IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 27, 2020, accepted March 8, 2020, date of publication March 13, 2020, date of current version March 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980767

SHyLoC 2.0: A Versatile Hardware Solution for On-Board Data and Hyperspectral Image Compression on Future Space Missions

YUBAL BARRIOS[1], ANTONIO J. SÁNCHEZ[1], LUCANA SANTOS[2], AND ROBERTO SARMIENTO[1]

[1]Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, 35001 Las Palmas de Gran Canaria, Spain
[2]European Space Research and Technology Centre, European Space Agency, 2220 AG Noordwijk, The Netherlands

Corresponding author: Yubal Barrios (ybarrios@iuma.ulpgc.es)

This work was supported in part by the ESA through the project CCSDS Lossless Compression IP-Core for Space Applications under ESA Contract 4000113182/15/NL/LF, and in part by the Spanish Government through the project PLATINO under Grant TEC2017-86722-C4-1-R.

ABSTRACT In this paper, we present the design, implementation and results of a set of IP cores that perform on-board hyperspectral image compression according to the CCSDS 123.0-B-1 lossless standard, specifically designed to be suited for on-board systems and for any kind of hyperspectral sensor. As entropy coder, the sample-adaptive entropy coder defined in the 123.0-B-1 standard or the low-complexity block-adaptive encoder defined by the CCSDS 121.0-B-2 lossless standard could be used. Both IPs, 123.0-B-1 and 121.0-B-2, are part of SHyLoC 2.0, and can be used together for compression of hyperspectral images, being also possible the compression of any kind of data using only the 121-IP. SHyLoC 2.0 improves and extends the capabilities of SHyLoC 1.0, currently available at the ESA IP Cores library, increasing its compression efficiency and throughput, without compromising the resources footprint. Moreover, it incorporates new features, such as the unit-delay predictor option defined by the CCSDS 121.0-B-2 standard, and burst capabilities in the external memory interface of the CCSDS 123-IP, among others. Dedicated architectures have been designed for all the possible input image sample arrangements, in order to maximise throughput and reduce the hardware resources utilization. The design is technology-agnostic, enabling the mapping of the VHDL code in different FPGAs or ASICs. Results are presented for a representative group of well-known space-qualified FPGAs, including the new NanoXplore BRAVE family. A maximum throughput of 150 MSamples/s is obtained for Xilinx Virtex XQR5VFX130 when the SHyLoC 2.0 CCSDS-123 IP is configured in Band-Interleaved by Pixel (BIP) order, using only the 4% of LUTs and less than the 1% of internal memory.

INDEX TERMS Hyperspectral imaging, compression algorithms, field programmable gate arrays, hardware implementations, space missions, on-board data processing.

I. INTRODUCTION

Nowadays, high-resolution hyperspectral imaging sensors are becoming more common in Earth Observation (EO) satellite missions due to their multiple applications for identification, surveillance and navigation purposes, among others. The Multispectral Instrument (MSI) on-board Sentinel-2 supported by ESA, the Hyperion imaging spectrometer integrated in the EO-1 NASA satellite or PRISMA, a mission fully funded by the Italian Space Agency (ASI) that combines a hyperspectral sensor with a medium-resolution panchromatic camera, are just a few examples of the use of multispectral and hyperspectral technology in the space environment [1]. This trend continues in the near future with the launch of new hyperspectral instruments, such as the Hyperspectral Infrared Imager (HyspIRI) mission, supported by NASA in order to provide critical information on natural disasters and vegetation health [2]; or the CHIME mission, planned as part of the new Copernicus 2.0 EO program supported by ESA [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Remigiusz Wisniewskis.

The large amounts of data acquired on satellites need to be either transmitted, stored or processed. Due to the limited computational and memory resources available in on-board hardware systems, processing or storing hyperspectral data on the satellite is not feasible. The limited downlink bandwidth with ground in comparison with the data size constitutes an additional bottleneck for missions that integrate this kind of sensors. This limitation will become more stringent in the near future with the progressive increase in the resolution of hyperspectral sensors [4], making on-board data compression mandatory.

Compression can be either lossless or lossy. Lossless compression preserves the information presented in the original data, being able to fully recover it during the decompression step. For that reason, this compression technique is interesting for the scientific community, maintaining the fidelity of the data captured by the sensor and useful for applications such as classification, and target and anomaly detection. On the other hand, lossy compression yields higher compression ratios than lossless techniques introducing losses in the compressed image; this is, the recovered information is not identical to the original one [5]. Lossy compression is a key enabler for deep space missions with high-resolution sensors on-board, in order to exchange information with ground.

Compression algorithms are normally based on two stages: decorrelation of the information acquired by the instrument and entropy coding. Decorrelation is done either using transform or prediction techniques. Transform-based methods, such as the widely used JPEG2000 [6] or the most recent Karhunen-Loeve Tranform (KLT) [7] and Pairwise Orthogonal Transform (POT) [8], are most focused on standard (2D) images compression and they are preferred for lossy compression because of the high compression ratios they achieve. The implementation of compression algorithms in space missions still supposes a challenge, taking into account hardware limitations Hence, transform-based algorithms are in general too complex to be implemented on-board satellites. Therefore, prediction-based techniques are preferred because they represent a good trade-off between compression efficiency (compression ratio achieved) and computational complexity [9].

With the aim of implementing efficient data compression algorithms with a reduced computational complexity, well suited for on-board systems, the Consultative Committee for Space Data Systems (CCSDS), an international organization comprised by the main space agencies in the world to define a common way for developing space data and information systems, has published different data compression standards specifically designed for space applications. Among these standards, the CCSDS 123.0-B-1 [10] focuses on the compression of multispectral and hyperspectral images, while the CCSDS 121.0-B-2 [11] constitutes a universal compressor applicable to any kind of data. Both standards are prediction-based compression standards, well suited for low-complexity implementations. The ESA IP Cores library [12] has recently incorporated a hardware solution for compressing data and hyperspectral images, named as SHyLoC 1.0 [13], [14]. This solution includes two different IP cores, compliant with the CCSDS 123.0-B-1 and CCSDS 121.0-B-2 lossless compression standards. These two IP cores have been designed with standard interfaces allowing its connection to the rest of the hardware platform in a plug & play manner.

SHyLoC 1.0 is a technology-agnostic and low-complexity hardware solution, favouring the design reusability between applications or projects, but specifically optimized for Field-Programmable Gate Arrays (FPGAs). Radiation-Hardened by Design (RHBD) FPGAs are becoming increasingly employed for space applications in comparison to Application-Specific Integrated Circuits (ASICs) because their lower cost, high performance and low-power consumption [15]. FPGAs allow the efficient implementation of low-complexity architectures on-board satellites, including the capability of changing dynamically all or some parts of the functionality of the on-board system during mission lifetime in order to adapt it to new requirements.

NanoXplore FPGAs devices have special interest for the European Space Agency because they offer a family of radiation-hardened reprogrammable FPGAs developed in Europe, avoiding the dependencies with foreign technologies that have dominated the space market during the last decades. The youngest and smallest device of the Big Reprogrammable Array for Versatile Environments (BRAVE) family, named NG-MEDIUM, is based on 65nm CMOS technology and includes 35k 4-input Look-Up Tables (LUTs) and D type Flip-Flops (DFFs), 2.8 Mb of dedicated RAM and 112 Digital Signal Processing (DSP) blocks, among other elements. With higher offer of logic resources, the NG-LARGE is almost four times bigger than the NG-MEDIUM (137k LUTs and 129k DFFs), 9.4 Mb of embedded RAM and 384 dedicated DSPs, together with an ARM Cortex-R5 core [16]. Different radiation hardening techniques are combined in the BRAVE FPGA family, such as an specific manufacturing process, Error Detection And Correction (EDAC) for dedicated memory blocks, Triple Modular Redundancy (TMR) flip-flops, Double Modular Redundancy (DMR) clock-tree, or a background scrubber to preserve the integrity of the FPGA memory configuration.

SHyLoC 1.0 lacks optional functions and configuration parameters defined by the standards and hence is not fully standard compliant. Moreover, it is optimized for low area footprint, and therefore it presents limited throughput in some configurations. This paper presents SHyLoC 2.0, an improved hardware solution for data and hyperspectral image compression on-board satellites, fully compliant with the CCSDS 123.0-B-1 and CCSDS 121.0-B-2 lossless compression standards. These IP cores are described in VHDL focusing on low-complexity and fitting on available on-board hardware resources. New features not presented in current state-of-the-art implementations are included, such as the optional unit-delay predictor defined by the CCSDS 121.0-B-2 standard or the burst transfers in the

communication with external memory on the CCSDS 123-IP in order to improve both throughput and memory bandwidth, among others. The implementation of prediction architectures that store intermediate results in an external memory allows to manage images with bigger size both in the spatial and the spectral domain and even with higher bit resolution, together with a reduction of the internal resources utilization. Moreover, the designed IP cores are technology-independent, being possible to implement them on ASICs and on a high variety of space-qualified FPGAs, including the new and aforementioned BRAVE family, entailing a novelty regarding other FPGA-based compression solutions. They also allow to select the suitable values for all the configuration parameters included in both standards, making possible to configure them for the specific necessities of a target application.

The rest of the paper is structured as follows. Section II provides a brief description of both CCSDS 121.0-B-2 and CCSDS 123.0-B-1 lossless compression standards. Then, Section III details the hardware implementation of each IP, highlighting the new features introduced with respect to SHyLoC 1.0. Next, Section IV analyses the mapping results for different space-qualified FPGA technologies, comparing them with the existing implementations in the state-of-the-art. Finally, Section V summarizes the main conclusions about this work.

II. ALGORITHM DESCRIPTION

A. CCSDS 121.B-0-2

The CCSDS 121.0-B-2 standard [11] describes a low-complexity universal lossless data compressor, specifically designed to work on-board satellites It consists in two main stages: a preprocessing stage and an entropy coder, as shown in Fig.1.

The preprocessor removes correlation between consecutive input samples and maps them into unsigned values which are then passed to the block-adaptive entropy coder in order to be properly encoded. The CCSDS 121.0-B-2 standard defines a simple and reversible unit-delay predictor, which uses just the previous sample as an estimator of the current one. Reference samples must be periodically inserted in the output bitstream in order to be able to regenerate the input image during the decompression step. In any case, this preprocessor can be omitted, working the CCSDS 121.0-B-2 algorithm only as entropy coder.

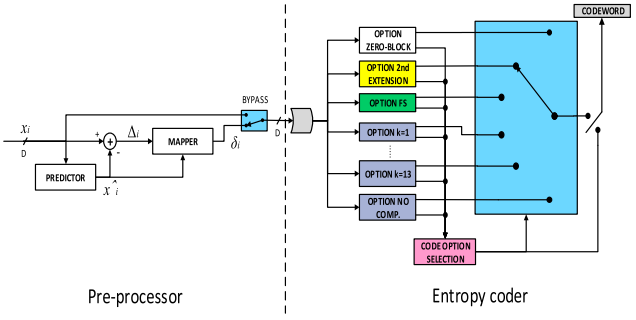


FIGURE 1. CCSDS-121 standard- Block-adaptive coder with preprocessing stage.

The entropy coder of the CCSDS 121.0-B-2 is based in adaptive Rice coding, a subset of Golomb codes that uses a power of two value as tunable parameter. This parameter makes Rice coding efficient for hardware implementations, because multiplication and division by powers of two can be easily implemented using binary arithmetic as logic shifts. This entropy encoder achieves efficient performance over different overlapping ranges of entropy. The incoming preprocessed samples are grouped into blocks of size J, parameter defined by the user. In this encoder, all the possible compression options are concurrently applied to a block of J consecutive input samples. Finally, each block is coded with the option which produces the shortest output, among the available ones:

- Fundamental sequence (FS). Each input sample δ_i is encoded as δ_i zeroes followed by a one.
- Sample splitting. First, each input sample is split by removing the k least significant bits. The MSBs of δ_i are coded with the FS, while the LSBs are left uncompressed.
- Second-extension. Each pair of input samples δ_i and δ_{i+1} is transformed into a new symbol γ, according to the formula $\gamma = (\delta_i + \delta_{i+1})(\delta_i + \delta_{i+1} + 1)/2 + \delta_{i+1}$, and then coded using FS.
- Zero-block. This option is thought for low-entropy images and it denotes one or more consecutive blocks of all-zeroes. It is the only case where a single codeword may represent more than one compressed block.
- No compression. Input samples are outputted without compressing them (i.e. the entropy coder is bypassed).

A unique identifier is attached to each compressed block for all the aforementioned compression options, in order to know which one of these options has been used.

B. CCSDS 123.B-0-1

The CCSDS 123.0-B-1 [10] is a lossless data compression standard specifically devised for multispectral and hyperspectral images, which is based on a predictive preprocessing stage as a way to reduce correlation among input samples. Experimental results in terms of compression ratio show that the CCSDS 123 standard is competitive with other state-of-the-art algorithms, providing the best trade-off between coding performance and computational complexity [17].

The CCSDS 123.0-B-1 standard has several options and configurable parameters, which may be tuned to modify the compression efficiency. The characteristics of the acquired images and the sensor should be taken into account when configuring these parameters. Depending on the hyperspectral sensor on-board, samples can be arranged in Band Sequential (BSQ) order (for snapshot or rotational filters, where the samples in a band are handled before processing the next one) or Band interleaved (BI) order. In the latter case

two arrangements are distinguished, Band Interleaved by Pixel (BIP), typical of whisk broom scanners, where a sample is acquired in all its bands before capturing the adjacent pixel, and Band Interleaved by Line (BIL), used by push broom sensors, obtaining a line of samples in the spatial domain for all the bands before acquiring the next one.

This standard also defines a header to allow a proper decompression, which is appended at the beginning of each compressed image, where the selected values for all the configuration options are specified.

1) PREDICTOR

The preprocessor stage of the CCSDS 123.0-B-1 standard estimates the value of each input sample using a set of samples in the vicinity of the current one, counting on samples in the same band as well as in previously processed bands, as it is shown in Fig. 2. Hence, the ordering of the input samples does not affect the compression efficiency, as opposed to the CCSDS 121.0-B-2 standard. The number of bands P used for prediction can be configured between 0 and 15, although it has been observed that no significant improvements are achieved when setting values of P higher than 3 [18].

The predictor processes the input image in a single pass, independently of the order in which the input samples are arranged. The compression algorithm is illustrated in Fig. 3, and summarized in the following lines. For each input sample, first a *local sum* $\sigma_{z,y,x}$ for the current band as well as the previous P bands is computed, by combining the values of the neighbour samples in the same band.

The set of samples which are used to compute these local sums is determined by the selected local sum type: in the neighbour-oriented mode, all the previously processed adjacent samples are used, while in the column-oriented mode just the sample right above is used. Equation 1 describes the way of computing the local sums with the neighbour-oriented mode, while equation 2 does the same for the column-oriented mode, both covering also the corner cases.

Then, the *local differences* are computed, by subtracting the neighbour sample values from the previously computed

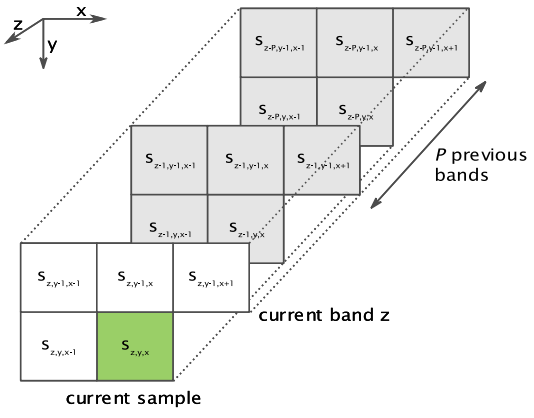


FIGURE 2. Set of samples used for prediction.

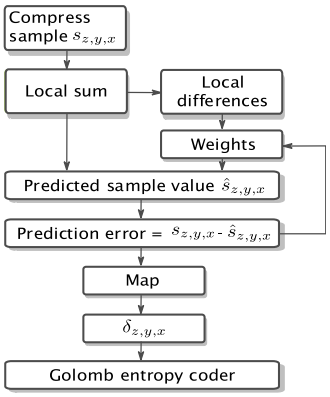


FIGURE 3. Scheme of the CCSDS 123 compression algorithm.

local sums. The local differences are defined for every pixel except for the first one (with $x = 0$ and $y = 0$). The *central difference* is computed as $d_{z,y,x} = 4s_{z,y,x} - \sigma_{z,y,x}$, while the *directional differences* are computed according to the equations 3, 4 and 5, as the standard defines [10]. More details about the theoretical basis behind these equations can be found in [19]. These are then grouped in the local differences vector $U_{z,y,x}$, which is built depending on the selected prediction mode. If the reduced mode is chosen, just the central differences of the P previous bands are included in the local differences vector. On the other hand, the directional differences of the current band are also included with the full mode.

In order to analyze the performance of the CCSDS 123.0-B-1 algorithm in terms of achieved bit per pixel per band (bpppb) depending on the combination of local sum and prediction mode used, a corpus of 7 representative multispectral and hyperspectral images (whose main features are summarized in Table 1) are used.

These results are shown in Table 2, observing that the optimal prediction scheme in terms of bpppb is highly dependent of the targeted sensor and the acquired image nature (e.g. if streaking artifacts are present or if the image is calibrated).

The selection of the local sum and prediction modes should achieve a tradeoff between the compression performance and the resources utilization. The use of the full prediction mode with respect to the reduced one supposes a slightly higher memory usage, due to the increased size of weight vectors ($P + 3$ elements versus P). On the other hand, when using neighbour-oriented local sums instead of the column-oriented

TABLE 1. Corpus of images used for performance analysis.

Instrument	Image size			Bit depth
	Nx	Ny	Nz	
AIRS	90	135	1501	12-14
M3-Target	640	1774	260	12
Hyperion	256	1024	242	12
AVIRIS(raw)	680	512	224	16
AVIRIS(calibrated)	677	512	224	16
MODIS-day	1354	2030	14	12
Landsat	1024	1024	6	8

TABLE 2. CCSDS123 performance in terms of bit per pixel per band (bpppb), depending on the selected local sum and prediction modes and using the sample-adaptive encoder.

Instrument	reduced + column	full + neighbour	reduced + neighbour
AIRS	4.72	4.40	4.41
M3-Target	3.09	5.08	6.71
Hyperion	4.33	4.65	5.23
AVIRIS(raw)	6.25	5.97	5.99
AVIRIS(calibrated)	4.22	3.88	3.91
MODIS-day	5.72	5.79	5.82
Landsat	3.73	3.37	3.44

mode, a slight increment in the logic resources consumption is observed, because of the additional calculations resulting of extending the neighbourhood for the prediction.

$$\sigma_{z,y,x} = \begin{cases} s_{z,y,x-1} + s_{z,y-1,x-1} + s_{z,y-1,x} + s_{z,y-1,x+1}, \\ \qquad y > 0,\ 0 < x < N_x - 1 \\ 4s_{z,y,x-1}, \\ \qquad y = 0,\ x > 0 \\ 2(s_{z,y-1,x} + s_{z,y-1,x+1}), \\ \qquad y > 0,\ x = 0 \\ s_{z,y,x-1} + s_{z,y-1,x-1} + 2s_{z,y-1,x}, \\ \qquad y > 0,\ x = N_x - 1 \end{cases} \tag{1}$$

$$\sigma_{z,y,x} = \begin{cases} 4s_{z,y-1,x}, & y > 0 \\ 4s_{z,y,x-1}, & y = 0,\ x > 0 \end{cases} \tag{2}$$

$$d^N_{z,y,x} = \begin{cases} 4s_{z,y-1,x} - \sigma_{z,y,x}, & y > 0 \\ 0, & x > 0,\ y = 0 \end{cases} \tag{3}$$

$$d^W_{z,y,x} = \begin{cases} 4s_{z,y,x-1} - \sigma_{z,y,x}, & x > 0,\ y > 0 \\ 4s_{z,y-1,x} - \sigma_{z,y,x}, & x = 0,\ y > 0 \\ 0, & x > 0,\ y = 0 \end{cases} \tag{4}$$

$$d^{NW}_{z,y,x} = \begin{cases} 4s_{z,y-1,x-1} - \sigma_{z,y,x}, & x > 0,\ y > 0 \\ 4s_{z,y-1,x} - \sigma_{z,y,x}, & x = 0,\ y > 0 \\ 0, & x > 0,\ y = 0 \end{cases} \tag{5}$$

Then, a weighted sum of the elements in the local differences vector is computed, which is in turn used to compute the *predicted sample* $\hat{s}_{z,y,x}$. This sum makes use of an internal *weight vector*, $W_{z,y,x}$.

A weight vector is separately maintained for each band, and their components are updated with each new sample based on the prediction residual, the local differences and some user-defined parameters. Finally, the prediction residual is mapped into an unsigned integer $\delta_{z,y,x}$, which is passed to the entropy coder.

The CCSDS 123.0-B-1 standard defines two possible ways of setting the initial values for the components of the weight vectors. With the default initialization, the components of the weight vectors are set to fixed values, equal for all the bands. With the custom weight initialization, the initial weights are provided by the user, and each band may have different values.

2) SAMPLE-ADAPTIVE ENCODER

With respect to the entropy coding, the CCSDS 123.0-B-1 standard allows two alternatives. On the one hand, it is possible to use the block-adaptive coder defined in the CCSDS 121.0-B-2 standard [11] described in Section II-A. On the other hand, a sample-adaptive coder is also proposed, which is a more sophisticated version of an adaptable Rice coder. With this encoder, samples are compressed individually instead of in blocks. The code used for each particular sample depends on the image statistics, which are updated with every new sample depending on the chosen compressor configuration. Statistics are computed for each separate band independently.

The performance in terms of compression efficiency of the CCSDS 123.0-B-1 algorithm for the instruments detailed in Table 1 has been evaluated depending on the selected option for the entropy coding stage [20], obtaining the results summarized in Table 3. For this comparison, the best combination of local sum calculation and prediction mode is considered for each targeted sensor, taking into account the results shown in Table 2. In general, the sample adaptive encoder offers better performance for hyperspectral and multispectral data. This is mostly noticeable in BIP order, where the block-adaptive coder under-performs, as it forms the blocks in the way the samples arrive, even mixing spatial and spectral information. Sample adaptive is also less complex than the block-adaptive one.

III. HARDWARE IMPLEMENTATION

SHyLoC 2.0 is comprised by two IP cores that implement the CCSDS 121.0-B-2 and CCSDS 123.0-B-1 lossless compression standards. The aim of these cores is to provide to the user a versatile solution for compressing hyperspectral images (3D), extended to any kind of uni-dimensional data acquired on-board satellites. A general overview of SHyLoC 2.0 is shown in Fig. 4.

Fig. 4a shows the SHyLoC configuration to perform hyperspectral image compression using the CCSDS 123.0-B-1 standard with the sample-adaptive encoder. The prediction stage of the CCSDS 123.0-B-1 standard can be also combined with the block-adaptive encoder defined by the CCSDS 121.0-B-2 standard (Fig. 4b). If the input data is acquired by any other sensor, the IP that it is fully compliant with the

TABLE 3. CCSDS123 performance in terms of bit per pixel per band (bpppb), depending on the selected entropy coder.

Instrument	Sample-adaptive	Block-adaptive (BIL order)	Block-adaptive (BIP order)
AIRS	4.30	4.36	4.39
M3-Target	3.09	3.12	3.12
Hyperion	4.33	4.38	4.47
AVIRIS(raw)	5.98	6.01	6.22
AVIRIS (calibrated)	3.74	3.78	4.04
MODIS-day	5.72	5.31	7.02
Landsat	3.37	3.39	3.47

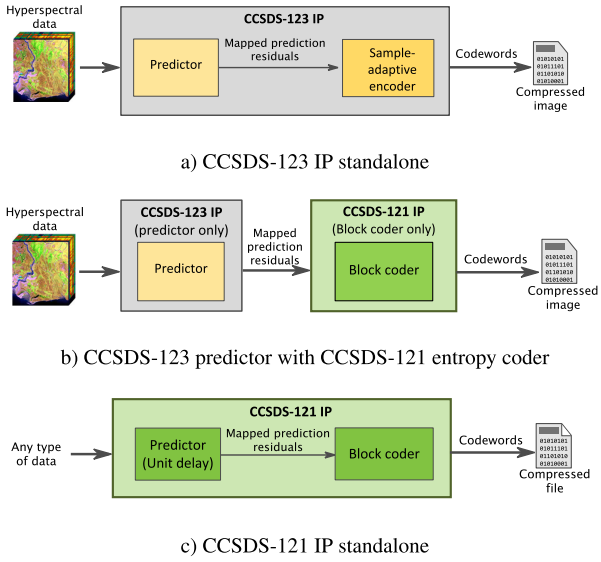


FIGURE 4. General overview of SHyLoC 2.0.

CCSDS 121.0-B-2 standard is implemented for compressing purposes, as shown in Fig. 4c.

Regarding input and output interfaces, both IPs implement a simple handshake protocol (not shown in Fig. 4 for simplicity). The data flow is handled through the *ForceStop* (it forces the stop of the compression) and the *Ready* signals at the input side, being the latter asserted when the IP is configured correctly and it is able to receive new samples for compression. On the output side, the *Ready_Ext* signal can be used by the external module responsible of managing the output data to inform the IP that it is ready to receive the compressed samples. Additionally, there are other signals that are part of the input interface to manage data control, such as *FIFO_full*, *EOP* (indicates that the compression of the last sample has started) or *Finished*, which is asserted when the compression has ended. Besides, both data input and output signals have their associated *Valid* flag.

A. SHyLoC 2.0 CCSDS121-IP

As it was aforementioned in Section II-A, the CCSDS 121.0-B-2 data compression standard defines two main stages, a predictive decorrelation and an entropy coding.

The SHyLoC 1.0 CCSDS-121 IP developed as previous work [14] presents an implementation of the entropy coding stage and includes a configuration engine to enable the setting of parameters allowed by the standard. The *EN_RUNCFG* parameter is provided to enable or disable the setting of configuration parameters at runtime. In case of disabling, the parameters are set at compile-time, reducing the overall complexity of the design. This module is reused in SHyLoC 2.0, denoted here as block-coder, with slight modifications.

The SHyLoC 2.0 CCSDS-121 IP extends the functionality of the previous IP by including an optional unit-delay predictor before the block-coder. The predictor, when present, is set-up by the configuration core of the block-coder itself, thus avoiding the replication of the configuration interface. This is depicted in Fig. 5. The overall functionality of the CCSDS-121 IP is summarised as follows.

First, the CCSDS-121 IP receives the runtime configuration values through the AMBA Advanced High-performance Bus (AHB) slave interface and after that, a valid flag is asserted to inform it is ready to receive new samples. The received configuration is stored in internal registers, read by the IP and then made available for all the modules that request them. The input samples are received through an ad-hoc parallel interface, designed with the purpose of easily connecting the module with an external pre-processing stage when its unit-delay predictor is not included (this is controlled with the parameter *PREPROCESSOR_GEN* different from 2), performing only the encoding step. If the CCSDS-121 predictor is enabled (*PREPROCESSOR_GEN* = 2), the input samples are pre-processed prior to being coded. The inclusion or absence of the unit-delay predictor does not affect the external interfaces but the data path inside the IP, as shown in Fig. 5, making it transparent for the user. If the unit-delay predictor is not implemented, the samples to be encoded may come from an external pre-processing stage, or directly from an Analog-to-Digital Converter (ADC), a SpaceWire interface or a mass memory.

1) UNIT-DELAY PREDICTOR

The unit-delay predictor core consists of a register that holds the value of the previously processed sample, a subtracting module to generate the prediction residuals (i.e. the difference between the current sample and the previous one) and a mapper, used to transform the prediction residuals to unsigned values prior to send them to the entropy coder. The top module of the CCSDS-121 IP unit-delay predictor core, depicted in Fig. 6, includes the necessary logic to bind the components that perform the reception of input samples, the prediction step and the flow control of the output of mapped prediction residuals, as well as periodic reference samples.

Efforts are made to respect the interfaces and configuration method already present in SHyLoC 1.0, so that the new module becomes transparently available for prospective users.

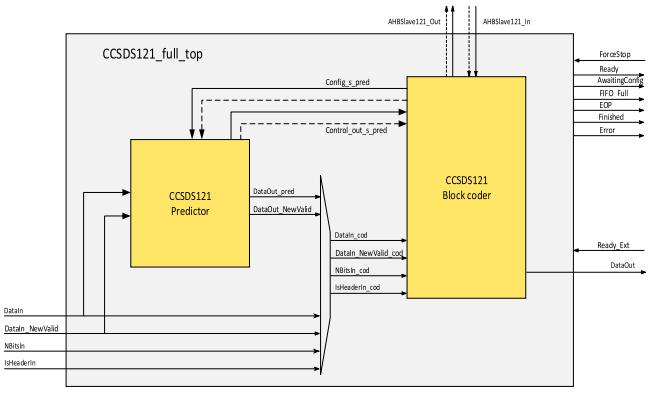


FIGURE 5. Schematic of the SHyLoC CCSDS121-IP.

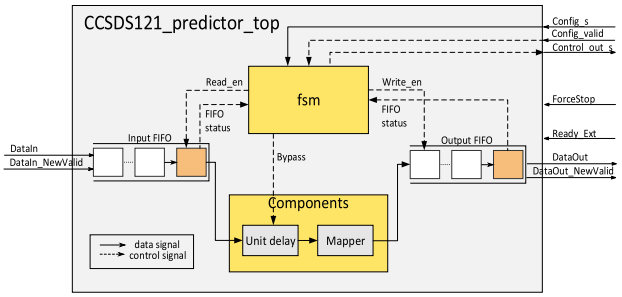


FIGURE 6. Simplified block diagram of the unit-delay predictor top module.

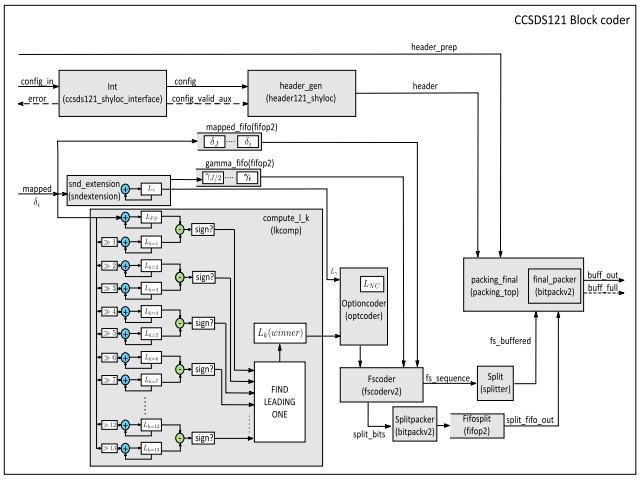


FIGURE 7. Simplified block diagram of the entropy coder core.

In this way, the runtime configuration is received from the block encoder module together with a valid flag, because this component is the one that has access to the AHB slave interface in SHyLoC 1.0. According to the received configuration, a Finite State Machine (FSM) module controls the operation of the rest of the modules in the design. The *components* module performs the preprocessing step itself. It contains the unit-delay predictor which computes the predicted samples based on the input data, and the mapper module.

Moreover, the unit-delay predictor includes input and output FIFOs in order to adapt the data transfers between modules. The input FIFO stores the incoming input data until they are preprocessed, while the output FIFO stores the preprocessed data in groups of size J until the block-adaptive coder requires them to continue with the compression flow.

2) BLOCK-ADAPTIVE ENCODER

On the other hand, the block-adaptive encoder includes, in addition to the encoding functionality, the necessary logic to receive the runtime configuration values through the AHB slave and the input samples, using an ad-hoc dedicated interface. In addition, the flow control of the output compressed bitstream is managed by this module. The block-coder core is shown in Fig.7.

First of all, the *int* module reads the runtime configuration values, after being adapted to the IP clock domain. The *int* module also validates the configuration provided, arising an error if the configuration values are out of range. If runtime configuration is disabled ($EN\_RUNCFG = 0$), the IP uses the configuration values defined during compile time.

Then, the configuration values are sent to the *header_gen* module, which generates the different header fields according to the configuration provided, including prediction fields if the unit-delay predictor is implemented, and transmitting them to the *packing_final* module. This module splits the output bitstream in words with a size specified by the user through the *W_BUFFER* parameter. Every time the output buffer is full with a word with a size of *W_BUFFER* bits, a valid flag indicates that the output value can be captured by an external module.

The rest of the modules constitute the compression engine itself, implementing the block-adaptive entropy coder described in Section II-A. The *snd_extension* module computes the length of a block encoded with the second-extension option. The *compute_l_k* module computes the length of a block encoded with the FS, as well as all the sample splitting options. The number of options to be evaluated depends on the dynamic range of the input samples and the user-selected configuration parameters. Besides, this module identifies if a block contains all zeroes, selecting in this case the zero-block option. The option that generates the codeword with the minimum length is stored in a register named as $L_k(winner)$, and it is compared with the lengths obtained from the second-extension and no compression modules using the *optioncoder* module, which selects the best coding option in terms of minimum size. Once the encoding option is selected, the *fscoder* module encodes the stream according to the FS sequence, joined with the option identifier. Finally, the sequence encoded by the *fscoder*, as well as the uncompressed sample splits, are sent to the final packer to be outputted.

If the unit-delay predictor is included, its *fsm* module may bypass the preprocessing step by activating the *Bypass* signal, in order to periodically insert the reference sample. This step is necessary to recover the original image during the decompression step. The entropy coder must manage these reference samples, and for this reason, it is necessary to apply some modifications in its structure. First of all, it should be able to identify which blocks of samples include a reference sample. This process is done introducing a new state in the block-coder FSM. In addition, the *snd_extension* and the *compute_l_k* modules have been modified in order to compute correctly the length of the compressed block for each compression option when a reference sample is introduced, following the rules detailed below:

- Fundamental sequence. The reference sample is not compressed, so its contribution to the length of the compressed block is the dynamic range of the input samples rather than its value. The rest of the samples in the block are compressed as usual.

- Sample splitting. Same case as the FS; the reference sample is uncompressed and no changes are presented in the management of the rest of the block samples.
- Second-extension. The reference sample is independently coded without compression. Then, the first sample of the block is replaced by a zero value when computing the gamma values.
- Zero-block. The reference sample is not taken into account when the zero-block condition is evaluated. This is, a block of all zeroes except for the first sample will be coded with this option if it includes a reference sample.
- No compression. No changes are presented with respect to version 1.0.

In all the aforementioned options, the reference sample is coded after the unique identifier for each coding option. In order to do so, the block-coder FSM has been properly modified.

In addition to the implementation of the unit-delay predictor, the SHyLoC 2.0 CCSDS-121 IP includes other new features, such as the management of input samples with a dynamic range up to 32 bits, instead of the maximum 16 bits supported by version 1.0. Endianness handling is the main change to apply in order to extend the dynamic range. Finally, signed samples are supported too. These two features are only supported by the CCSDS-121 IP if the unit-delay predictor is included.

Both the new configuration parameters introduced for the SHyLoC 2.0 CCSDS-121 IP and the modified ones are summarized in Table 4.

B. SHyLoC 2.0 CCSDS123-IP

The SHyLoC 2.0 CCSDS-123 IP includes both the predictor and the sample-adaptive entropy coder which are defined in the CCSDS 123.0-B-1 compression standard. As for the CCSDS-121 IP, constants and runtime configuration parameters are detailed in [14], including the local sum calculation and prediction modes, or the number of previous bands P used by the prediction stage, among others. The option of implementing EDAC to the internal embedded memories is offered, relevant to provide robustness in critical environments. This IP can work either independently or jointly with the CCSDS-121 IP counterpart, which implements the block-coder of the CCSDS 121.0-B-2 standard (*ENCODING_TYPE* $= 0$ and *ENCODER_SELECTION_GEN* $= 2$).

Additionally to the modules performing the compression operations, i.e. the predictor and sample-adaptive encoder blocks, the CCSDS-123 IP implements additional components which take care of control, configuration and interface management. Standard interfaces are provided to ease the interaction with other co-processors. In particular, the CCSDS-123 IP includes two AHB interfaces: one acts as a configuration port (slave interface), while the other (master interface) is used to communicate with an external memory to store intermediate values during the compression, which is exploited by certain compressor architectures to reduce the embedded block RAM (BRAM) usage. However, the AHB master interface in SHyLoC 1.0 does not have burst transfers capabilities, which limits the compressor performance when the external memory is used, forcing to use only single transfers. The ad-hoc data input and output interfaces are complemented with input and output control signals. The IP also includes a configuration block, which receives the user-defined configuration parameters from the AHB slave interface, validates it and disseminates it to the rest of modules, adapting clock frequencies between external and internal domains and generating the compressor header according to the CCSDS standard. Finally, a dispatcher module collects and packs all the output data [21].

TABLE 4. CCSDS121 IP-new and modified parameters.

Constant	Allowed values	Description
PREPROCESSOR_GEN	0	no preprocessor presented
	1	CCSDS123 preprocessor
	2	**CCSDS121 preprocessor**
	3	another preprocessor used
D_GEN	[2:**32**]	maximum dynamic range of the input samples
IS_SIGNED_GEN	[0,1]	(0) unsigned; (1) signed

1) PREDICTION STAGE

SHyLoC 1.0 was originally designed focusing on versatility, and because of that, all data arrangements in multispectral and hyperspectral applications are supported: BIP, BIL and BSQ. A specific architecture of the CCSDS-123 IP predictor has been designed for each ordering with the aim of optimizing computations, taking into account the data dependencies present on each case. The use of dedicated predictor architectures for each image order allows a better coupling with the targeted sensor, processing the samples on-the-fly without any previous extra reordering step. Memory requirements and data dependencies for each predictor architecture are also analysed in [14]. In addition, for the BIP ordering two different architectures are offered: with and without access to an external memory. The use of an external memory to store intermediate results allows to manage images with bigger size both in the spatial and the spectral domain and even with higher bit resolution. Therefore, architectures that use an external memory are suitable for reducing the IP resources utilization when the memory demand is too high, at the cost of reduced throughput.

In summary, a total of 4 predictor architectures were devised in SHyLoC 1.0, named BIP, BIP-MEM (BIP with external memory), BIL and BSQ. The architecture must be selected at implementation time, determining which predictor design is implemented, defined by the *PREDICTION_TYPE* constant.

Nevertheless, all these predictor architectures share the same internal structure, represented in Fig. 8, which is then customized for each scenario.

A set of FIFOs allows arranging the input samples in such a way that, for every sample, the neighbour samples used in the prediction are available at the output of each FIFO.

This means that the size of the FIFOs may vary depending on the compressor configuration, including the maximum allowed input image dimensions. Then, the predictor estimates the current sample, maps the prediction residual and updates the weights, all according to the CCSDS 123.0-B-1 compression standard. In order to save resources, the local differences are stored to be reused in subsequent bands. Additional memory is required for the Band-Interleaved architectures to store and retrieve the weight vectors when changing between bands. SHyLoC 1.0 only supports the default weight initialization defined in the CCSDS 123.0-B-1 standard, because of the high complexity of implementing the custom mode in comparison with the limited improvement in the compression ratio. Finally, there are storage elements which are placed in external memories depending on the selected architecture: in the BIP-MEM architecture, the input FIFO corresponding to the top right neighbour is moved outside; furthermore, the local differences are stored in an external memory when the samples arrangement is BSQ.

Assuming that the spectral dimension of the hyperspectral image to be compressed is large enough to fill a pipeline (dependent on the hypercube size and the number of bands P used for prediction), the BIP architecture is the one that achieves a throughput of one sample per clock cycle, taking advantage of performing both the dot product needed by the predictor and the weights update operations concurrently. Although the BIP-MEM architecture works internally in the same way than the BIP one, the communication with the external memory used to store and retrieve intermediate values during the compression introduces a slight performance penalty. The throughput of the BIP-MEM architecture can be improved taking advantage of the burst transfer capabilities offered by the AMBA AHB interface [22], a feature introduced in SHyLoC 2.0.

On the other hand, the BSQ ordering presents strong data dependencies between adjacent input samples, preventing parallelism among operations and thus, imposing a noticeable throughput limitation. Because of this, this architecture works in a serial manner, in order to reduce resources utilization.

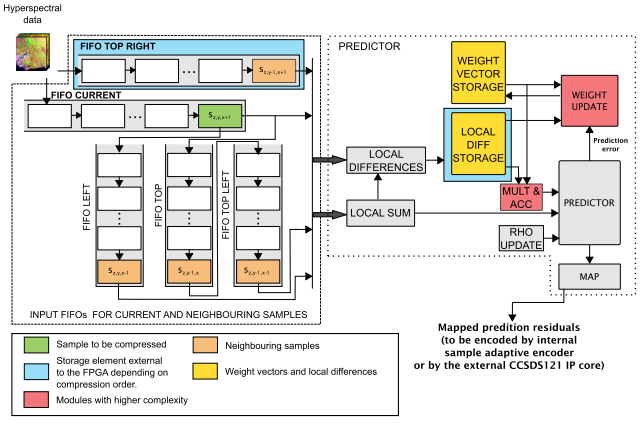


FIGURE 8. CCSDS-123 IP - Predictor block diagram [21].

Additionally, in this architecture it is required to store a complete vector of local differences per sample. As the memory demand can be too high, the storage of these elements is moved to an external memory, accessed through the AHB master interface [14].

Finally, the BIL architecture is a combination of the BIP and BSQ architectures. It takes most of the components of the BIP architecture, but uses a dedicated set of FIFOs in order to store the local differences internally. Regarding data dependencies, there are two possible situations, depending on whether the next sample is in the same band or not. Therefore, a specific operation scheduling was designed for the BIL architecture, taking into account this duality.

2) NEW FEATURES INTRODUCED TO THE PREDICTOR

Although SHyLoC 1.0 offers a efficient compression solution for hyperspectral images in terms of compression ratio and resources utilization, its throughput shall be improved, specially for the architectures that use an external memory to store intermediate results. In addition, the CCSDS-123 IP does not implement some of the optional features present in the compression standard, which can be of interest to further increase the compression performance in terms of data reduction in some applications.

SHyLoC 2.0 includes design optimizations for improving the throughput, as well as standard-compliant features which were not included in SHyLoC 1.0. The use of burst transfers through the AHB interface to increase the memory bandwidth, and the design of a new architecture to process samples in BIL ordering with the aid of an external memory, named as BIL-MEM, are considered to efficiently enhance compression performance in terms of latency.

In addition, we have studied the viability of implementing the custom weights initialization, analyzing its impact in the current predictor architectures. It is not a functionality completely included in SHyLoC 2.0, though parameters to do it in an efficient way have been defined.

a: BURST TRANSFERS AND BIL-MEM ARCHITECTURE

In the BIP-MEM architecture, the input samples are written in the external memory by the AHB master interface of the CCSDS-123 IP, which controls when the intermediate samples are read and written, properly setting the address and control signals of the transaction and avoiding any data losses. Asynchronous FIFOs are placed between the predictor and the AHB master, in order to adapt data rates between both modules. During the processing, stores and reads are one spectral line apart, i.e. only after all samples in a line (with all its bands) are stored, they are required to be read. From that point onward, reads and stores are required, until the last spectral line, where only read operations are performed.

SHyLoC 1.0 supports only single transfers, forcing the AHB master to request the control of the bus on every single sample, which may impact the overall performance. However, in SHyLoC 2.0, the AHB master interface has been redesigned to support incremental burst transactions with a

maximum of 16 beats per burst, with a scheme where read and write bursts are interleaved. In this way, the potential delays because of the control requests to the bus arbiter are noticeably reduced, as they only take place once per burst, and they can be masked if the bus control is not requested by any other peripheral. The maximum size of the burst transfers is established at compile time using the *HMAXBURST* constant, but the AHB master is able to manage shorter bursts to meet certain checkpoints along the execution (e.g. when the end of the image is reached).

In addition, a new architecture has been implemented, which processes input samples in BIL arrangement, but uses the AHB master interface to store the contents of the top right FIFO in an external memory (similarly as in the BIP-MEM architecture). Hence, this new architecture, denoted as BIL-MEM, allows to process images in BIL ordering even if the target device does not have enough memory resources for the demands of the target configuration. The same optimizations of the AHB master interface from the BIP-MEM architecture are applied here, including the communication with the external memory in burst mode.

b: CUSTOM WEIGHT INITIALIZATION

The CCSDS 123.B-0-1 compression standard optionally allows to use a set of custom weight vectors instead of the default initialization values, even band-independent vectors for a fine grain adjustment. According to [20], a compression data rate reduction up to 4% can be achieved with the custom weight initialization if their values are properly tuned. This motivates the analysis of the weights initialization mode, revealing the trade-offs to be considered for its implementation. In particular, additional storage resources may need to be allocated in order to support the configuration of the custom initial weights, the configuration of the IP may be longer and, according to [10], custom weight values have to be coded in the header of the compressed image, which puts into question the benefits in terms of compression efficiency.

Being the SHyLoC CCSDS-123 IP designed with the goal of full compliancy with the CCSDS 123.B-0-1 compression standard, the implementation of the custom weight initialization mode must be taken into account, considering the implementation complexity and the identification of new configuration options. At the same time, its implementation must be transparent to the user, preserving all the modules and architectures developed for SHyLoC. Reusing already existing components would be desirable, thus reducing the resource utilization penalties.

Taking all these factors into account, the proposed solution consists in preloading the weight FIFOs with custom initial values at the beginning of the compression process. If the custom weight initialization mode is used (*WEIGHT_INIT_GEN* $= 1$), these values would be received by the AHB configuration port and loaded into the weight FIFOs by means of a dedicated logic. In the case of the BSQ architecture there is no internal memory storage for weight vectors, as there is no need to maintain more than one weight vector at any given time. Therefore, it would be required to implement an extra memory to store the custom initial weights, and load them one by one with each new band. The size of this memory would depend on the compressor configuration, with a maximum of $N_z \cdot (P + 3)$ elements.

The specification of custom weight vectors must be supported at runtime through the AHB configuration interface. Two configuration modes are foreseen: different weight vectors for each band (fine-grain adjustment, high configuration penalty) and the same vector for all the bands (coarse-grain adjustment, low configuration penalty), selected through a new parameter *CWI_GEN*. In order to reduce the configuration time penalty, implementing an extra memory is suggested to store the weight vectors received from the configuration port (which is in any case mandatory for the BSQ architecture), so they can be reused in consecutive runs. Alternatively, this memory could be preloaded, allowing to be configured at implementation time. This feature could help in reducing the design complexity when the nature of the images to be processed has little variability and it is known beforehand.

The coding of the custom initial weights in the header is done at the same time they are loaded into the weight FIFOs, for a minimum performance penalty. The exception is the BSQ architecture, where custom weights are progressively loaded along the compression process. In order to reduce the compression efficiency penalties, two non-standard solutions are proposed: if the same weight vector is used for all bands, then coding a single vector in header would be enough, signaled with appropriate flags. Alternatively, there is the possibility of not including them in the header, although in that case the decompressor must know a priori the initial weight values.

Finally, in the Band-Interleaved architectures it is possible to reuse the final state of weight vectors as custom weights for the next compression, reducing the communication needs in case of image partitioning along the Y axis. This feature, enabled with $WR = 0$, would require a previous compression run with any of the standard initialization modes. This mode would not be available for the BSQ architecture.

The new parameters defined for the predictor stage of the SHyLoC 2.0 CCSDS-123 IP, as well as the ones modified regarding its first version, are shown in Table 5.

3) SAMPLE-ADAPTIVE ENCODER

The sample-adaptive encoder is formed by two main modules, the compression engine and the bit packer, together with a FSM that manages the complete coding process, as shown in Fig. 9. As soon as the mapped residuals are received from the predictor, the compression engine calculates the code for each input sample using the *createcdw* module, taking into account the counter and accumulator values. Then, these image statistics are updated in the *update_counters* module. For that purpose, two FIFOs are needed when some of the available Band-Interleaved architectures are selected for the prediction stage, in order to store the accumulator

TABLE 5. CCSDS123 IP-new and modified parameters.

Constant	Allowed values	Description
PREDICTION_TYPE	0	BIP architecture
	1	BIP-MEM architecture
	2	BSQ architecture
	3	BIL architecture
	4	**BIL-MEM architecture**
CWI_GEN	[0,1]	(0) different weight vectors for each band (1) same weight vector for all the bands
WR	[0,1]	(0) reusing weight vectors for the next compression (1) weight vectors are reset after each compression
HMAXBURST	[1:16]	AHB master burst beat limit

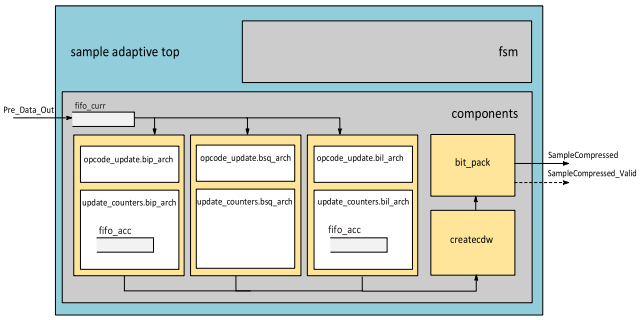


FIGURE 9. CCSDS-123 IP - Sample-adaptive block diagram.

values of a specific sample with all its spectral components (i.e. in all the bands), necessary to compress the next one. The last part of the encoding flow is the *bit_pack* module, which packs the generated codewords according to the value of the *W_BUFFER_GEN* constant. Every time the output buffer is full, a valid flag is asserted to indicate that the output bitstream can be captured.

IV. EXPERIMENTAL RESULTS

This section summarizes the verification stage and the synthesis results obtained for both SHyLoC 2.0 IPs on different space-qualified FPGA technologies, detailing also the selected parameters for each test. Concretely, the target devices under study in this work are the Xilinx Virtex5QR XQR5VFX130, the Microsemi RTG4 150 and the NanoXplore devices, NG-MEDIUM and NG-LARGE. More comprehensive results for other FPGA technologies (including antifuse) and ASICs are included in the SHyLoC datasheet [13], [21].

For the BRAVE FPGA family, the NG-LARGE device is selected as implementation target for the CCSDS123-IP, since it provides enough resources to map the IP in most configurations, including the most complex ones, that can be too demanding for smaller devices such as the NG-MEDIUM.

The synthesis results for both IPs have been obtained using Synopsys Synplify Premier N-2018.03 for the different FPGA technologies under study, applying pipelining optimization synthesis attributes in order to obtain better results in terms of timing. Similarly, the NanoXplore NXmap software suite (version 2.9.2) was used to obtain mapping results for the BRAVE family, setting the synthesis options *TimingDriven* and *UseSynthesisRetiming* to *Yes* and the option *MappingEffort* to *Medium*, in order to achieve the maximum clock frequency possible without compromising the logic resources utilization.

A. SHyLoC 2.0 CCSDS121-IP

The SHyLoC 2.0 CCSDS121-IP has been widely verified by simulation prior to obtaining synthesis results. The verification methodology is based in comparing the output of the IP for each one of the tests performed with the golden reference compressed streams generated using the CCSDS-121 reference software, provided by ESA [23]. Concretely, a total of 85 tests were executed successfully, including basic sanity tests to verify the main functionality and corner cases. A total of 24 different configuration sets (10 in compile time and 14 in runtime) and 29 images with different number of samples, samples distribution and dynamic range were used. With this large testbench, all the possible situations the compressor can face during its functional lifetime are taken into account. The baseline configuration used for mapping purposes is summarized in Table 6. The parameters that change among configurations are *PREPROCESSOR_GEN*, which specifies if the unit-delay predictor is included or not; and the dynamic range, using 16 and 32 for analysing the impact of the bit depth extension. As it is shown, we have fixed the maximum cube size of the IP to 1024 for each one of the coordinates. The same constraint in terms of data size is applied to the CCSDS-123 IP. This constraint is specified for synthesis purposes. The maximum cube size supported by both IPs is 65535 for each coordinate, restricted by the precision used (16 bits).

Synthesis results for the SHyLoC 2.0 CCSDS-121 IP on Virtex5QR XQR5VFX130, RTG4 150 and NG-MEDIUM are shown in Table 7, Table 8, and Table 9, respectively. Results for BRAVE technology are shown on the NG-MEDIUM device because the CCSDS-121 IP fits well in the available resources. Each table includes the resources

TABLE 6. CCSDS121 IP-baseline configuration for synthesis results.

Generic	Value	Description
EN_RUNCFG	1	enables runtime configuration
PREPROCESSOR_GEN	0,2	(0) only block-coder; (2) 121-IP predictor added
Nx_GEN	1024	max. number of columns
Ny_GEN	1024	max. number of rows
Nz_GEN	1024	max. number of bands
D_GEN	16,32	input samples bit depth
ENDIANNESS_GEN	0	little endian
IS_SIGNED_GEN	0	unsigned samples
DISABLE_HEADER_GEN	0	header is always generated
J_GEN	16	block size
CODESET_GEN	0	basic code option
REF_SAMPLE_GEN	256	reference sample interval
W_BUFFER_GEN	32	bits in the output buffer

utilization in terms of memory blocks, arithmetical units and logic resources, as well as the estimated maximum throughput. As the SHyLoC 2.0 CCSDS121-IP is able to process one sample per clock cycle, the maximum throughput will be equal to the maximum clock frequency achieved.

The results show that best outcome in terms of throughput is obtained for the Virtex5QR XQR5VFX130, achieving a maximum value of 106.3 MSamples/s when the IP works only as block-adaptive encoder and limiting the maximum dynamic range to 16.

However, it must be taken into account that Virtex5QR architecture uses larger logic elements than the rest of the technologies analysed, integrating 6-input LUTs while RTG4 and NG-MEDIUM use 4-input LUTs. The NG-MEDIUM performance is also limited by the novel NanoXmap synthesis tool, which is evolving with each new version. In general, best throughput results are obtained for all the technologies analysed when the unit-delay predictor is not implemented as part of the design and consequently the dynamic range is up to 16 bits, as it was expected. A minimum penalty is observed in timing results when the unit-delay predictor is introduced, going from the 2% for Virtex5QR to the 16% for RTG4 150. For all the technologies analysed, the extension of the dynamic range up to 32 bits when the predictor is implemented implies a significant impact in timing performance, reducing throughput results an average of 20%.

In terms of resources utilization, the maximum logic usage for Virtex5QR is approximately the 9% of the available LUTs, without using any embedded memory block. These results are obtained when the predictor is included and the dynamic range is set to 32 bits, the most critical configuration in terms of resources consumption. This maximum area occupation is repeated for Microsemi RTG4 150, using in this case the 7% of memory resources.

This situation is probably due to the fact that in Virtex5QR the synthesis tool exploits the possibility of using LUTs as distributed memory when not so much storage is needed, while for RTG4 the tool tries to achieve a trade-off between the use of BRAMs and logic resources.

Occupation results for NG-MEDIUM are in the line with the rest of FPGA technologies analysed. A maximum usage of the 32% of LUTs and the 36% of Block RAMs (BRAMs) is shown again when predictor is included and the dynamic range is set to 32 bits. In this case, it is remarkable the high usage of Carry Cells (a maximum of 70%) due to the extended resolution in the arithmetic operations.

As it can be observed, the bit depth extension and the inclusion of the unit-delay predictor imply an increment of the logic resources utilization for the different FPGA technologies analysed. In addition, though for the included results the block size generic *J_GEN* was fixed to 16, the increment of this parameter also supposes a boost of the logic resources utilization, as it is shown in [21].

TABLE 7. CCSDS121 IP- Synthesis results on Xilinx Virtex5QR XQR5VFX130.

	With predictor D=16		With predictor D=32		Without predictor D=16	
36Kb BRAM	0	0%	0	0%	0	0%
DSP48E	4	1%	5	1%	3	1%
Registers	1563	2%	2291	3%	1505	2%
LUTs	4381	5%	7670	9%	3351	4%
Throughput (MSamples/s)	104.0		79.9		106.3	

TABLE 8. CCSDS121 IP- Synthesis results on Microsemi RTG4 150.

	With predictor D=16		With predictor D=32		Without predictor D=16	
RAM64x18	11	5%	19	9%	11	5%
DSP	5	1%	5	1%	4	1%
Carry cells	1290	1%	2865	2%	962	1%
Sequential cells	1342	1%	1918	1%	1319	1%
LUTs	6648	4%	11055	7%	5365	3%
Throughput (MSamples/s)	52.5		46.0		62.2	

TABLE 9. CCSDS121 IP- Synthesis results on NanoXplore NG-MEDIUM.

	With predictor D=16		With predictor D=32		Without predictor D=16	
48Kb BRAM	13	23%	20	36%	11	20%
DSP	2	2%	2	2%	2	2%
Carry cells	2297	28%	5660	70%	1925	24%
Registers	1550	5%	2670	8%	1463	5%
LUTs	5916	18%	10309	32%	4747	15%
Throughput (MSamples/s)	25.8		21.9		28.1	

B. SHyLoC 2.0 CCSDS123-IP

The SHyLoC 2.0 CCSDS-123 IP has been also verified by means of an extensive validation campaign, in the same way than the CCSDS-121 IP. A corpus of 18 multispectral and hyperspectral images have been tested, including both synthetic images to verify corner cases and images from sensors used in the space and aircraft industries, such as the Atmospheric Infrared Sounder (AIRS), the Airbone Visible/Infrared Imaging Spectrometer (AVIRIS) or the Moderate Resolution Imaging Spectroradiometer (MODIS), whose features were mentioned in Table 1. The IP has been tested using up to 22 different sets of configuration parameters (11 in compile-time and 11 in runtime), trying to cover a broad range of cases and verifying all the predictor architectures. Table 10 shows the values used during the synthesis step for some relevant configuration parameters, excluding image dimensions which are image-dependent. A total of 110 different tests were passed, whose results have been compared against golden references generated by a reference software implementing the CCSDS 123.0-B-1 compression standard, provided by ESA [23].

1) TIMING RESULTS

Results for the SHyLoC 2.0 CCSDS-123 IP on Virtex5QR XQR5VFX130, RTG4 150 and NG-LARGE are shown in Table 11, Table 12, and Table 13, respectively. As it was

TABLE 10. CCSDS123 IP-baseline configuration for synthesis results.

Generic	Value	Description
EN_RUNCFG	0,1	(0) compile time; (1) runtime
PREDICTION_TYPE	[0:4]	(0) BIP architecture; (1) BIP-MEM architecture; (2) BSQ architecture; (3) BIL architecture; (4) BIL-MEM architecture
ENCODING_TYPE	1	sample-adaptive encoder
D_GEN	[8,12,14,16]	samples bit depth
ENDIANNESS_GEN	0,1	(0) little; (1) big
IS_SIGNED_GEN	0,1	(0) unsigned; (1) signed
DISABLE_HEADER_GEN	0	header is always generated
W_BUFFER_GEN	32	bits in the output buffer
P_MAX	3	bands used for prediction
PREDICTION_GEN	0	full prediction
LOCAL_SUM_GEN	0	neighbour-oriented mode
OMEGA_GEN	13	weight resolution
HMAXBURST	[1,4,8,16]	burst transfers size

TABLE 11. CCSDS123 IP- Maximum frequency on Xilinx Virtex5QR XQR5VFX130.

Config	f_clk (MHz)	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	AHB	-	162.3	153.5	-	122.7
	System	150.1	153.3	132.7	150.4	113.7
AVIRIS	AHB	-	165.4	131.0	-	111.2
	System	138.3	138.3	112.9	138.1	105.7
AIRS	AHB	-	157.6	125.8	-	115.0
	System	140.4	139.2	110.9	136.9	112.7
Runtime config.	AHB	237.5	157.6	130.9	223.3	120.5
	System	130.6	140.8	125.6	131.0	105.4

TABLE 12. CCSDS123 IP- Maximum frequency on Microsemi RTG4 150.

Config	f_clk (MHz)	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	AHB	-	101.5	81.7	-	72.0
	System	85.3	83.6	80.4	85.1	69.6
AVIRIS	AHB	-	101.2	85.2	-	76.3
	System	80.9	80.5	80.8	77.1	76.2
AIRS	AHB	-	101.3	79.5	-	82.5
	System	81.2	81.2	79.5	84.3	76.6
Runtime config.	AHB	152.1	99.0	82.7	152.1	76.3
	System	80.5	80.5	81.4	77.1	74.2

aforementioned, results for BRAVE technology are shown in this case on the NG-LARGE device, because NG-MEDIUM does not have enough resources to fit Band-Interleaved architectures where only internal memory resources are used. The group of the images presented on Table 1 has been considered for the synthesis step, including multispectral (Landsat, 6 bands), hyperspectral (AVIRIS, 224 bands) and ultraspectral (AIRS, 1501 bands) images. Additionally, the AVIRIS sensor was also used to obtain results when runtime configuration is enabled.

The system clock frequency achieved for Virtex-5QR is above 105 MHz for all the configurations implemented (see Table 11). For the Landsat configuration, which takes the least resources, some architectures can go even faster, achieving up to 153.3 MHz for the BIP-MEM architecture. However, not all the architectures provide the same

TABLE 13. CCSDS123 IP- Maximum frequency on NanoXplore NG-LARGE.

Config	f_clk (MHz)	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	AHB	-	33.0	43.9	-	34.0
	System	44.3	34.8	41.4	42.0	35.7
AVIRIS	AHB	-	35.8	40.2	-	26.4
	System	41.9	35.9	43.8	32.0	34.0
AIRS	AHB	-	28.9	42.2	-	26.4
	System	34.8	36.0	40.7	31.9	37.9
Runtime config.	AHB	108.5	29.7	38.3	80.1	29.7
	System	35.0	32.9	37.9	28.7	30.8

throughput, being BIP the one that can reach the maximum throughput of one sample per clock cycle, provided enough bands in the input image to be compressed. Besides, the CCSDS-123 IP has been designed in such a way that the AHB clock is faster than the system clock in order to avoid delays in the processing due to the communications with the external memory, a condition that is fulfilled in all the implemented architectures.

For the Microsemi RTG4 technology, the system clock frequencies range from 69 to 85 MHz approximately, with some of the Landsat configurations going even slightly faster. Finally, clock frequencies between 28 and 45 MHz are obtained for the NG-LARGE, and in this case the AHB clock frequency can be slightly slower than the system clock in certain configurations.

The throughput of the different compressor architectures has been measured by means of simulation using the Mentor QuestaSim software. In order to provide a fair comparison, the same test image (in different arrangements) has been compressed with all the compressor architectures, using always the same set of configuration parameters (with $P = 3$). In addition, for the BIP-MEM and BIL-MEM architectures, different values of the *HMAXBURST* parameter have been tested, in order to study the impact of the burst length as well as the presence or absence of bursts in the compressor performance. The AHB data and address interfaces were configured with a width of 32 bits. The bus clock is always twice the IP clock frequency for the three cases analysed in Table 14, in order to avoid a bottleneck on the data transfers. Finally, coupling FIFOs placed between the IP core and the AHB interface have a width equal to D and a depth of 16, the maximum burst size allowed. These FIFOs include a control flow protocol, including empty/full flags.

The image used for these measurements was a small image from the CCSDS dataset, with dimensions of $75 \times 78 \times 80$ pixels. The results are summarized in Table 14.

According to these results, the performance for the BIP-MEM architecture improves slightly more than 1.5 times with respect to using just single transfers when bursts of size 4 are used, and by a factor of almost 2.2 with bursts of size 16. If we compare against the BIP-base architecture, the fastest one, the performance penalty incurred by communicating with the external memory is 15% with bursts of size 16. This is a significant improvement to SHyLoC 1.0,

TABLE 14. CCSDS123 IP- Throughput in Msamples/s of the compressor architectures for different burst sizes and clock frequencies.

Architecture	Burst size	Throughput (MSamples/s)		
		@50 MHz	@100 MHz	@130 MHz
BIP-MEM	1	16.93	33.86	44.01
	4	27.00	54.00	70.20
	8	35.06	70.13	91.17
	16	41.22	82.44	107.17
BIP-base	-	50.00	99.99	129.99
BIL-MEM	[1:16]	5.62	11.24	14.61
BIL-base	-	5.62	11.24	14.61
BSQ	-	5.62	11.11	14.44

where the penalty was 60% for BIP-MEM, since it did not support burst transfers. These results confirm the benefits of implementing burst transactions in order to speed up the compressor performance when the BIP-MEM architecture is used.

The same tests have been performed for the BIL-MEM architecture, although in this case all the configurations reported exactly the same execution times, roughly an 87% slower than BIP-base, also identical to the BIL-base architecture.

This means that in the BIL-MEM architecture the processing bottleneck is not related to the communications with the external memory, but with the data dependencies present in the prediction computation (i.e. the previous sample in the same band). Therefore, for this architecture it is preferable to use the memory controller without support for burst transfers already present in SHyLoC 1.0, as it is simpler and lighter. Finally, the BSQ is the slowest architecture in terms of throughput, due to its pure sequential scheduling, and about a 89% slower than BIP-base. This is consistent with the idea of designing a BSQ architecture optimized for low-complexity, in order to balance the performance penalty introduced by its limited throughput [24].

2) RESOURCES UTILIZATION

Synthesis results in terms of resources utilization are summarized in Figs. 10, 11 and 12 for the different target FPGA technologies. In general, the compressor makes a low utilization of logic resources in all the FPGAs under study, which proves the low complexity of the compression algorithm. The usage of DSP units, LUTs, registers and carry cells is almost constant for each target FPGA technology, with slight differences depending on the implemented compressor architecture, the target image size or if the run-time configuration is enabled or not. However, these differences are noticeable in the case of memory usage, which is mainly determined by the predictor architecture. For those architectures without external storage (BIP and BIL), the image size has also a great impact on memory resources.

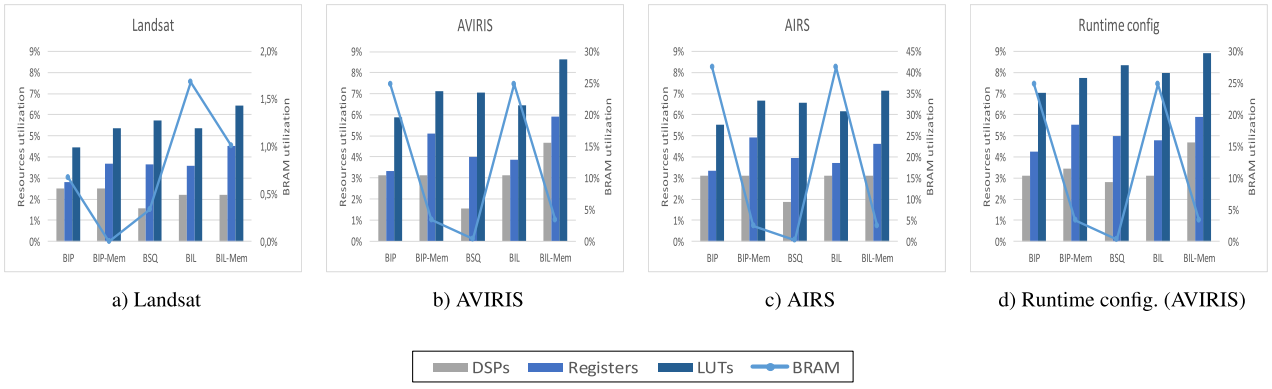
The parameter that has a higher impact in the resources utilization is the the size of an spectral line (N_xN_z). Concretely, this feature determines the use of BRAMs because it fixes the size of different memory elements, such as the FIFOs that store the adjacent samples for the local sum and differences calculation during the prediction, or the dimension of the accumulator array used if the sample-adaptive encoder is selected. The number of P previous bands used for prediction also has an influence in the BRAMs consumption, since it specifies the weights vector size and the number of elements to be considered during the local differences calculation.

Regarding the logic resources consumption (i.e. LUTs and FFs), the dynamic range D supposes the main constraint, since it defines the bit width of different internal elements, such as the local differences values. In addition, the weight resolution Ω has also a slight influence in the LUTs consumption, because it specifies the precision of each element of the weights vector.

For Xilinx Virtex-5QR (Fig. 10), up to 40% of memory resources are used with the largest image in the implemented configurations, so the resource utilization is not a problem for any of the configurations considered. With respect to the DSP units and LUTs utilization, all the architectures and configurations provide similar results, with ranges between 1.5% and 4.7% for DSPs, and between 4.5% and 9% for LUTs. However, the DSP utilization tends to be lower for the BSQ architecture, due to its serial implementation, and higher for the BIP-MEM and BIL-MEM ones, due to the parallelized implementation of some operations and the AHB interface with the external memory. In addition, the BIP and BIL architectures use less LUTs compared with the others, due to the absence of the AHB interface to communicate with the external memory, and the LUT utilization slightly increases with the image dimensions. Comparing both synthesized configurations for the AVIRIS sensor, it can be observed that the inclusion of the AHB configuration interface increases the LUT utilization in only about 1%.

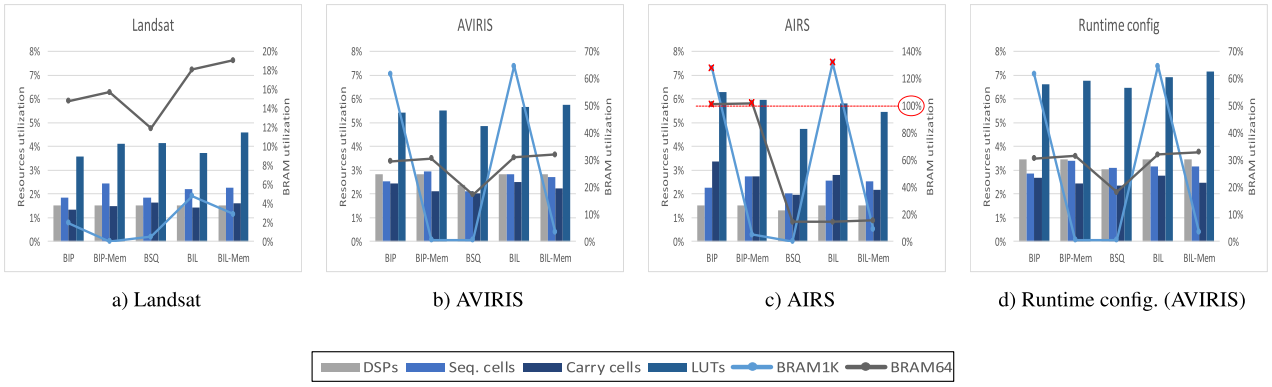
For Microsemi RTG4 (Fig. 11), the memory demands on certain configurations with the largest image are so high that they exceed the FPGA resources, forcing to use an external memory. However, for the BIP-MEM and BIL architectures a better balance in the utilization of both kinds of memory could solve the problem, as in these cases just one of the internal memory resources is overused, while utilization of the other remains low. Regarding the rest of logic resources, they are kept almost constant for each one of the target configurations (between 1.3% and 3.5% for carry cells and DSPs, 1.8% - 3.4% for sequential cells, and 3.5% - 7.2% for LUTs). The implementation of the runtime configuration port increases the reource usage of all these elements in about 1%.

Finally, in the case of NG-LARGE (Fig. 12), the memory usage grows up to a maximum of 80% for the AIRS image, reason why this implementation does not fit on the NG-MEDIUM device. Regarding logic resources, LUTs consumption is in the range of 2.5% - 6%, while DSPs are under the 2% for all the cases. With these results, we demonstrate that this novel FPGA technology is suitable to execute complex designs, such as the targeted hyperspectral image compressor.



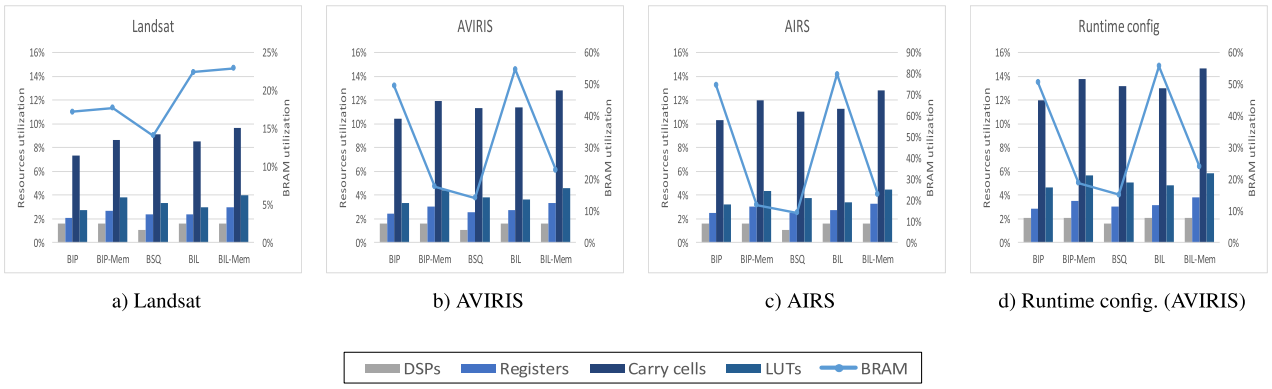
a) Landsat b) AVIRIS c) AIRS d) Runtime config. (AVIRIS)

FIGURE 10. CCSDS123 IP- Synthesis results on Xilinx Virtex5QR XQR5VFX130.



a) Landsat b) AVIRIS c) AIRS d) Runtime config. (AVIRIS)

FIGURE 11. CCSDS123 IP- Synthesis results on Microsemi RTG4 150.



a) Landsat b) AVIRIS c) AIRS d) Runtime config. (AVIRIS)

FIGURE 12. CCSDS123 IP- Synthesis results on NanoXplore NG-LARGE.

C. COMPARISON WITH STATE-OF-THE-ART IMPLEMENTATIONS

The comparison of the SHyLoC 2.0 CCSDS-123 IP with different FPGA implementations available in the state-of-the-art is not easy, because SHyLoC 2.0 allows a high quantity of architectures and configurations that results in different performance capabilities. This flexibility to adapt the IPs to the target application is not generally present in other implementations that focus the attention in obtaining the best architecture generally in terms of either maximum throughput or minimum resources utilization. Moreover, the different FPGA implementations of the CCSDS 123-0-B-1 standard do not target the same technologies, using even in some cases commercial devices as target implementation. It is also

possible to find some implementations of this algorithm on Graphics Processing Units (GPUs) [25], [26]. GPUs can achieve high throughputs in applications that are inherently data-intensive. However, their higher consumption and lack of tolerance to ionising radiation makes them unfit to be used in space. A direct comparison with a GPU implementation is not possible and could be misleading, and therefore we think it is not appropriate to incorporate them to the comparison presented here.

In the case of SHyLoC 2.0, both IPs are technology-independent and consequently they have been mapped for a high variety of FPGA technologies (including the new European BRAVE family) and even for ASICs.

Despite this fact, we consider the architecture of SHyLoC 2.0 CCSDS123-IP that achieves the best throughput for the AVIRIS sensor ($D = 16$), in order to provide a fair comparison with the existing FPGA implementations [14], [24], [27]–[33]. Besides, resources utilization (if available) is analyzed for the mentioned implementations. This comparison is summarized in Table 15.

The implementation proposed by Santos *et al.* [24] focuses on obtaining a low-complexity architecture in terms of resources utilization, in order to fit in older generations of space-grade FPGA technologies. BSQ was selected as processing order, compromising the throughput due to data dependencies in the prediction stage. In addition, local differences are recomputed when needed to optimize area, requiring non-sequential accesses to the external memory and decreasing timing performance. In [27], the authors proposed a BIP architecture for pushbroom instruments, fitting well in different Xilinx FPGA technologies. Nevertheless, the possibilities of parallelism that the algorithm allows are not exploited, incurring in a penalty in terms of throughput.

The implementation proposed by Bascones *et al.* [28] works also in BIP order and it is optimized to use only internal memory storage, proposing the use of FIFOs to store the weights values, together with the local sum and differences calculations, being available when the next sample is processed. Using this scheme, they achieve a throughput of almost 48 Msamples/s when it is implemented on a Xilinx Virtex-7 XC7VX690T. In a later development by the same authors [29], a parallel implementation is proposed, defining multiple copies of the compressor working simultaneously and using a shared memory to resolve data dependencies between consecutive samples. In this case, a maximum throughput of 179.7 MSamples/s is obtained targeting a Xilinx Virtex-5 XQR5VFX130, when 7 compression instances are working in parallel. However, the resources utilization is incremented significantly, consuming almost the 60% of BRAMs available in this device.

To the best of our knowledge, the fastest implementation available in the state-of-the-art using only one compression instance is the proposed by Tsigkanos *et al.* [30]. In this work, the authors exploit the parallelism under BIP ordering and implement a fine-grained pipeline in critical feedback loops based on C-slow retiming, achieving up to 213 MSamples/s on a Xilinx Virtex-5 FX130T, the equivalent commercial version of the space-qualified Virtex-5 XQR5VFX130. They introduce an extra module, named as Spectral Slice Buffer, in order to have the necessary neighbouring available when the local sum calculation is going to take place. In a more recent work [35], the authors also provide results on the next-generation Xilinx XQRKU060 FPGA, achieving a maximum throughput of 315 MSamples/s impelementing the same architecture described in [30]. Although this work will be useful as a reference for future developments, it is out of our comparison because of the high technology gap with the FPGAs selected as target device by the rest of the implementations analysed.

In addition to implementations targeting space-grade FPGA technologies, there is a trend of using Commercial-Off-The-Shelf (COTS) FPGAs for space-related applications, such as SmallSats and Low-Earth Orbit (LEO) missions. Although the performance of this technology is higher than the achieved by RHBD FPGAs, commercial (generally, SRAM-based) FPGAs are vulnerable to ionizing radiation present in harsh environments. As it is well known, radiation may cause different kinds of Single Event Effects (SEEs), such as bit flips at memory configuration (SEUs) or even functional interrupts of the electronic system (SEFIs). Therefore, mitigation techniques shall be applied to the design in order to provide robustness against radiation, as it has not been considered during the device manufacturing process.

In this way, Pereira *et al.* [31] propose a low-complexity implementation of the prediction stage (column-oriented local sum and reduced prediction mode are selected), targeting a Xilinx Zynq-7020 Multi-Processor System-on-Chip (MPSoC) and achieving a maximum throughput of 20.4 MSamples/s. This architecture works in BIP order, storing the weights in an internal memory (implemented with flip-flops) but out of the processing core, in order to share these values among multiple copies of the compression instance, if needed for higher throughput.

Rodríguez *et al.* propose in [34] a parallel implementation of HyLoC, previously described in this study. For this purpose, the ARTICo^3 framework is used, a hardware-based processing architecture for high-performance embedded systems developed by some the same authors. Using the toolchain provided by ARTICo^3, 16 instances of HyLoC are implemented in parallel in a Xilinx Zynq-7100 MPSoC, achieving a maximum throughput of 67.04 MSamples/s. In this case, a considerable penalty is observed in terms of resources utilization, being difficult its implementation on current space-grade FPGAs if the number of accelerators are not limited to lower values.

The work of Fjeldtvedt *et al.* [32], also targeting a Xilinx Zynq-7020 MPSoC, proposes a BIP architecture with a Sample Delay module that works in a similar way that the Slice Buffer proposed by [30]. Different pipelining strategies are applied in the prediction module in order to reduce delay,

TABLE 15. Comparison with CCSDS-123 FPGA implementations.

Implementation	Order	P	D	Encoder	Device	LUTs	FFs	DSPs	BRAMs	freq. (MHz)	Throughput (MSamples/s)
HyLoC, Santos et al. [24]	BSQ	3	16	Sample	Virtex-5 XQR5VFX130	2342	1535	1	0	134	11.3
Keymeulen et al. [27]	BIP	3	13	Golomb-Rice	Virtex-5 SX50T	12697	1586	3	8	40	40
Bascones et al. [28]	BIP	0-15	16	Sample	Virtex-7 XC7VX690T	-	-	-	-	50	47.6
Bascones et al. (per core) [29]	BIP	0-15	16	Sample	Virtex-7 XC7VX690T	6931	-	18	88	-	-*
Tsigkanos et al. [30]	BIP	3	16	Sample	Virtex-5 FX130T	9462	9990	6	83	213	213
Pereira et al. [31]	BIP	3	16	No	Zynq-7020	2244	630	3	0	142.9	20.4
Fjeldtvedt et al. [32]	BIP	0-15	16	Sample	Zynq-7020	3012	2528	6	84	147	147
Orlandic et al. (per core) [33]	BIP	0-15	16	Sample	Zynq-7035	3747	2887	9	33	150	150
Rodríguez et al. [34]	BSQ	3	16	Sample	Zynq-7100	51070	26830	16	256	100	67.04
SHyLoC 1.0 [14]	All	0-15	16	Sample/Block	Virtex-5 XQR5VFX130	5815	3658	10	74	113.9	113.3
This work	All	0-15	16	Sample/Block	Virtex-5 XQR5VFX130	4809	2736	10	74	138.3	138.3
This work	All	0-15	16	Sample/Block	Zynq-7035	4619	2765	8	74	151.1	151.1

**It is not possible to infer a per-core throughput from [29]. The authors provide a value of 219.4 MSamples/s for a parallel implementation with 7 cores*

splitting the critical paths located in the local sum calculation and in the weights updating. With this scheme, they achieve a maximum throughput of 147 MSamples/s. In [33], the same authors propose a parallel implementation of the previous work, sharing the Sample Delay module and the central differences, weights and accumulator memories among the compression instances. In this case, the design is mapped on a Xilinx Zynq-7035 MPSoC, obtaining different performances depending on the number of pipelining stages implemented. For example, a maximum throughput of 750 MSamples/s is achieved when 5 compression copies are working simultaneously.

Finally, the SHyLoC 2.0 is also compared against its version 1.0 [14]. Both versions are quite similar, except for the new features aforementioned in section III: the inclusion of the unit-delay predictor in the CCSDS-121 IP, along with the necessary changes in the block-coder to process reference samples, the extension of the dynamic range and the support of signed samples; as well as the support for burst transfers in the CCSDS-123 IP. Regarding the CCSDS-123 IP, both versions can be easily compared except for the new BIL-MEM architecture, which is exclusive of the SHyLoC 2.0. As it can be observed in Table 15, SHyLoC 2.0 achieves an improvement of the 22% in terms of throughput regarding its predecessor, working both versions in BIP ordering and targeting the AVIRIS sensor. At the same time, a reduction of the logic resources utilization is achieved (around the 21% of LUTs and the 34% of FFs). Therefore, results prove that SHyLoC 2.0 offers an improved on-board compression solution, including also new features not present in other state-of-the-art implementations of the CCSDS 123-0-B-1 standard.

The SHyLoC 2.0 CCSDS-123 IP proposed in this work achieves a maximum throughput of 138.3 MSamples/s targeting a Xilinx Virtex-5 XQR5VFX130. This IP has been also mapped on the Xilinx Zynq-7035 MPSoC in order to compare it with implementations on COTS devices, reaching a throughput of 151.1 MSamples/s. For both cases, the AVIRIS sensor was taken as reference, being the BIP ordering the architecture that obtains the best results because of it is the only one capable of processing one sample per clock cycle. If higher throughput is required, several measures can be taken by the user, such as breaking the critical path with intermediate registers, or instantiating several copies of SHyLoC to work in parallel. Although our implementation of the CCSDS 123-0-B-1 standard exhibits lower throughput than the implementation presented in [30], it represents a good trade-off among timing performance, resources utilization, versatility and portability, based on the results provided in this paper.

V. CONCLUSION

This work describes the SHyLoC 2.0, a novel implementation of the CCSDS 121.0-B-2 and 123.0-B-1 lossless compression standards in the form of two reusable IP Cores, implementing respectively each one of the aforementioned CCSDS standards. SHyLoC 2.0 offers new features and performance improvements that were not present in its predecessor, SHyLoC 1.0, which belongs to the portfolio of ESA IP cores for space missions. The throughput in terms of compressed samples per second is increased thanks to the introduction of burst transactions with selectable burst size to communicate with an external memory. Additionally, the compression efficiency of the CCSDS 121-IP is potentially improved by offering the possibility to use a unit-delay predictor to enhance the entropy coder performance. To ensure reusability, the IP cores are highly configurable, offering control to the user on all possible trade-offs, by selecting the appropriate parameters that will result in an optimum implementation that fits in a huge variety of target space-qualified FPGA devices, ranging from small anti-fuse FPGAs to bigger and faster devices such as the Xilinx Virtex-5 QV130T. This is demonstrated in this paper by presenting the most remarkable results obtained in terms of throughput, maximum clock frequency and resources utilization. To the best of our knowledge, this is the most complete and versatile implementation of the CCSDS lossless compression standards, supporting most of the configuration options described on them.

A hardware implementation of the new CCSDS 123-0-B-2 compression standard [36] is proposed as future work. The main feature introduced by this Issue 2 is a quantization loop which enables near-lossless compression. In addition, support for input samples with up to 32 bits of resolution, the definition of a hybrid encoder for low-entropy data or a narrow mode to avoid the use of the previous sample in the current band for the local sum calculation are introduced, among other new characteristics. The quantization loop introduces new data dependencies that were not present in the Issue 1 of the standard, thus a new schedule must be defined in order to obtain similar throughput to the purely lossless version.

REFERENCES

[1] J. Transon, R. d'Andrimont, A. Maugnard, and P. Defourny, "Survey of hyperspectral Earth observation applications from space in the Sentinel-2 context," *Remote Sens.*, vol. 10, no. 3, p. 157, 2018. [Online]. Available: http://www.mdpi.com/2072-4292/10/2/157

[2] C. M. Lee, M. L. Cable, S. J. Hook, R. O. Green, S. L. Ustin, D. J. Mandl, and E. M. Middleton, "An introduction to the NASA hyperspectral InfraRed imager (HyspIRI) mission and preparatory activities," *Remote Sens. Environ.*, vol. 167, pp. 6–19, Sep. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0034425715300419

[3] J. Nieke and M. Rast, "Towards the copernicus hyperspectral imaging mission for the environment (CHIME)," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2018, pp. 157–159.

[4] C. Thiebaut and R. Camarero, "CNES studies for on-board compression of high-resolution satellite images," in *Satellite Data Compression*, B. Huang, Ed. New York, NY, USA: Springer, 2011, ch. 2, pp. 29–46.

[5] E. Christophe, "Hyperspectral data compression tradeoff," in *Optical Remote Sensing: Advances in Signal Processing and Exploitation Techniques*, S. Prasad, L. M. Bruce, and J. Chanussot, Eds. Berlin, Germany: Springer, 2011, ch. 2, pp. 9–29.

[6] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2000, pp. 523–541.

[7] I. Blanes and J. Serra-Sagrista, "Cost and scalability improvements to the Karhunen–Loêve transform for remote-sensing image coding," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 7, pp. 2854–2863, Jul. 2010.

[8] I. Blanes and J. Serra-Sagrista, "Pairwise orthogonal transform for spectral image coding," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 3, pp. 961–972, Mar. 2011.

[9] A. J. Hussain, A. Al-Fayadh, and N. Radi, "Image compression techniques: A survey in lossless and lossy algorithms," *Neurocomputing*, vol. 300, pp. 44–69, Jul. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231218302935

[10] *Lossless Multispectral and Hyperspectral Image Compression*, Standard Recommended CCSDS 123.0-B-1, CCSDS, 2015.

[11] *Lossless Data Compression*, Standard Recommended CCSDS 121.0-B-2, CCSDS, May 2012.

[12] ESA. *ESA HDL IP Cores Portfolio Overview*. Accessed: Jun. 11, 2019. [Online]. Available: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Microelectronics/ESA_HDL_IP_Cores_Portfolio_Overview

[13] ESA. *SHyLoC IP Core*. Accessed: Sep. 14, 2018. [Online]. Available: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/SHyLoC_IP_Core

[14] L. Santos, A. Gomez, and R. Sarmiento, "Implementation of CCSDS standards for lossless multispectral and hyperspectral satellite image compression," *IEEE Trans. Aerosp. Electron. Syst.*, to be published.

[15] S. Lopez, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos, and A. Plaza, "The promise of reconfigurable computing for hyperspectral imaging onboard systems: A review and trends," *Proc. IEEE*, vol. 101, no. 3, pp. 698–722, Mar. 2013.

[16] *NX Products Overview*, 2nd BRAVE FPGA Days, NanoXplore, Ottawa, ON, Canada, Nov. 2018.

[17] J. E. Sánchez, E. Auge, J. Santalo, I. Blanes, J. Serra-Sagrista, 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 Compress., Commun. Process.*, Jun. 2011, pp. 222–228.

[18] E. Augé, J. E. Sánchez, A. Kiely, I. Blanes, and J. Serra-Sagrisà, "Performance impact of parameter tuning on the CCSDS-123 lossless multi- and hyperspectral image compression standard," *J. Appl. Remote Sens.*, vol. 7, no. 1, Aug. 2013, Art. no. 074594. [Online]. Available: https://ddd.uab.cat/record/129769

[19] M. Klimesh, "Low-complexity lossless compression of hyperspectral imagery via adaptive filtering," Jet Propuls. Lab. (JPL), Nat. Aeronaut. Space Admin. (NASA), Pasadena, CA, USA, Progress Rep. 42-163, Jan. 2005.

[20] *Lossless Multispectral and Hyperspectral Image Compression*, Informational Report CCSDS 120.2-G-1, CCSDS, 2015.

[21] University of Las Palmas de Gran Canaria. (Oct. 2017). *SHyLoC Product Datasheet*. [Online]. Available: https://amstel.estec.esa.int/tecedm/ipcores/SHyLoC_Datasheet_v1.0.pdf

[22] *AMBA Specification, Rev 2.0*, ARM, ARM Ltd., Cambridge, U.K., May 1999.

[23] ESA. *Data Compression Tools*. Accessed: Mar. 27, 2018. [Online]. Available: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Data_Processing/Data_Compression_Tools

[24] L. Santos, L. Berrojo, J. Moreno, J. F. Lopez, 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.

[25] 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.

[26] B. Hopson, K. Benkrid, D. Keymeulen, and N. Aranki, "Real-time CCSDS lossless adaptive hyperspectral image compression on parallel GPGPU multicore processor systems," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Jun. 2012, pp. 107–114.

[27] 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. Adapt. Hardw. Syst. (AHS)*, Jul. 2014, pp. 278–284.

[28] D. Bascones, C. Gonzalez, 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.

[29] D. Báscones, C. González, and D. Mozos, "Parallel implementation of the CCSDS 1.2.3 standard for hyperspectral lossless compression," *Remote Sens.*, vol. 9, no. 10, p. 973, 2017. [Online]. Available: https://www.mdpi.com/2072-4292/9/10/973

[30] 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.

[31] L. M. V. Pereira, D. A. Santos, C. A. Zeferino, and D. R. Melo, "A low-cost hardware accelerator for CCSDS 123 predictor in FPGA," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[32] 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.

[33] M. Orlandić, J. Fjeldtvedt, and T. Johansen, "A parallel FPGA implementation of the CCSDS-123 compression algorithm," *Remote Sens.*, vol. 11, no. 6, p. 673, 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/6/673

[34] A. Rodriguez, L. Santos, R. Sarmiento, and E. De La Torre, "Scalable hardware-based on-board processing for run-time adaptive lossless hyperspectral compression," *IEEE Access*, vol. 7, pp. 10644–10652, 2019.

[35] A. Tsigkanos, N. Kranitis, and A. Paschalis, "CCSDS 123.0-B-1 multispectral & hyperspectral image compression implementation on a next-generation space-grade SRAM FPGA," in *Proc. 6th Int. Workshop Board Payload Data Compress (OBPDC)*, Sep. 2018, pp. 1–8.

[36] *Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression*, document CCSDS 123.0-B-2, Consultative Committee for Space Data Systems, Feb. 2019, vol. 2.

YUBAL BARRIOS was born in Las Palmas de Gran Canaria, Spain, in 1993. He received the degree in telecommunications engineering from the University of Las Palmas de Gran Canaria, in 2016, and the M.S. degree in telecommunications technologies in 2017. In 2017, he has been funded by the Institute for Applied Microelectronics (IUMA), where he has conducted his research activities at the Integrated Systems Design Division in the context of hardware implementations for hyperspectral image compression on FPGAs and MPSoCs, applying both High-Level Synthesis and Register-Transfer Level design methodologies. He currently combines his job in the IUMA research lines with his Ph.D. thesis. In 2019, he was invited as a Visiting Researcher by the Microelectronics Section, European Space Research and Technology Centre (ESTEC), Core of the European Agency (ESA), Noordwijk, The Netherlands. His current research interests include the development of efficient algorithms for on-board hyperspectral image compression and reconfigurable hardware architectures optimized in terms of throughput, memory usage, and power consumption.

ANTONIO J. SÁNCHEZ received the degree in industrial engineering from the Universidad Carlos III de Madrid (UC3M), in 2011, and the master's degree in advanced electronic systems and the Ph.D. degree in electric, electronic and automatic engineering UC3M, in 2013 and 2017, respectively. Then, he joined the Microelectronic Design and Applications Research Group, Electronic Technology Department, UC3M, where he worked as an Assistant Researcher. His Ph.D. thesis was focused on the design of approximate logic circuits and their application to the fault-tolerance field. He was a Visiting Researcher with the European Space Research and Technology Centre, The Netherlands. He is currently a member of the Integrated Systems Design Division, Institute for Applied Microelectronics (IUMA), University of Las Palmas de Gran Canaria. His research at IUMA is focused on the hardware implementation of algorithms for hyperspectral image processing for space applications, including the use of high level synthesis tools. Additionally, his research interests include fault tolerant hardware design, approximate computing, and formal verification methods.

LUCANA SANTOS received the degree in telecommunication engineering 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. 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. 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 coauthored several scientific articles 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 was the Dean of the Faculty, from 1994 to 1998, and a Vice Chancellor of academic affairs and a Staff with ULPGC, from 1998 to 2003. He is currently a Full Professor of electronic engineering with the Electronics and Telecommunication Engineering School, University of Las Palmas de Gran Canaria, Spain. He is also a Co-Founder of the Research Institute for Applied Microelectronics (IUMA), where he is also the Director of the Integrated Systems Design Division. He has published more than 90 journal articles and more than 160 conference papers. He has been awarded with five six years research periods by the National Agency for the Research Activity Evaluation in Spain. He has participated in more than 60 projects and research programmes funded by public and private organizations. His current research interest is related to electronics system on-board satellites.