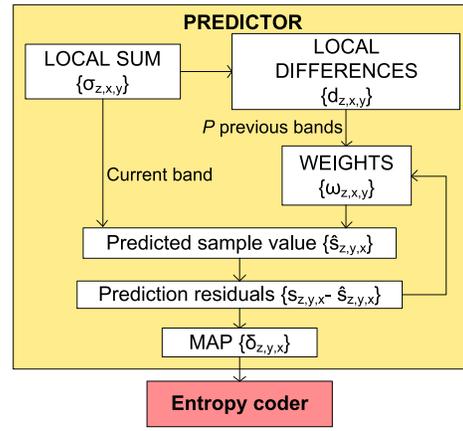


Fig. 3. CCSDS123 IP core: prediction flow.

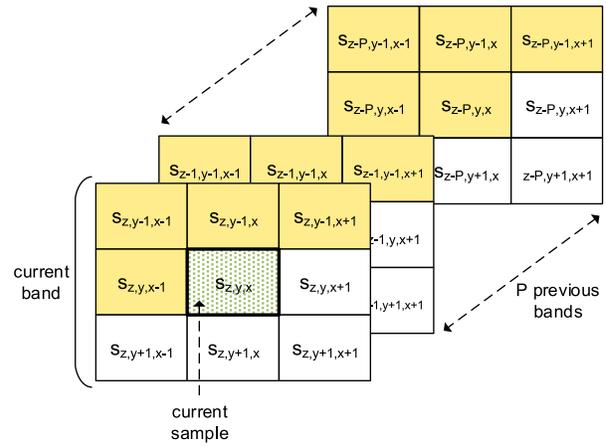


Fig. 4. CCSDS123 IP core: prediction neighborhood.

image metadata and compression parameters, followed by a body consisting of encoded output words.

The prediction of each sample is based on a specific spatial neighborhood in P previous spectral bands. The predicted values $\hat{s}_{z,y,x}$ are computed first, and then the prediction residuals are calculated from the input samples $s_{z,y,x}$ and mapped to integer values $\delta_{z,y,x}$. The flow for this process is shown in Fig. 3 and explained below. The possible extension of the neighborhood is shown in Fig. 4. The user may limit or extend this expanse by means of neighbor-oriented or column-oriented mode selection, and P value selection.

First, a local sum $\sigma_{z,y,x}$ is computed from neighboring samples in the current band. A local sum is a weighted addition of neighboring samples, whose extension is shown in Fig. 5. Under a neighbor-orientation, the local sum is obtained adding the previously processed adjacent samples. On the other hand, under a column orientation only the upper sample is considered. Each local sum is used to calculate a central local differences value $d_{z,y,x}$ and the directional local differences $d_{z,y,x}^N$, $d_{z,y,x}^W$, and $d_{z,y,x}^{NW}$. The user decides, which local differences to use, selecting a reduced or full mode, i.e., using only central local differences in the current and P previous bands or using also the directional local difference in the current band, respectively. At this

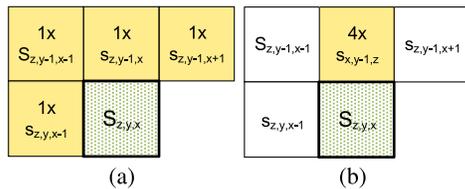


Fig. 5. CCSDS123 IP core: local sum orientation. (a) Neighbor oriented. (b) Column oriented.

point, a predicted sample $\hat{s}_{z,y,x}$ can be obtained through a relation between the local sum of the current band and a weighted sum of local differences of the current band and P previous bands. These weighting factors are updated as predictions are computed based on the prediction error. Finally, prediction residuals are obtained, as the difference between the predicted values and the real value of each sample $\{s_{z,y,x} - \hat{s}_{z,y,x}\}$. These residuals are mapped to positive integer values $\delta_{z,y,x}$ and are the inputs of the encoding process.

Residuals are sequentially encoded in the selected order: BSQ, BIP, or BIL. This encoding order specifies likewise the order in which the encoded samples are arranged in the compressed file in the scope of this paper. The encoding shall be performed using either a sample-adaptive entropy coding or a block-adaptive entropy coding approach.

With the sample-adaptive encoder, each mapped prediction residual is encoded with a Golomb power-of-two variable-length binary codeword. This encoder is based on two different statistics: an accumulator $\sum z(t)$ of mapped residuals and a counter $\gamma(t)$ for each spectral band. These statistics are adaptively updated during the encoding process and periodically reset according to an interval defined by the user. Values k and U are computed from the accumulator $\sum z(t)$ and the counter $\gamma(t)$. They are used to create the Golomb power-of-two codewords, so each nonnegative integer δ can be represented as a $U2^k + r$ bit word.

The CCSDS 123.0-B-1 standard provides the definition of the sample-adaptive encoder (defined in the previous paragraph) or optionally gives the possibility to use the CCSDS 121.0-B-2 as the encoder stage. The CCSDS 121.0-B-2 standard describes a block-adaptive entropy encoder, which utilizes a Rice's adaptive encoding technique. Under this technique, several algorithms are concurrently applied to each block of J consecutive preprocessed samples, as shown in Fig. 6. The encoding option that minimizes the number of bits in a coded block of J samples is selected to encode such block. Each encoding option has a unique identifier that must be placed before resulting encoded stream, in order to enable the decompression.

III. LOSSLESS SATELLITE COMPRESSION SYSTEM

A. Background

Previously to this paper, a basic CCSDS 123.0-B-1 implementation with a limited functionality had been developed [23]. The IP core is described in VHDL and its name is hyperspectral lossless compressor for space applications

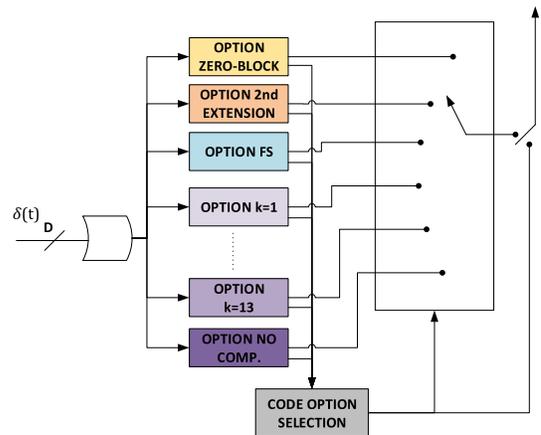


Fig. 6. CCSDS121 IP core: block-adaptive entropy encoder.

(HyLoC). The hyperspectral lossless compressor for space applications (HyLoC) IP core is a low complexity implementation able to fit in eligible FPGAs for space applications. HyLoC implements the CCSDS 123.0-B-1 standard in BSQ order and with the sample-adaptive encoding option only. The features of HyLoC and the main findings made during its development are summarized herein, serving as basis to establish the improvements to achieve with this new implementation.

The HyLoC implementation fits in a small space-qualified FPGA. It performs the compression in the BSQ order only, and serializes the computation of the local differences and weight vectors, reducing complexity as much as possible. HyLoC is fully compliant with the standard and allows for tuning the CCSDS 123.0-B-1 parameters at compile time. The VHDL description of HyLoC was validated against the reference software implementation from the European Space Agency (ESA) [26], and the Empordà software [27], developed by Universitat Autònoma de Barcelona (UAB); and synthesized for a space qualified FPGAs, the RTAX1000S, demonstrating that the goal of low complexity was achieved, with a hardware occupancy of 34% for a typical compressor configuration and 44% for the combination of parameters, which yields the highest complexity. In summary, HyLoC offers a low-complexity implementation that fits in a small FPGAs eligible for space applications. Nevertheless, despite being compliant with the standard, many options are not included and we consider there is room for improvement. In particular, in space missions a higher throughput could be desired. HyLoC exhibits a rather low throughput of 70 Mb/s on the RTAX1000S and 11.3 MSamples per second on a Virtex5 FX130. Furthermore, it accepts samples in BSQ order only, which can be a strong limitation for some applications that would require reordering the samples before compressing them, preventing it from achieving real-time performance. HyLoC allows for setting the different parameters only at compile time. This implies synthesizing and placing a new configuration on the FPGAs whenever the configuration needs to be changed. Run-time configuration capabilities would make the IP core much more flexible.

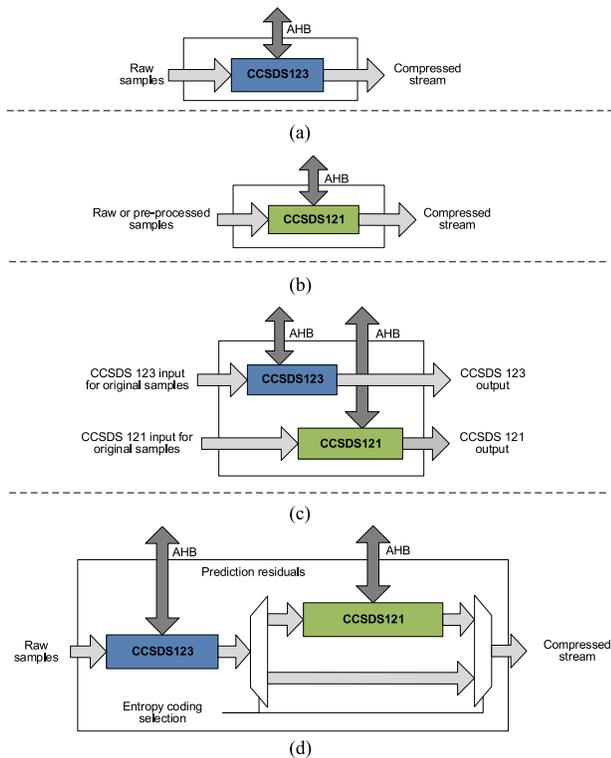


Fig. 7. CCSDS121 and CCSDS123 configurations. (a) Stand-alone CCSDS123 IP core. (b) Stand-alone CCSDS121 IP core. (c) CCSDS123 and CCSDS121 working unconnected. (d) CCSDS123 able to choose entropy encoder.

All these aspects have been considered to design the new compression system presented in this paper.

B. SHyLoC Concept

The hyperspectral lossless compressor for space applications (SHyLoC) system is formed by two new IP cores compliant with the CCSDS 121.0-B-2 and CCSDS 123.0-B-1 standards. They are part of the ESA IP core's library [28]. The compression system meets the requirements established by the ESA in the scope of the commissioned Technology Research Project (TRP AO8032) [29] entitled: "CCSDS Lossless Compression IP-Core Space applications."

The designed IP cores may be used in different configurations, as illustrated in Fig. 7, working independently as well as jointly. Setting the values of the compression parameters and selecting of the entropy coder option (sample-adaptive or CCSDS121 IP) shall be possible at run time if desired, i.e., without the need of turning OFF the device or loading a new bitstream to the FPGA. Notice in Fig. 7(d), the special case where both cores are instantiated, but the sample-adaptive entropy encoding is an available option for the compression as well; under this configuration, the user is able to select by means of the run-time configuration either the sample-adaptive option or the block adaptive, which is accomplished by the CCSDS121 IP core. The system shall be able to work in any of the possible sample arrangement orders: BSQ, BIP, or BIL. Regarding the throughput, both

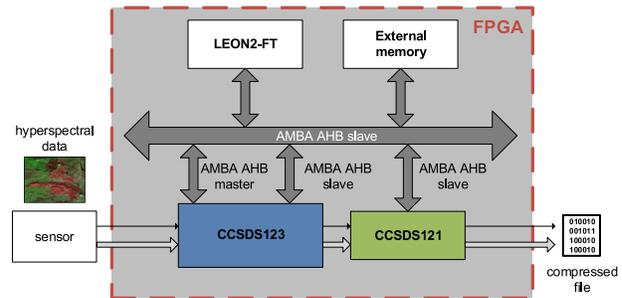


Fig. 8. Hyperspectral Lossless Compressor for space applications (SHyLoC) instantiation in an embedded system.

IP cores shall be able to compress images transmitted up to a maximum rate of 1 Gb/s, considering input samples 16 bits wide. The Xilinx FPGA Virtex 5 XQR5VFX130 is considered as reference technology on which to measure this performance requirement. Results obtained and shown in this document confirm both IP cores are in compliance with these requirements.

The designed IP cores can act as a coprocessor that is part of an embedded system on-board a satellite. They are designed to be compatible with GRLIB [30] and the LEON2-FT processor, a fault tolerant 32-bit processor conforming to the IEEE-1754 (SPARC V8) [31], widely used in space. An example of a possible instantiation of the compression system is shown in Fig. 8. In this hypothetical scenario, the LEON2-FT processor sets the configuration for both cores through an advanced microcontroller bus architecture-advanced high-performance bus (AMBA AHB). Since the IP cores are able to work in any compression order and with different dynamic ranges, the input samples can be provided directly from the ADC of the remote sensor. Additionally, the inclusion of an external memory is considered to store intermediate values during the compression when necessary.

IV. SHYLOC SYSTEMC MODEL

As a first design step of the SHyLoC development, both IP cores have been modeled in the electronic system level language SystemC, making it possible to iterate through different hardware architecture possibilities in a relatively short time. This fact allows to generate specifications for the subsequent VHDL description stages that will lead to efficient implementations.

The CCSDS 121.0-B-2 and CCSDS 123.0-B-1 standards allow for the selection of a fair amount of configuration parameters. Each of these parameters has its particular influence on the compression performance, features, and complexity of the resulting compression system. The different key features can be measured at the modeling stage, e.g., throughput figures or the amount of memory elements can be easily estimated. Several explorations have been carried out considering different parameters of the algorithms. Decisions related to design optimizations have been taken at this stage. Therefore, one of the main goals of the modeling stage has been to determine the parameters to optimize.

Beyond that, the models make it also possible to identify common operations between the CCSDS 121.0-B-2 and the CCSDS 123.0-B-1 algorithms. This has made it possible to design as many reusable VHDL components as possible. As an addition, having the SHyLoC system in a SystemC model meeting the specifications and requirements of the aforementioned TRP activity, ensures that the subsequent VHDL description of the IP cores meets those specifications and requirements as well.

The SHyLoC SystemC models can also be included in the SoCRocket platform [32]. This virtual platform aims at easing design-space exploration at various levels of abstractions and development stages. To that extent, SoCRocket has a set of SystemC modeling library components providing tools, mechanisms, and interconnection infrastructures to model and simulate embedded systems. These library components are typical of embedded systems used for space applications, like the LEON processor. For these reasons, developing SHyLoC models compatible with the SoCRocket platform entails a clear advantage: the behavior of the model can be analyzed within a complete system, cooperating with others modules in a unified simulation environment.

The models have been developed to keep concordance with the intended IP cores, i.e., interfaces and configuration modes are basically the same for both implementations. The throughput requirements and the ability to work in any of the compression orders: BSQ, BIP, or BIL, are present in the models as well. These models are detailed in Section IV-A and IV-B. Both models present ad hoc data and control interfaces, as well as AMBA AHB interfaces. The interfaces have been designed to make the CCSDS121 and the CCSDS123 models fully compatible. Configuration modes are essentially the same than the IP cores: compile-time and run-time configuration is allowed, with mechanisms that will be explained in detail in Section V.

A deep study of the algorithms allows throwing some light regarding data dependencies and possible throughput limitations. On the side of the CCSDS121 no important data dependencies have been found, therefore, no throughput limitations are expected to arise when implemented. However, the CCSDS123 model deserves more attention, because of its high complexity. It stood out the influence of the compression order in the impact of data dependencies in the achievable throughput. Since the models must respect and perform compression in every possible order, the moment in which data dependencies are found differ between orders and this fact has an obvious influence in the throughput. Therefore, with the aim of performing as many parallel operations as possible, and minding the memory occupancy, different schedules have been designed for the compression. Studying the algorithms theoretically, we were also able to observe the impact on complexity of the P parameter. It has been considered not necessary to optimize the designed hardware architecture for any possible P value. Conversely, it is more reasonable to offer the user the possibility to choose, with a compile-time configuration

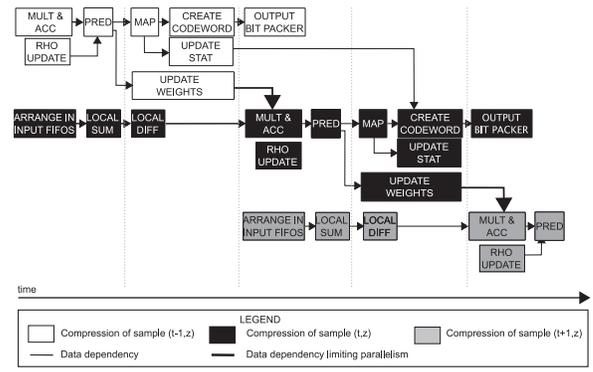


Fig. 9. BSQ and BIL schedule for samples $s(t - 1, z)$, $s(t, z)$, $s(t + 1, z)$.

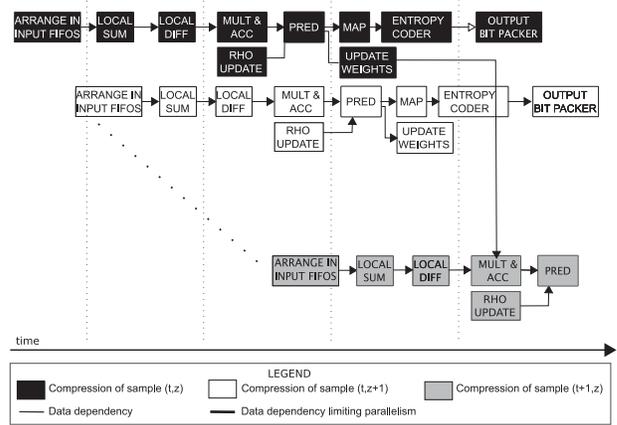


Fig. 10. BIP schedule for samples $s(t, z)$, $s(t, z + 1) \dots s(t + 1, z)$.

parameter, the maximum selectable P value, to reduce the complexity of the implementation.

The schedule for the compression of a sample in the different compression orders is proposed at this modeling stage. The BSQ and BIL schedule is illustrated in Fig. 9. Since the weight vector must be updated before the prediction of a new sample, which shall be computed by the MULT and ACC module, we can observe the implied limitation in the number of operations that can be scheduled in parallel.

The schedule of Fig. 10 corresponds to the BIP arrangement. Notice in this case that, although there are the same data dependencies, a sample $s(t, z)$ is processed in all the bands before the next sample in the same line $s(t + 1, z)$ is processed. Hence, the data dependency does not prevent the compression of a new sample $s(t, z + 1)$ to start before the prediction of the current sample $s(t, z)$ has finished, because the corresponding weight vector will be already updated once we reach $s(t + 1, z)$. This opens the possibility of starting the compression of a new sample in every clock cycle, by designing an appropriate pipeline.

With the SystemC models described previously, the design space exploration has been performed. An iterative refinement of the architecture has been possible, retrieving information about performance limitations for both IP cores, storage requirements, data dependencies, and prospective

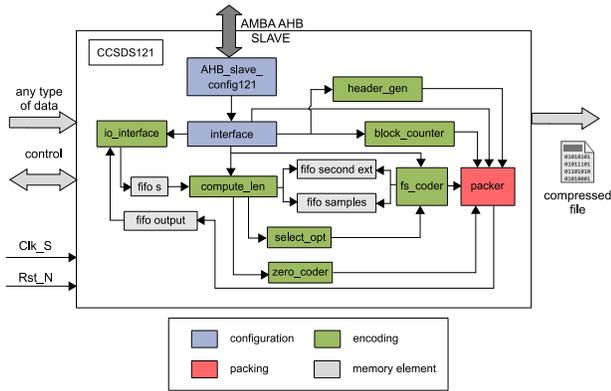


Fig. 11. CCSDS121 SystemC model.

hardware requirements. Taking into account these results, we conclude that the real-time compression target cannot be achieved with a common hardware architecture that allows run-time selection of the compression order. The existing data dependencies result in a throughput limitation for BSQ and BIL cases, as highlighted before. Different architectures for each compression order, BSQ, BIP, and BIL, have been finally designed. Eventually, the architecture to be implemented is specified by the user at compile time.

A. CCSDS121 SystemC Model

We present next the main design considerations and components of the CCSDS121 model. The components are displayed in Fig. 11. The *AHB_slave_config121* and *interface* implement the necessary steps to set the compression configuration parameters. The former receives the possible run-time configuration, through different transaction level modeling (TLM) functions that model the Advanced Microcontroller Bus Architecture - Advanced High-performance Bus (AMBA AHB) bus. Transaction Level Modeling (TLM) provides the means to separate the details of communication among modules from the details of the implementation of the communication architectures. Communication mechanisms such as buses or FIFOs are modeled as channels, and are presented to other modules using SystemC interface classes. The final configuration is validated by the *interface* module, and then propagated to the rest of the modules. The *io_interface* block manages the configuration, data and control for the compression process, which is performed by the rest of green modules in Fig. 11. *header_gen* creates the header; *compute_len* and *select_opt* calculates the length of each encoding option and select the option to encode the current block of samples at the time, respectively. *fs_coder* is the module that performs the encoding of the block of samples with the selected option, whereas *zero_coder* keeps track of the zero-blocks and encodes them. The *packer* fits the encoded output words to the output buffer size, and *block_counter* is a simple counter of received blocks. A set of FIFO channels are used to store different values: *fifo s*, to store received samples; *fifo samples*, to store the samples while lengths for the encoded block are being calculated; *fifo second ext* for the second

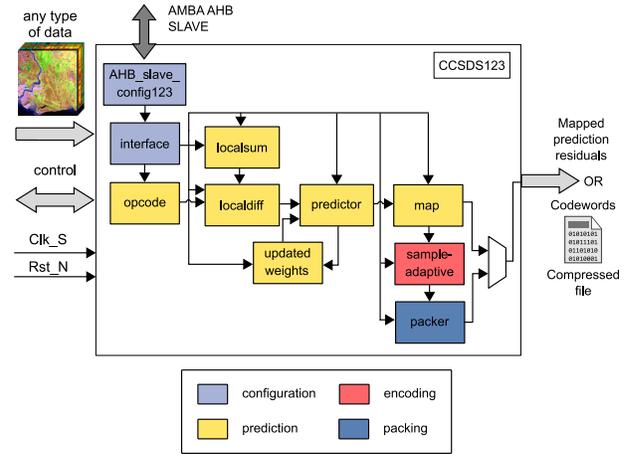


Fig. 12. CCSDS123 SystemC model.

extension values; and finally, *fifo output* to store the compressed bitstream.

B. CCSDS123 SystemC Model

As shown in Fig. 12, the dedicated *AHB_slave_config123* and *interface* modules also conform part of the configuration for the CCSDS123 SystemC model. The prediction process is performed by the following modules: *opcode*, which manages appropriately input samples in the different FIFOs; *localsum* and *localdiff*, which calculate the necessary values to compute the predicted samples by *predictor*; while *updated weights* updates the weight vectors for later predictions, *map* module maps the current prediction residual. The *sample-adaptive* module encodes the mapped prediction residuals. The *packer* is present again for the latter part of the process. Since the amount of FIFOs for this case is higher and it is only intended to outline the models, they have been omitted for simplicity of the diagram.

C. SystemC Results and Conclusions

The models have been verified with reference images obtained with software implementations from the ESA and UAB, as it was done for the development of the HyLoC IP core [23]. Once the appropriate behavior of the models was verified, a set of experiments were carried out in order to benchmark the performance.

Regarding the CCSDS121 model, the only parameter impacting the performance is the J parameter. The maximum possible block size, J affects the size of the FIFO, as shown in Fig. 13. Additionally, it has an impact in the maximum latency, as shown in Fig. 14. Here, we define the maximum latency as the maximum time elapsed between the beginning of the processing of a valid input and the generation of a valid output during the compression of an image. The throughput of the CCSDS121 model reaches 100 Msamples per second, after the configuration and the initial latency (number of cycles elapsed from the moment the first sample is processed until the first output is produced), assuming a clock period of 10 ns.

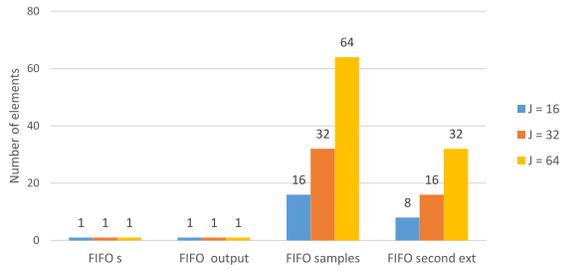


Fig. 13. CCSDS121 SystemC model: Number of stored elements.

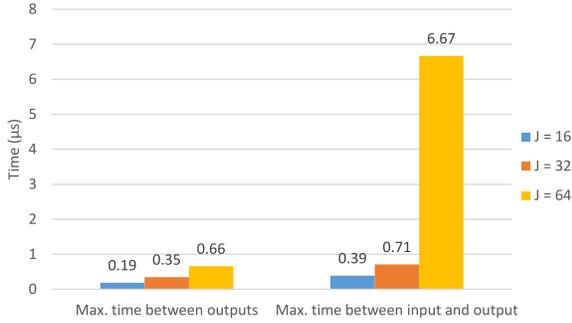


Fig. 14. CCSDS121 SystemC model: Timing measures.

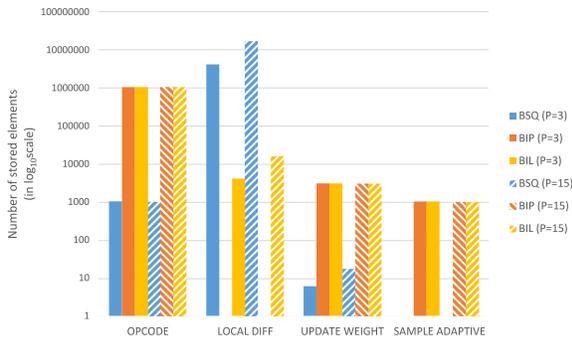


Fig. 15. CCSDS123 SystemC model: Number of stored elements.

On the other hand, some early assumptions have been confirmed when benchmarking the CCSDS123 model regarding the compile-time configuration parameters, the number of bands used for prediction, P , and the arrangement of the input samples. We have observed that the influence of the P parameter extends to memory size and complexity. Besides the obvious memory size dependency on the image size, P affects significantly for BSQ and BIL compression orders, as shown in Fig. 15. Since the amount of used memory might vary significantly with all the outlined factors, Table I provides formulas to illustrate those dependence relationships. Also the number of multipliers and multiplications on the prediction stage depends directly on P , as summarized in Table II. However, it can also be noticed that compression order does not affect the number of multipliers and multiplications.

The simulations executed with the CCSDS123 model reveal a maximum throughput of 100 MSamples per second, after the configuration and the initial latency, for BIP order. This is possible because each component in the model has a one clock cycle latency, and the processing of a sample

TABLE I
Memory Elements Definition per Module

	BSQ	BIP	BIL
OP CODE			
Size	$N_x \cdot \text{GEN} + 4$	$N_z \cdot \text{GEN} \cdot (N_x \cdot \text{GEN} + 1) + 3$	$N_z \cdot \text{GEN} \cdot (N_x \cdot \text{GEN} + 1) + 3$
Elements	$N_x + 4$	$N_z \cdot (N_x + 1) + 3$	$N_z \cdot (N_x + 1) + 3$
LOCAL DIFF			
Size	$(P_{\text{MAX}} + 1) \cdot (N_x \cdot \text{GEN} \cdot N_y \cdot \text{GEN})$	-	$(P_{\text{MAX}} + 1) \cdot N_x \cdot \text{GEN}$
Elements	$(P + 1) \cdot N_x \cdot N_y$	-	$(P + 1) \cdot N_x$
UPDATE WEIGHT			
Size	$C_z \cdot \text{GEN}$	$C_z \cdot \text{GEN} \cdot N_z \cdot \text{GEN}$	$C_z \cdot \text{GEN} \cdot N_z \cdot \text{GEN}$
Elements	C_z	$C_z \cdot N_z$	$C_z \cdot N_z$
SAMPLE ADAPTIVE			
Size	1	$N_z \cdot \text{GEN}$	$N_z \cdot \text{GEN}$
Elements	1	N_z	N_z

TABLE II
Number of Multipliers and Multiplications

	P=3			P=15		
	BSQ	BIP	BIL	BSQ	BIP	BIL
Multipliers	6	6	6	18	18	18
Multiplications	32256	32256	32256	96768	96768	96768

can be pipelined. When compression is performed in BSQ or BIL order, it reaches 25 MSamples per second, i.e., accepting a sample every four clock cycles. These results do not consider the possible requirement of having an external memory, that might be accessed through a shared communication bus; if such a memory is present, the throughput for BSQ and BIL might be negatively affected.

V. SHYLOC VHDL DESCRIPTION

SHyLoC consists of two IP cores: the CCSDS121 and the CCSDS123, which are described in VHDL. The input and output interfaces and possible ways to connect for the CCSDS121 and CCSDS123 IP cores are depicted in Fig. 16. Both IP cores include data and control interfaces, and standard AMBA AHB interfaces. Ad hoc FIFO-like interfaces are used to receive the input samples to be compressed with high data-rate and output the compressed images. Standard AHB interfaces are used to receive the run-time configuration parameters and to access an external memory for the storage of intermediate values during the compression. Moreover, the ad hoc interfaces are designed for an easy plug and play of an external entropy encoder, so both IP cores can be easily connected to work jointly.

The interface signals are mostly common for both IP cores, and they can be grouped in: system interface, comprising the clock signals and the resets (which can be configured to be either synchronous or asynchronous); input data interface, where input data requires a validation signal;

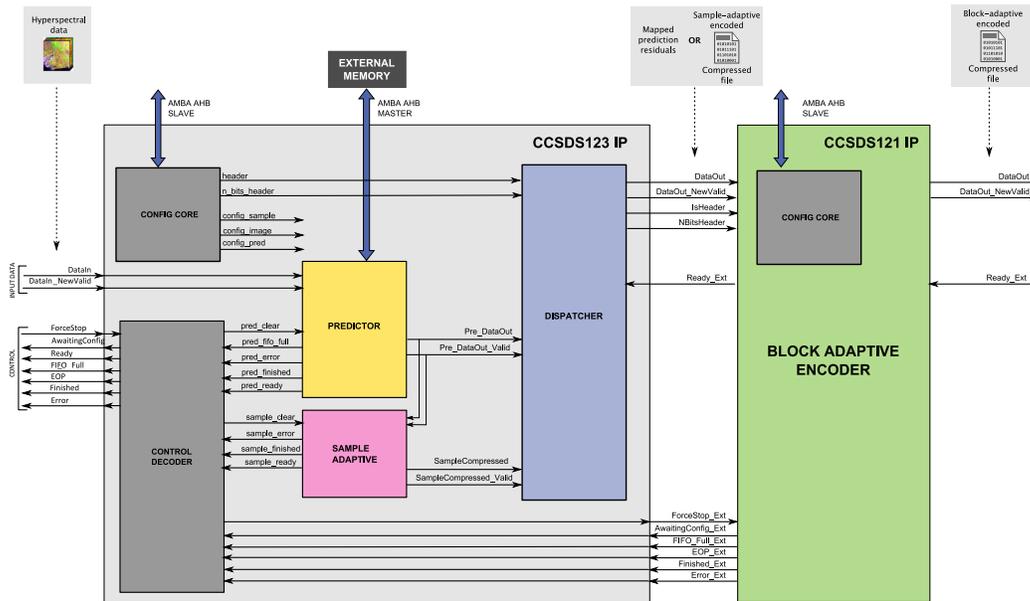


Fig. 16. CCSDS 123 and 121 IP Core interface overview and connection.

output data interface, whereby outputs are validated; control interface, which comprises availability and status information; and AMBA AHB slave interface, with its own clock domain and reset signal, and the input and output AMBA AHB slave interface signals. Additionally, the CCSDS123 includes an AMBA AHB master interface to communicate with an external memory to store intermediate values if needed; and an external encoder control interface, which is utilized when an external entropy encoder is attached to the CCSDS123 IP core, allowing communication between CCSDS123 as preprocessor and the external encoder.

The input samples may be received from different sources: a mass memory, a SpaceWire interface, an ADC or another IP. Similarly, the output data may be sent to a mass memory, a SpaceWire interface or to another IP core. The output data consists of the compressed bitstream defined in the CCSDS 121.0-B-2 and CCSDS 123.0-B-1 standards, including side information (headers), if necessary.

Fig. 16 shows how the output data interface of the CCSDS123 IP core is connected to the input data interface of the CCSDS121 IP core. The CCSDS123 IP includes an additional control interface, the external encoder interface, that is meant to be connected to the control interface of the CCSDS121 IP core. In order to appropriately transfer the header values from the CCSDS123 IP to the CCSDS121 IP, two specific signals are included in the output and input data interfaces of the CCSDS123 and CCSDS121 IPs, respectively. These signals mark transferred data as a header and specify the amount of valid bits.

Both IP cores present a high level of parametrization and configurations, as defined in the standards [13], [15]. The configurations can be selected either at run time (they are stored in configuration registers) or compile time (setting a number of constants before synthesis). These configurations result in a large variety of IP core versions with

different occupancy and performance and tree different architectures (BIP, BIL, and BSQ).

At compile time the design options are set by VHDL constants. These constants are stored in specific packages and propagated to the submodules or passed directly by constants. On both IP cores, *EN_RUNCFG* constant is used to enable or disable the configuration of the compressor parameters at run time. When run-time configuration is enabled, the constant parameters affect or limit the run-time configuration options. A detailed description of the constants are shown in Section V-A and V-B for CCSDS 121 and CCSDS 123, respectively.

For both configuration modes the IP cores will check that the selected configuration environment meets a variety of rules. These checks are preventive for consistency between the configuration values, as well as to ensure that the selected values are within the allowed values for each one of them. For the run-time configuration, some extra analysis are performed to ensure a proper configuration via the bus, e.g., that all necessary registers have been transferred.

In case a configuration error is detected by the IP cores, they will inform the user by the *Error* signal assertion, returning to an initial state in which the IPs expects to be configured again for a following compression.

A. CCSDS121 IP Core

The CCSDS121 IP core performs the encoding stage of the compression of digital data according to the CCSDS 121.0-B-2 standard. This encoding stage has been developed in a way that it can be easily attached to any preprocessor IP core that suits the interface that will be explained herein below, but being at the same time self-contained.

1) *Design Options*: Constant options of the CCSDS121 IP core are detailed in Table III. The constants are grouped based on the affected functionality

TABLE III
CCSDS121 IP Core: Constant Options

Constant	Description
SYSTEM CONSTANTS	
EN_RUNCFG	Enables or disables run-time configuration (the configuration is set at run-time reading the memory-mapped register or it is set at implementation time)
RESET_TYPE	Implements asynchronous or synchronous reset
PREPROCESSOR_GEN	The pre-processor is not present, the CCSDS121 IP is working as an independent compressor, generating its own header. The CCSDS123 IP pre-processor is present, and the CCSDS121 IP is working as entropy coder of the CCSDS123 IP. Any header sent from the CCSDS123 pre-processor is bypassed; and afterwards the CCSDS121 IP generates its own header. Any other pre-processor is present. The CCSDS121 IP is working as entropy coder of another pre-processor, different from the CCSDS123 IP. Any header sent from the pre-processor is bypassed.
IMAGE CONSTANTS	
Nx_GEN	Maximum allowed number of samples in a line
Ny_GEN	Maximum allowed number of samples in a row
Nz_GEN	Maximum allowed number of bands
D_GEN	Maximum dynamic range of the input samples
ENDIANESS_GEN	Little endian or big endian
ENCODING CONSTANTS	
DISABLE_HEADER_GEN	Selects whether to disable or not the header generation
J_GEN	Block size
CODESET_GEN	Code option
REF_SAMPLE_GEN	Reference sample interval
W_BUFFER_GEN	Number of bits in the output buffer
AHB CONSTANTS	
HSINDEX	AHB slave index
HSCONFIGADDR	ADDR field of the AHB Slave. Sets the 12 most significant bits in the 32-bit AHB address
HSADDRMASK	MASK field of the AHB Slave

as system constants, image constants, encoding constants, and AMBA AHB bus constants. The *PREPROCESSOR_GEN* constant in the system group specifies if a preprocessor is present or not, and the reset signal can be defined as synchronous or asynchronous. *PREPROCESSOR_GEN* also specifies if the CCSDS123 IP core is the present preprocessor. Image constants characterize the size of the image and the dynamic range and the endianness. The IP allows different block sizes and different reference sample intervals for encoding by means of *J_GEN* and *REF_SAMPLE_GEN* constants, respectively.

2) *Run-Time Configuration*: When *EN_RUNCFG* is asserted the IP core is configured through AMBA bus. A set of memory-mapped configuration registers is defined, allowing to select the encoding settings in run time. All options presented in Table III can be set by the user in run time, with the exception of the system constants.

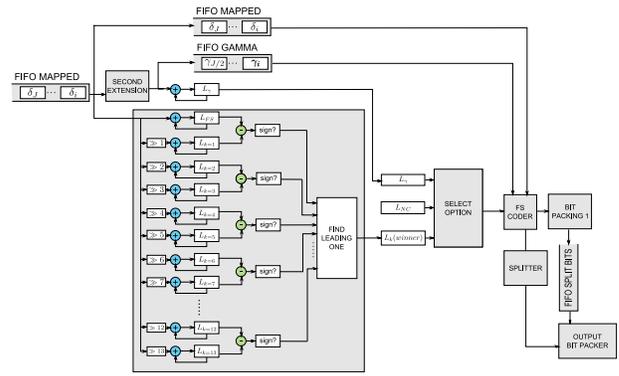


Fig. 17. CCSDS121 IP core compression schematic.

3) *Compression*: Once the configuration values have been checked by the IP core, the header is issued. Since the CCSDS121 IP core can be attached or not to a preprocessor module, the header may be issued in different ways, depending on the configuration. A header coming from a preprocessor may be bypassed and attached to the encoding header created by the CCSDS121 IP core. Furthermore, the header generation can be disabled for different purposes, by means of the *DISABLE_HEADER* parameter.

At this point, the CCSDS121 IP core is ready to receive new samples, and compress them as shown in the schematic of Fig. 17. The input raw samples or mapped residuals coming from a preprocessor are stored in a FIFO of size J , where J specifies the block size. As samples are stored in the FIFO, the different compression options and their lengths are calculated. Besides, the intermediate values needed for the second-extension option are computed and stored for the cases in which the second-extension option is the chosen one. Once the IP core calculates the amount of bits taken for each encoding option, the one that comprises the minimum amount of bits is the one selected. Thereafter samples are encoded according to the selected option including, if necessary, the noncompression possibility. Encoded samples are packed and sent through the output interface preceded by the identifier of the chosen encoding option. The IP core considers also a set of possibilities related to one or more blocks of samples being equal to zero.

B. CCSDS123 IP Core

The CCSDS123 IP core performs the compression of a set of input samples according to the specifications of the CCSDS 123.0-B-1 standard. The CCSDS123 includes the implementation of the sample-adaptive entropy encoder. However, the block-adaptive encoder can be included in the compression by piecing together the CCSDS123 and the CCSDS121 IP cores. In that case, the CCSDS123 IP core produces the corresponding header and the mapped prediction residuals that are then received by the CCSDS121 IP core. The latter encodes these residuals and produces the bitstream.

1) *Design Options*: Tables IV and V summarize the constant options of the CCSDS123 IP core. *EDAC*

TABLE IV
CCSDS123 IP Core: System and AMBA AHB Constant Options

Constant	Description
SYSTEM CONSTANTS	
EN_RUNCFG	Enables or disables run-time configuration. (Parameters set at run-time reading the memory-mapped configuration registers or set at implementation time reading the constants)
EDAC	Inhibits or implements EDAC for embedded memories
RESET_GEN	Implement synchronous or asynchronous reset.
ENCODING_TYPE	Indicates if sample-adaptive is implemented or not (if only pre-processor is implemented, an external encoder IP core can be attached)
PREDICTION_TYPE	Architecture implemented for predictor (BIP, BIP-mem, BSQ, or BIL)
ExtMemAddress_GEN	External memory address. Sets the 12 most significant bits in the 32-bit AHB address.
AMBA AHB CONSTANTS	
HSINDEX	AHB slave index.
HSCONFIG_ADDR	ADDR field of the AHB Slave. Sets the 12 most significant bits in the 32-bit AHB address.
HSADDR_MASK	MASK field of the AHB Slave
HMINDEX	AHB master index
HMAX_BURST	AHB master burst beat limit

constant determines if error detection and correction (EDAC) is implemented or not for the embedded memories in the CCSDS123 IP core. The combination of the *ENCODING_TYPE* and the *ENCODER_SELECTION_GEN* shall determine, which encoder is implemented (either or both sample adaptive and block adaptive) and the actual encoder used for the process. *PREDICTION_TYPE* shall be set according to the order in which the samples are arranged in the raw multispectral/hyperspectral image. *ExtMemAddress_GEN* specifies the external memory address.

The compressor considers not only the dynamic range of the input samples, *D_GEN*, and the endianness, *ENDIANESS_GEN*, but also the sign, *IS_SIGNED*. The prediction parameters includes the possibility to limit the number of bands used for the prediction of a sample, *P_MAX*; additionally it provides configuration constants for the local differences and local sums, *PREDICTION_GEN* and *LOCAL_SUM_GEN*; and several weight-related constants. The sample-adaptive presents several constants related to the accumulator and the counter used for the encoding.

2) *Run-Time Configuration*: Similarly to the CCSDS121 IP core, a set of memory-mapped configuration registers is defined for run-time configuration. All options presented in Tables IV and V can be set by the user, with the exception of the system constants. For the CCSDS123 IP core more registers are needed to store the configuration, since the number of settings considering image, prediction, and sample-adaptive encoding configuration is fairly higher.

This IP core holds a configuration engine, a control decoder, a predictor module, a sample-adaptive encoder,

TABLE V
CCSDS123 IP Core: Image and Sample-Adaptive Encoding Constant Options

Constant	Description
IMAGE CONSTANTS	
Nx_GEN	Maximum allowed number of samples in a line
Ny_GEN	Maximum allowed number of samples in a row
Nz_GEN	Maximum allowed number of bands
D_GEN	Maximum dynamic range of the input samples
IS_SIGNED_GEN	Signed or unsigned input samples
ENDIANESS_GEN	Little endian or big endian
DISABLE_HEADER_GEN	Generates header or inhibits header generation
W_BUFFER_GEN	Bit width of the output buffer
PREDICTOR CONSTANTS	
P_MAX	Number of bands used for prediction
PREDICTION_GEN	Full or reduced prediction
LOCAL_SUM_GEN	Neighbour or column oriented local sum
OMEGA_GEN	Weight component resolution
R_GEN	Register size
VMAX_GEN	Factor for weight update
VMIN_GEN	Factor for weight update
T_INC_GEN	Weight update factor change interval
WEIGHT_INIT_GEN	Weight initialization mode
Q_GEN	Custom weight resolution
SAMPLE-ADAPTIVE ENCODER CONSTANTS	
ENCODER_SELECTION_GEN	Disables encoding, or selects either sample-adaptive or block-adaptive
INIT_COUNT_E_GEN	Initial count exponent.
ACC_INIT_TYPE_GEN	Accumulator initialization type
ACC_INIT_CONST_GEN	Accumulator initialization constant
RESC_COUNT_SIZE_GEN	Rescaling counter size
U_MAX_GEN	Unary length limit
ACC_GEN_TAB1-TABn	Accumulator initialization table values to be used when <i>ACC_INIT_TYPE_GEN</i> = 1; not used otherwise

and a dispatcher in order to receive the configuration and the input samples, compress them and send them to the output, as illustrated in Fig. 16.

Configuration module, *config_core*, includes several submodules to perform the configuration of the whole compression set, including clock adaptation. Compression itself can be performed by *predictor* and *sample adaptive* when selected by the user by means of

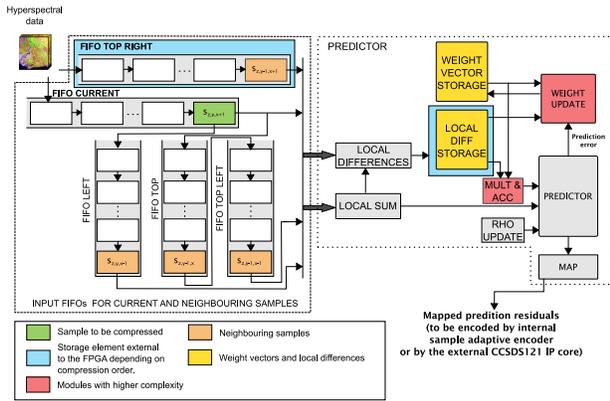


Fig. 18. CCSDS123 IP core: prediction schematic.

configuration. Input samples are transferred to the predictor, where mapped residuals are calculated. Predictor might have access to an external memory to store intermediate values if the chosen architecture includes it. The *control decoder* ensures synchronization of the different control and data signals. Finally, the *dispatcher* receives the output from the different compression modules, organizing them in different FIFOs, and addressing the appropriate output according to the compression configuration.

3) *Prediction Architecture*: The complexity of the prediction stage is rather high. Three different architectures, one for each allowed compression order, are required: BIP, BIL, and BSQ. This idea was also supported by the results of the SystemC modeling. The configuration, however, is independent from the compression order and, hence, a common structure has been designed to read the configuration registers and validate the received values. The compression engine consists of two main modules: a component module together and a finite states machine module. The former instantiates all the necessary components to perform prediction, and the latter contains several finite state machines, which schedule the prediction process according to the specific compression order, and so to the arrangement of the input samples, through a series of component activations when appropriate. An outline of the prediction core is represented in Fig. 18, which shows a high-level representation of the components that are common to all the predictor architectures.

The input samples are arranged by the IP core in a set of FIFOs, according the compression order. As samples are compressed, they are rearranged in a way that the already processed samples become the neighboring samples for subsequent samples. This is how the IP core organizes samples to cover the explained possible neighborhood extensions in Section II. The size of these FIFOs are summarized on Table VI, along with the most remarkable memory elements: the local differences and the weight vectors. The amount of memory needed depends on the shown compile-time parameters. Besides the image size, there is a great dependence in terms of storage with the P parameter. On the BIP architecture the FIFO that stores the top right neighbors, *FIFO_TOP_RIGHT*, can reach large sizes

TABLE VI
CCSDS123 IP Core: FIFO Size for Intermediate Values

Order	Neighbouring samples	Local differences	Weights
BSQ	$N_x + 1$	$P \times N_x \times N_y$	C_z
BIP	$(N_x + 1) \times N_z$	P	$N_z \times C_z$
BIL	$(N_x + 1) \times N_z$	$N_x \times P$	$N_z \times C_z$

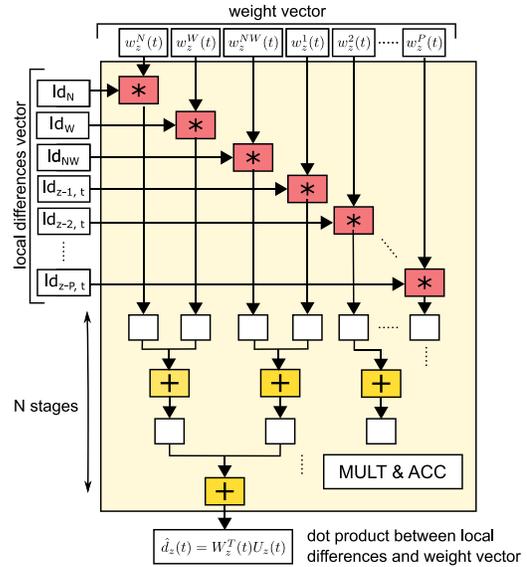


Fig. 19. CCSDS123 IP core: Dot product realization in BIP and BIL.

and the possibility of allocating such FIFO in a memory external to the FPGAs has been considered. As a result, a fourth architecture, named band interleaved by pixel memory (BIP-MEM), has been designed. This architecture can be used on FPGAs such as Microsemi RTAX or NanoXplore NX-eFPGA.

Prediction process imposes certain data dependencies. They are the same for every compression order, but each order makes those data dependencies more or less restrictive.

a) *BIP*: The way the samples are issued in the BIP architecture, ensures processing spatially previous samples in all bands before processing the current sample. In particular, provided enough samples in the spectral dimension, BIP architecture is able to accept one compressed sample per clock cycle. This feature makes this prediction architecture capable of providing the highest possible throughput.

The multiply and accumulate (MAC) units are used to obtain the predicted samples. The number of multiplier instances depends on the compile-time parameter P_MAX . Fig. 19 reflects the organization related to these operators. It shows how dot products of the local differences and weight vectors are performed. Since the weight vectors are updated every clock cycle, this structure is able to provide one dot product per clock size. The weight vector update module is shown in Fig. 20. Parallel *weight update* units are used this purpose. The number of weight vectors depends also on the compile-time parameter P_MAX , and so does the number of weight update units.

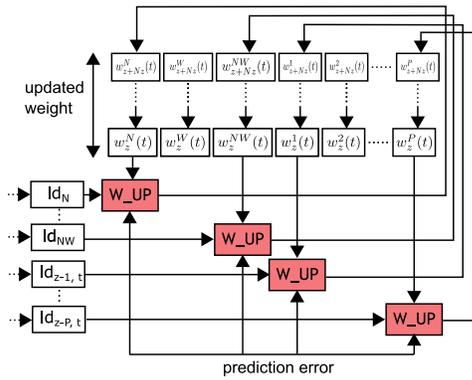


Fig. 20. CCSDS123 IP core: Weight vector update in BIP and BIL.

As an alternative to the BIP architecture, BIP-MEM architecture offers the user the possibility of using an external memory to store the mentioned *FIFO_TOP_RIGHT*. The access to this memory is performed by the AMBA AHB master interface present in the IP core. One read and one write operations are needed per sample compression.

b) *BSQ*: Since the size of the local differences memory elements needed for BSQ is quite high as shown in Table VI, this particular architecture stores this local differences vector in an external memory, unlike the other architectures. Provided that the local differences vectors need to be retrieved from the external memory, we observed that there is no clear advantage in having the multiply and accumulate operations and the weight vector update performed with the structures depicted in Figs. 19 and 20. Conversely, this operation and the weight vector update are serialized in order to reduce the resource utilization. The rest of the predictor uses the same components as the BIP architecture.

c) *BIL*: Inheriting most of the components from the BIP architecture, in the BIL order case, the data dependencies from the two previous architectures are present. When compressing samples in a line, BSQ-type data dependencies are present. When compressing the last sample of a line and first sample of the next line, BIP-type data dependencies occur. To ensure the best performance in this case, a specific scheduling has been designed considering the two previous situations. Such scheduling is shown in Fig. 21, where different FIFOs are used for a specific storage of the *local difference* vector.

4) *Encoding Architecture*: The encoder receives mapped residuals from the predictor and stores them in an input FIFO to then perform the entropy encoding. Fig. 22 shows a basic block diagram of the sample-adaptive encoder, which is based on a *components* module ruled by a *fsm* module. The *components* module is formed by two specific-architecture modules followed by a create codeword generator and a bit packer. As defined for the prediction stage, different compression orders lead to different architectures for the *opcode_update* module, which generates operation codes based on the sample position within the image for the sample to encode; and *update_counters* module, where the computation of the counters and accumulator values is performed. Fig. 22 reflects two FIFOs in

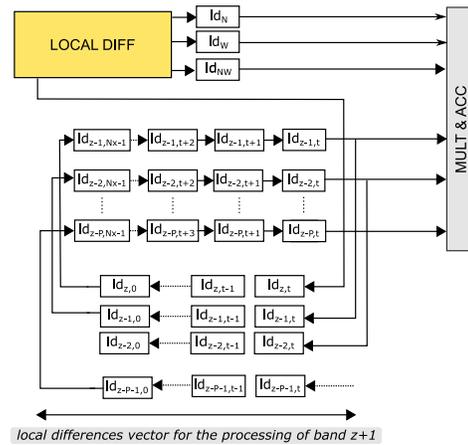


Fig. 21. CCSDS123 IP core: Handling of *local differences* vector in BIL order.

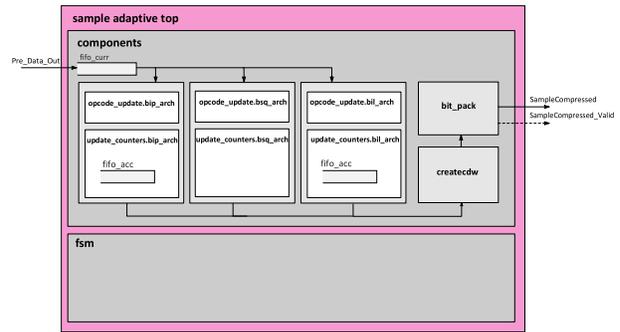


Fig. 22. CCSDS123 IP core: Sample-adaptive encoder block diagram.

BIP and BIL architectures to store the accumulator values of a sample in all bands, necessary for the compression of the next sample. The final part is the *createcdw* module and the *bit_pack* module. The former performs the encoding itself from the input mapped residuals and the computed counter and accumulator values. The latter packs the generated codewords in a proper bit-width output buffer. A validation flag is generated every time the output buffer is full, so output values can be captured.

VI. EXPERIMENTAL RESULTS

The VHDL models of the CCSDS121 and CCSDS123 IP cores are compliant with the CCSDS 121.0-B-2 and CCSDS 123.0-B-1 standards, respectively, both in terms of functionality and configurability. The IP cores are technology independent, being able to target any technology interesting for the space industry for the implementation.

In order to verify the correct behavior of the cores, more than 200 simulations tests have been carried out, using 47 different patterns and images as stimuli. The resulting compressed images were compared with golden reference results for each simulation. The reference software implementation from the ESA and Empordá software have been used again to obtain those golden reference results.

The synthesis results of both IP cores are presented in the following sections. These results have been obtained with Synopsys Synplify Premier 2016.09-1 [33].

TABLE VII
CCSDS121 IP Core: Synthesis Configuration Parameters

	Set 1	Set 2	Set 3	Set 4
GENERIC PARAMETERS				
EN_RUNCFG	0	0	0	1
RESET_TYPE	1			
PREPROCESSOR_GEN	0	0	0	0
Nx_GEN	1024			
Ny_GEN	1024			
Nz_GEN	1024			
D_GEN	16			
ENDIANESS_GEN	le			
J_GEN	16	32	64	16
CODESET_GEN	0			
REF_SAMPLE_GEN	256			
W_BUFFER_GEN	32			
DISABLE_HEADER_GEN	0			

TABLE VIII
CCSDS121 IP Core: Synthesis Results on Virtex5 FX130T

Resources	Set 1	Set 2	Set 3	Set 4
Block RAMs (298)	0	3	5	0
DSP48 (320)	1	1	1	3
LUTs	2982 (3%)	3505 (3%)	4582 (5%)	3660 (4%)
Max Freq (MHz)	115	124.1	104.7	118.8

TABLE IX
CCSDS121 IP Core: Synthesis Results on RTG4 150

Resources	Set 1	Set 2	Set 3	Set 4
MACC	3 (1%)	3 (1%)	3 (1%)	4 (1%)
RAM64x18 RT	9 (5%)	10 (5%)	19 (10%)	11 (6%)
LUTs	4137 (3%)	4650 (4%)	6196 (5%)	5259 (4%)
Max Freq (MHz)	51.8	49.7	48.2	38.7

A. CCSDS121 IP Core Synthesis Results

The synthesis of the CCSDS121 IP core has been performed with four different sets of generic parameters, shown in Table VII. The block size J varies from 16 to 64 and the size of the output buffer is set to 32 bits. The image size has been tailored up to $1024 \times 1024 \times 1024$. The set of generic parameters in the last row of Table VII enables run-time configuration.

Tables VIII and IX show the synthesis results for the Xilinx Virtex5 FX130T and the RTG4 150 among different targeted devices. The memory increase related to J is reflected in the obtained results, as it is higher for the parameter sets with increased J . The difference in resource usage when run-time configuration is enabled can be noticed by comparing the results for set1 and set4, where the latter provides the same generic parameters as the former, but with run-time configuration enabled.

TABLE X
CCSDS123 IP Core: Implementation Target Sizes

Sensor	Ny	Nx	Nz	bpp
LANDSAT	1024	1024	6	8
AVIRIS	680	512	224	16
AIRS	90	135	1501	14

TABLE XI
CCSDS123 IP Core: Synthesis Configuration Parameters

Parameter	Value
RESET_TYPE	1
EDAC	0
ENCODING_TYPE	1
ENCODER_SELECTION_GEN	1
W_BUFFER_GEN	32
IS_SIGNED_GEN	uint
ENDIANESS_GEN	be
P_MAX	3
PREDICTION_GEN	0
LOCAL_SUM_GEN	0
OMEGA_GEN	13
R_GEN	32
VMAX_GEN	3
VMIN_GEN	-1
T_INC_GEN	6
INIT_COUNT_E_GEN	1
ACC_INIT_TYPE_GEN	0
ACC_INIT_CONST_GEN	5
RESC_COUNT_SIZE_GEN	6
U_MAX_GEN	16

The CCSDS121 IP core is able to accept one sample per clock cycle. According to the synthesis results, it reaches a maximum clock frequency of 124.1 MHz, and so a maximum throughput of 124.1 MSamples per second can be obtained.

B. CCSDS123 IP Core Synthesis Results

The CCSDS123 IP core has been synthesized targeting specific images summarized in Table X. Specific parameters related to the image configuration are varied for created situations to be representative of different acquisition scenarios: multispectral, hyperspectral and ultraspectral. Some of these values, like P and P_MAX values have been established from the performance studies in [17]. Synthesis is performed assuming every compression order for each image: BIP, BIP-MEM, BSQ, and BIL, with the same common configuration parameters shown in Table XI. These common configuration parameters have been considered

TABLE XII
CCSDS123 IP Core: Synthesis Results on Virtex5 FX130

LANDSAT				
Resources	BIP	BIP-mem	BSQ	BIL
Block RAM (298)	2	0	1	5
DSP48s (320)	7	7	5	7
LUTs	3685 (5%)	4537 (6%)	4718 (6%)	4242 (6%)
Max. Frequency (MHz)	154.3	143.8	112.4	113.7
Throughput (Msamples/sec)	153.5	89.38	8.74	20.64
AVIRIS				
Resources	BIP	BIP-mem	BSQ	BIL
Block RAM (298)	74	10	1	74
DSP48s (320)	8	10	5	10
LUTs	4723 (6%)	5459 (7%)	5825 (8%)	5193 (7%)
Max. Frequency (MHz)	112.4	112.7	112.6	113.2
Throughput (Msamples/sec)	111.82	70.05	8.75	20.55
AIRS				
Resources	BIP	BIP-mem	BSQ	BIL
Block RAM (298)	123	11	1	123
DSP48s (320)	8	8	6	10
LUTs	4535 (6%)	5345 (7%)	5359 (7%)	5000 (7%)
Max. Frequency (MHz)	111.1	112.7	108.8	114.2
Throughput (Msamples/sec)	110.52	70.05	8.46	20.73
RUN-TIME CONFIGURABLE				
Resources	BIP	BIP-mem	BSQ	BIL
Block RAM (298)	74	10	1	74
DSP48s (320)	10	13	9	10
LUTs	5815 (8%)	6614 (9%)	6692 (8%)	6309 (8%)
Max. Frequency (MHz)	113.9	125	105.9	113.4
Throughput (Msamples/sec)	113.3	77.69	8.23	20.59

TABLE XIII
CCSDS123 IP Core: Synthesis Results on RTG4 150

LANDSAT				
Resources	BIP	BIP-mem	BSQ	BIL
MACC	7 (2%)	7 (2%)	7 (2%)	7 (2%)
RAM64x18_RT	31 (15%)	33 (16%)	25 (12%)	38 (19%)
RAM1K18_RT	4 (2%)	0 (0%)	1 (1%)	10 (5%)
LUTs	4799 (4%)	5996 (4%)	5973 (4%)	5211 (4%)
Maximum Frequency	78.3	78.3	68.5	75.2
AVIRIS				
Resources	BIP	BIP-mem	BSQ	BIL
MACC	13 (3%)	13 (3%)	11 (3%)	13 (3%)
RAM64x18_RT	62 (30%)	64 (31%)	36 (18%)	65 (31%)
RAM1K18_RT	129 (62%)	1 (0%)	1 (1%)	135 (65%)
LUTs	7174 (5%)	7569 (5%)	7123 (5%)	7572 (5%)
Maximum Frequency	61	69.3	70.1	61.1
AIRS				
Resources	BIP	BIP-mem	BSQ	BIL
MACC	7 (2%)	7 (2%)	6 (1%)	7 (2%)
RAM64x18_RT	20 (10%)	22 (11%)	30 (15%)	30 (15%)
RAM1K18_RT	278 (134%)	22 (11%)	0 (0%)	275 (134%)
LUTs	7516 (5%)	7281 (5%)	6780 (5%)	7925 (6%)
Maximum Frequency	59.4	77.6	70.7	58.5
RUN-TIME CONFIGURABLE				
Resources	BIP	BIP-mem	BSQ	BIL
MACC	16 (4%)	16 (4%)	14 (4%)	16 (4%)
RAM64x18_RT	64 (31%)	66 (32%)	38 (19%)	67 (32%)
RAM1K18_RT	129 (62%)	1 (1%)	1 (1%)	135 (65%)
LUTs	9254 (7%)	9796 (7%)	9269 (7%)	9624 (7%)
Maximum Frequency	60.2	69.3	54.6	56.7

for compile-time configuration. In particular, the AVIRIS image has been also considered also for synthesis with run-time configuration.

The synthesis results are shown in Tables XII and XIII, for Virtex5 FX 130T and Microsemi RTG4 150 devices, respectively. Implementation results depend on the selected architecture and the image size. The architectural differences between the BIP and BIP-MEM architectures are reflected in the lower use of block RAMs by the latter, since it makes use of an additional external memory. Whereas both architectures are able to accept one sample per clock cycle, the memory in the BIP-MEM imposes a penalty throughput. The maximum throughput for the BIP and BIP-MEM architectures is 153.5 and 89.38 MSamples per second, respectively. The BIL architecture presents a similar resource usage as BIP, but a lower throughput, as expected considering the data dependencies when compressing the samples in a line. The serial scheduling for the BSQ architecture reveals the lowest resource usage, but also the lowest possible throughput among all the implementations. All the summarized implementations can be implemented on the Virtex5 device.

Analogous results are obtained when targeting the RTG4 device, but a peculiarity is found for the AIRS image. The size of the image and the FIFOs used in the design for some architectures prevents memory constraints to be met. Specifically, the high amount of bands in the AIRS image

requires more embedded memory than the available in the RTG4 for the BIP and BIL architectures.

Whereas look-up table (LUT) and digital signal processor (DSP) usage is low in general, the usage of block RAM for the BIL architecture stands out for RTG4 device. It is expected that the versatility of the IP core might be further improved by considering an external memory interface in the BIL architecture, in order to reduce the usage of block RAM and make the implementation feasible also for bigger images.

Finally and for clarity, Fig. 23 comprehends the usage resource considering every predictor architecture for the run-time configuration case, as graphics, for both architectures Virtex5 FX130 and RTG4 150.

C. Comparison With Existing Implementations

Comparison of the SHyLoC IP Core with other state-of-the-art implementations is not straightforward due to its high amount of configurations with several performance capabilities, which are generally not present in other implementations. Most existing implementations do not offer the selection of all the spectrum of configuration options offered by the CCSDS standards; and limit the selection to a subset that enables to achieve a particular tradeoff between compression efficiency and complexity. Nevertheless, we present in Table XIV a comparison

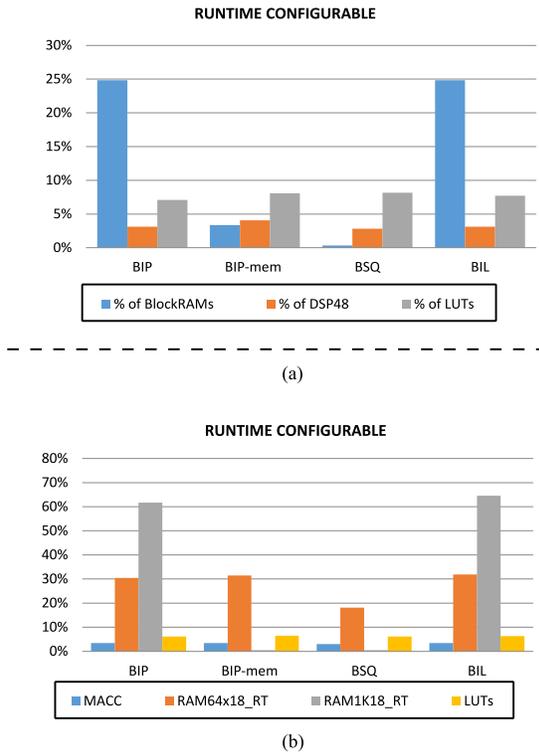


Fig. 23. CCSDS123 IP core resource usage. (a) Results for Virtex5 FX130. (b) Results for RTG4 150.

TABLE XIV
Comparison With Other State-of-the-Art Implementations

	HyLoC	SHyLoC	NASA FL	DSCAL
Compression order	BSQ	BIP	BIP	BIP
Number of bands for prediction	3	3	3	3
Max. bit depth	16	16	16	16
Image dimensions (Nx, Ny, Nz)	680, 512, 224	640, 32, 427	4096, 4096, 4096	
Target device	V5 FX130	V5 FX130	V5 FX130	V5 FX130
Device utilization				
- Slice LUTs	2342	5815	12697	5532
- BRAM	74	0	8	22
- DSP48E	1	10	3	6
Max. Frequency (MHz)	134	113.9	40	120
Throughput (MSamples/s)	11	113.3	40	120

between several implementations of the CCSDS lossless algorithms based on the configuration with the highest performance in terms of throughput (MSamples per second) that each of the implementations can achieve for a target Xilinx Virtex 5 FPGA. In particular, we compare with our previous HyLoC development [18], with an implementation of the fast-lossless algorithm (adopted for standardization and later formalized as CCSDS-123) [34], and with DSCAL [35], which is a technology-specific implementation of the CCSDS-123 for Xilinx.

Note that for the fairness of the comparison, in the case of SHyLoC we consider the set of parameters in

which run-time configuration is enabled, although a higher throughput is achievable when only compile-time options are considered.

We can observe that SHyLoC and DSCAL exhibit similar performance results, which are significantly higher than the other implementations. The main difference between these two implementations is versatility, SHyLoC is technology independent and offers all configuration modes, whereas DSCAL has been coded and optimized for Xilinx, and for a particular compressor configuration. Moreover, SHyLoC takes care of the arrangement and storage of the incoming samples from the sensor, while DSCAL maintains these tasks outside the IP core. This fact explains the higher Block RAM (BRAM) usage in SHyLoC, as for the presented configuration the storage is internal to the FPGA. Nevertheless, thanks to the multiple configuration options offered, a tradeoff can be found with SHyLoC by implementing the BIP-mem architecture, where the block RAM usage is reduced to 10; and the throughput is still high (77.9 MSamples per second).

Other approaches for implementing the CCSDS-123 algorithm include multicore implementations, such as the one in [36]. This approach exploits the possibility of parallelizing some operations by finding a suitable scheme for splitting the image in independent blocks. Their implementation is 3.20 times faster than a single threaded CPU when four cores are utilized. Taking into account the throughput results of the single threaded implementations presented in [37], where the authors report a maximum of 5 MSamples per second for an Intel i7 processor, the maximum throughput of the multicore implementation would be around 16 MSamples per second, which is far from the presented FPGA alternatives. If a higher throughput is desired, graphics processing units (GPU) implementations, such as the one presented in [38], offer results of up to 300 MSamples per second. Nevertheless, despite the high performance of GPUs, they are currently not suited for space applications, due to their high power consumption and the lack of tolerance to radiation.

In summary, FPGA implementations, and in particular SHyLoC, offer a versatile and energy efficient solution to implement lossless compression on-board satellites. Different tradeoffs between FPGA utilization, throughput and compression efficiency can be found thanks to the high amount of compile time and run-time configuration options offered. Furthermore, SHyLoC can be easily adapted to be implemented in new technologies designed to meet the space demands, such as the European BRAVE FPGA [39].

VII. CONCLUSION

Hardware architectures and VHDL descriptions of CCSDS121 and CCSDS123 IP cores have been presented. Although the cores are in principle independent, these IP cores can be combined into a logical single entity, in which the CCSDS121 is connected at the output of the CCSDS123 IP in order to perform the block-adaptive

encoding process as recommended by the CCSDS 123.0-B-1 standard. The designed control interface for the two IP cores eases considerably the interaction of the IP cores, providing plug-and-play connectivity. The wide and versatile parametrization and configuration options offered by both IPs allow them to be configured from a set of compile-time or run-time parameters. It is possible to tailor the IP cores in order to reduce their complexity if needed, by using the provided compile-time parameters. This is particularly important for the CCSDS123 IP core, more complex than the CCSDS121, when dealing with devices with a limited amount of resources.

The SystemC models implemented as part of this paper have demonstrated how the data flow in the different compression orders makes it mandatory to design different hardware architectures for the IP cores. The selected compression order affects both throughput and the permanence of the already processed samples in the system. The length of the data path from the moment in which intermediate values are generated until they are retrieved to continue the computations, together with the image size and the number of previous bands used for prediction P , affect the amount of memory needed.

Different implementation results have been presented for the specific FPGAs Virtex5 FX130 and RTG4 150. A maximum compression throughput of 153.5 Msamples per second has been stated. The LUT and DSP usage has proven not to be a concern, but implementation has exposed a size limitation for the CCSDS123 IP core regarding the block RAMs for the RTG4 device, being the IP core not able to fit on the device when compressing images with a high number of bands and columns. Technology or vendor specific architectural optimizations can be introduced in future implementations of SHyLoC to help overcoming this limitation.

In summary, SHyLoC offers a versatile and reusable implementation of the CCSDS 121.0-B-2 and CCSDS 123.0-B-1 standards that can potentially meet the requirements of numerous space missions in the future, in terms of compression efficiency, low complexity and high throughput of the final implementation.

ACKNOWLEDGMENT

The authors would like to thank Thales Alenia Space, Spain, for implementation of the IP cores on DARE 180 nm ASIC technology and to TELETTEL, Greece, for validation of the IP cores on its proprietary demonstrator.

REFERENCES

- [1] E. S. A. (ESA), Gaia. 2013. [Online]. Available: http://www.esa.int/Our_Activities/Space_Science/Gaia
- [2] E. S. A. (ESA), Proba missions. 2014. [Online]. Available: http://www.esa.int/Our_Activities/Technology/Proba_Missions
- [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. Zabala, R. Vitulli, and X. Pons
Impact of CCSDS-IDC and JPEG2000 compression on image quality and classification
J. Elect. Comput. Eng., vol. 2012, pp. 4:4–4:4, Jan. 2012. [Online]. Available: <http://dx.doi.org/10.1155/2012/761067>
- [5] I. Blanes and J. Serra-Sagristá
Cost and scalability improvements to the karhunen-loève transform for remote-sensing image coding
IEEE Trans. Geosci. Remote Sens., vol. 48, no. 7, pp. 2854–2863, Jul. 2010.
- [6] C. Egho and T. Vladimirova
Hardware acceleration of the integer karhunen-loève transform algorithm for satellite image compression
In *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2012, pp. 4062–4065.
- [7] I. Blanes and J. Serra-Sagristá
Pairwise orthogonal transform for spectral image coding
IEEE Trans. Geosci. Remote Sens., vol. 49, no. 3, pp. 961–972, Mar. 2011.
- [8] D. Taubman and M. Marcellin
JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards and Practice (ser. The Springer International Series in Engineering and Computer Science). New York, NY, USA: Springer, 2012. [Online]. Available: <https://books.google.es/books?id=y7HeBwAAQBAJ>
- [9] R. Guerra, Y. Barrios, M. Díaz, L. Santos, S. López, and R. Sarmiento
A new algorithm for the on-board compression of hyperspectral images
Remote Sens., vol. 10, no. 3, 2018. [Online]. Available: <http://www.mdpi.com/2072-4292/10/3/428>
- [10] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam
Availability analysis for satellite data processing systems based on SRAM FPGAs
IEEE Trans. Aerosp. Electron. Syst., vol. 52, no. 3, pp. 977–989, Jun. 2016.
- [11] R. B. Atitallah, V. Viswanathan, N. Belanger, and J. L. Dekeyser
FPGA-Centric design process for avionic simulation and test
IEEE Trans. Aerosp. Electron. Syst., vol. 54, no. 3, pp. 1047–1065, Jun. 2018.
- [12] M. R. Maheshwarappa, M. D. J. Bowyer, and C. P. Bridges
Improvements in CPU FPGA performance for small satellite SDR applications
IEEE Trans. Aerosp. Electron. Syst., vol. 53, no. 1, pp. 310–322, Feb. 2017.
- [13] *Lossless Data Compression, Recommended Standard CCSDS 121.0-B-2*, The Consultative Committee for Space Data Systems, 2012.
- [14] *Image Data Compression, Recommended Standard CCSDS 122.0-B-1*, The Consultative Committee for Space Data Systems, 2005.
- [15] *Lossless Multispectral and Hyperspectral Image Compression, Recommended Standard CCSDS 123.0-B-1*, The Consultative Committee for Space Data Systems, 2012.
- [16] J. E. Sánchez, E. Auge, J. Santalo, I. Blanes, J. Serra-Sagristá, and A. Kiely
Review and implementation of the emerging CCSDS recommended standard for multispectral and hyperspectral lossless image coding
In *Proc. 1st Int. Conf. Data Compression, Commun. Process.*, Jun. 2011, pp. 222–228.
- [17] E. Augé, J. Enrique Sánchez, A. Kiely, I. Blanes, and J. Serra-Sagristá
Performance impact of parameter tuning on the CCSDS-123 lossless multi- and hyperspectral image compression standard
vol. 7, Aug. 2013, Art. no. 074594.

- [18] L. Santos, J. F. López, R. Sarmiento, and R. Vitulli
FPGA implementation of a lossy compression algorithm for hyperspectral images with a high-level synthesis tool
In *Proc. NASA/ESA Conf. Adaptive Hardware Syst.*, Jun. 2013, pp. 107–114.
- [19] H. Shen, W. D. Pan, Y. Dong, and Z. Jiang
Golomb-Rice coding parameter learning using deep belief network for hyperspectral image compression
In *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2017, pp. 2239–2242.
- [20] G. Yu, T. Vladimirova, X. Wu, and M. N. Sweeting
A new high-level reconfigurable lossless image compression system for space applications
In *Proc. NASA/ESA Conf. Adaptive Hardware Syst.*, Jun. 2008, pp. 183–190.
- [21] N. Kranitis, I. Sideris, A. Tsigkanos, G. Theodorou, A. Paschalis, and R. Vitulli
Efficient field-programmable gate array implementation of ccsds 121.0-b-2 lossless data compression algorithm for image compression
J. Appl. Remote Sens., vol. 9, no. 1, 2015, Art. no. 097499. [Online]. Available: <http://dx.doi.org/10.1117/1.JRS.9.097499>
- [22] B. Hopson, K. Benkrid, D. Keymeulen, and N. Aranki
Real-time CCSDS lossless adaptive hyperspectral image compression on parallel gpgpu amp; multicore processor systems
In *Proc. NASA/ESA Conf. Adaptive Hardware Syst.*, Jun. 2012, pp. 107–114.
- [23] 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.
- [24] J. Fjeldtvedt, M. Orlandi, 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.
- [25] L. Santos, A. Gómez, P. Hernández-Fernández, R. Sarmiento, and L. Fossati
SystemC modeling of lossless compression IP cores for space applications
In *Proc. Conf. Des. Architectures Signal Image Process.*, Oct. 2016, pp. 65–72.
- [26] E. S. A. (ESA), CCSDS 123.0-B-1 multispectral and hyperspectral lossless data compression, 2013. [Online]. Available: http://www.esa.int/TEC/OBDP/SEM069KOXDG_2.html
- [27] Group on Interactive Coding of Images - Universitat Autònoma de Barcelona, Emporda software (a CCSDS-123 implementation), 2013. [Online]. Available: <http://gici.uab.es/GiciWebPage/emporda.php>
- [28] European Space Agency, ESA IP Cores, 2016. [Online]. Available: http://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/About_ESA_IP_Cores
- [29] *Appendix 1 to ESA ITT AO/1-8032/14/NL/AK, Statement of Work, CCSDS Lossless Compression IP-Cores for Space Applications*, European Space Agency, 2014.
- [30] C. G. AB, GRLIB IP library, 2017. [Online]. Available: <https://www.gaisler.com/index.php/products/ipcores/soclibrary>
- [31] “IEEE standard for a 32-bit microprocessor architecture IEEE Std. 1754-1994, pp. 1–, 1995.
- [32] T. Schuster, R. Meyer, R. Buchty, L. Fossati, and M. Berekovic
SoCRocket - a virtual platform for the European Space Agency’s SoC development
In *Proc. 9th Int. Symp. Reconfigurable Commun.-Centric Syst.-Chip*, May 2014, pp. 1–7.
- [33] Synopsys, Synplify Premier. 2019, [Online]. Available: <https://www.synopsys.com/implementation-and-signoff/fpga-based-design/synplify-premier.html>
- [34] D. Keymeulen, N. Aranki, A. Bakhshi, H. Luong, C. Sarture, and D. Dolman
Airborne demonstration of FPGA implementation of fast lossless hyperspectral data compression system
In *Proc. NASA/ESA Conf. Adaptive Hardware Syst.*, Jul. 2014, pp. 278–284.
- [35] G. Theodorou, N. Kranitis, A. Tsigkanos, and A. Paschalis
High performance CCSDS 123.0-b-1 multispectral& hyperspectral image compression implementation on a space-grade sram FPGA
In *Proc. 5th Int. Workshop On-Board Payload Data Compression*, Frascati, Italy, 2016, pp. 28–29.
- [36] M. Olaru and M. Craus
Lossless multispectral and hyperspectral image compression on multicore systems
In *Proc. 21st Int. Conf. Syst. Theory, Control Comput.*, Oct. 2017, pp. 175–179.
- [37] 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.
- [38] R. L. Davidson and C. P. Bridges
GPU accelerated multispectral EO imagery optimised CCSDS-123 lossless compression implementation
In *Proc. IEEE Aerosp. Conf.*, Mar. 2017, pp. 1–12.
- [39] E. S. Agency, igh density European rad-hard SRAM-based FPGA. 2017. [Online]. Available: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Shaping_the_Future/High_Density_European_Rad-Hard_SRAM-Based_FPGA_-_First_Validated_Prototypes_BRAVE



Lucana Santos received the Telecommunication Engineering degree from the University of Las Palmas de Gran Canaria, Las Palmas, Spain, in 2008, and the European Ph.D. degree from the Research Institute for Applied Microelectronics (IUMA), Las Palmas, Spain, in 2014.

Since then she has conducted her research activity at the Integrated System Design Division, IUMA. She was a visiting researcher with the European Space Research and Technology Centre in The Netherlands in 2011, where she did research on hardware architectures for hyperspectral and multispectral image compression on GPUs and FPGAs, funded by the European Network of Excellence on High Performance and Embedded Architecture and Compilation. From 2012 to 2017, she has participated actively in industrial projects in this field for Thales Alenia Space España and the European Space Agency. She has coauthored several scientific papers and has been a reviewer for the major international journals in her research areas. Her current research interests include hardware architectures for onboard data processing, video coding standards, and reconfigurable architectures.

Dr. Santos is currently a member of the CCSDS Multispectral/hyperspectral Data Compression (MHDC) working group sponsored by ESA.



Ana Gómez received the Telecommunication Engineering degree from the University of Las Palmas de Gran Canaria, Las Palmas, Spain, in 2013, and the master's degree in telecommunication technologies from the Research Institute for Applied Microelectronics, Las Palmas, Spain, in 2015.

Since then, she has conducted her research activity in the Integrated System Design Division at the Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, in the field of electronic and hardware architecture. In 2015, she started to work as researcher in an European Space Agency Research Project. She has coauthored several scientific papers. Her current research interests include an efficiently use of multispectral and hyperspectral compression algorithms, which she is focuses on to get her Ph.D.



Roberto Sarmiento is Full-Professor at the Telecommunication Engineering School, University of Las Palmas de Gran Canaria, Spain, in the area of Electronic Engineering. He contributed to set this school up, he was the Dean of the Faculty from 1994 to 1995 and Vice-Chancellor for Academic Affairs and Staff at the ULPGC from 1998 to 2003. He is a founder of the Research Institute for Applied Microelectronics and the Director of the Integrated Systems Design Division of this Institute. He has authored and coauthored more than 60 journal papers and book chapters and more than 150 conference papers. His current research interest includes electronics system on-board satellites.

Prof. Sarmiento is the recipient five six-year research periods by the National Agency for the Research Activity Evaluation in Spain. He has participated in more than 50 projects and research programmes funded by public and private organizations.