EXPRO+ES AO/1-8032/14/NL/AK

IUMA/1410/AO8032

SHyLoC 2.0 Final Report

By

University of Las Palmas de Gran Canaria
Institute for Applied Microelectronics (IUMA)
Spain

19 August 2020

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada

DISCLAIMER

DOCUMENT HISTORY

Date	Version	Author	Description	Status
21-02-2019	1.0	Yubal Barrios	[SHyLoC-e] project (CCN2)	Ongoing
26-02-2019	1.1	Yubal Barrios	Review	Ongoing
04-03-2019	1.2	Antonio Sánchez	Review	Ongoing
06-03-2019	1.3	Yubal Barrios	Review	Ongoing
28-03-2019	1.4	Yubal Barrios	CCSDS-123 results updated	Ongoing
29-03-2019	1.5	Roberto Sarmiento	Review	Delivered FR

LIST OF AUTHORS

Partner	Authors
IUMA	Yubal Barrios
IUMA	Antonio Sánchez
IUMA	Roberto Sarmiento

TABLE OF CONTENT

FIGURES

TABLES

GLOSSARY

Acronym	Meaning
CCSDS	Consulting Committee for Space Data System
FPGA	Field Programmable Gate Array
ASIC	Application-Specific Integrated Circuit
VHDL	VHSIC Hardware Description Language
ESL	Electronic System Level
ITT	Invitation to Tender
IP	Intellectual Property
ADC	Analog-to-Digital Converter
AMBA	Advanced Microcontroller Bus Architecture
AHB	Advanced High-Performance Bus
SpW	SpaceWire
EDAC	Error Detection and Correction
EGSE	Electrical Ground Support Equipment
DUT	Device Under Test

1 INTRODUCTION

1.1 Document scope

This document corresponds to the deliverable D8 (Final Report) of the ESA Contract No. 4000113182/15/NL/LF entitled CCSDS Lossless Compression IP-Core Space applications. The document was updated based on the "Extension of the SHyLoC IP Cores: Improving Lossless Compression for Space Application" approved by ESA on December 2017.

1.2 Applicable documents

[AD-1] *Lossless Multispectral & Hyperspectral Image Compression*. Recommendation for Space Data System Standards, CCSDS 123.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 2012.

[AD-2] *Lossless Data Compression*. Recommendation for Space Data System Standards, CCSDS 121.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, April 2012.

[AD-3] ESA ITT AO/1-8032/14/NL/AK, *CCSDS Lossless Compression IP-Core Space Applications*, Statement of Work, ESA, September 2014.

[AD-4] ESA Express Procurement "EXPRO", *Extension of the SHyLoC IP Cores: Improving Lossless Compression for Space Application*, ESA, February 2018.

[AD-5] TRP AO8032 – Deliverable D5, Verification and Validation Plan, IUMA, October 2018

[AD-6] TRP AO8032 – Deliverable D2, Requirements Specifications, IUMA, October 2018

1.3 Reference Documents

[RD-1] Lossless data compression recommended standard CCSDS 121.0-B-2, The Consultative Committee for Space Data Systems, 2012.

[RD-2] Lossless multispectral and hyperspectral image compression recommended standard CCSDS 123.0-B-1, The Consultative Committee for Space Data Systems, 2012.

[RD-3] ESA Express Procurement (EXPRO), Statement of Work, Extension of the SHyLoC IP Cores: Improving Lossless Compression for Space Application, European Space Agency, 2018.

[RD-4] European Space Agency. (2016) ESA IP Cores. [Online]. Available: http://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/About_ESA_IP_Cores

[RD-5] European Space Aagency (ESA). (2013) CCSDS 123.0-B-1 multispectral and hyperspectral lossless data compression. [Online]. Available: http://www.esa.int/TEC/OBDP/SEM069KOXDG_2.html

[RD-6] J. Sanchez, A. E., K. A., B. I., and J. Serra-Sagristà, "Performance impact of parameter tuning on the CCSDS-123 lossless multi- and hyperspectral image compression standard," in Proceedings of 2012 ESA workshop on Onboard Payload Data Compression (OBPDC), 2012.

[RD-7] TRP AO8032 – Deliverable D3, IP cores Datasheet, IUMA, October 2018

[RD-8] L. Santos, A. Gómez, R. Sarmiento, L. Fossati and D. Merodio, "," in Proceedings of 2016 ESA workshop on Onboard Payload Data Compression (OBPDC), 2016.

[RD-9] European Space Agency. (2018) SHyLoC IP Core. [Online]. Available: http://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/SHyLoC_IP_Core

[RD-10] Test vectors for CCSDS121 standard: http://cwe.ccsds.org/sls/docs/sls-dc

1.4 Cross-reference

This deliverable D8 (SHyLoC-e Final Report) has been created based on the document TRP-AO8032_D8_Summary_Report_v1.4.docx, delivered on July 2017 for the Final Review of the project EXPRO+ES AO/1-8032/14/NL/AK.

2 SHYLOC SUMMARY REPORT

2.1 Introduction and motivation

High data rate multispectral and hyperspectral sensors and the limitations of the on-board storage and bandwidth make on-board data reduction mandatory. The Consultative Committee for Space Data Systems (CCSDS), which represents the major space agencies in the world, has issued two lossless standards for satellite data compression: the universal CCSDS-121 [RD-1] and the CCSDS-123 [RD-2], which targets multispectral and hyperspectral data.

These two lossless standards are capable of reducing the data volume by removing redundancies in the data source, in such a way that the original data can be fully recovered after decompression. Lossless compression is particularly important in those applications in which data integrity cannot be compromised. The CCSDS-121 is a universal compressor, which consists of a unit-delay predictor and an entropy coder that utilizes Rice coding technique to encode blocks of data by selecting among a set of pre-defined codes the one that yields the shortest encoded block. The CCSDS-123 focuses on three-dimensional images (multispectral or hyperspectral), consisting of a pre-processor that removes redundancies among samples in a three-dimensional neighbourhood and a subsequent entropy coding stage. The CCSDS-123 offers two options for the entropy coder: the sample-adaptive encoder, which uses Golomb power-of-two codes; or the entropy coder defined by the CCSDS-121 standard (block-adaptive encoder).

Despite the availability of algorithms, efficiently computing the compression in the hardware technologies available on a satellite is an important challenge, due to the limitations imposed by the space environment. Generally, the algorithms are implemented on FPGAs or ASICs in order to optimize their performance and power consumption. In order to endorse on-board compression presence, it is essential to design low-complexity and high-throughput hardware architectures that can be efficiently implemented in the aforementioned technologies.

We present the design and description in VHDL of two synthesizable IP cores that implement lossless compression algorithms, as defined by the CCSDS-123 and CCSDS-121 standards. Such IPs are capable of working independently, as well as jointly. In the latter case, the CCSDS-123 IP works as a pre-processor and the CCSDS-121 IP performs only the entropy coding stage. The designed IP cores are technology independent and can be mapped to several FPGA targets representative of space-grade hardware.

Additionally, this project required the IP cores to be able to accept all possible configuration modes allowed by the standards. This represents an important challenge, because the design of the compression hardware architecture has to consider various trade-offs between data reduction and potential complexity that arise from the selection of the compression parameters. In this sense, an extensive hardware validation against reference software at RTL is performed.

2.2 Overview of the CCSDS lossless compression algorithms

The CCSDS-121 standard defines a universal lossless compressor consisting in an optional unit-delay predictor and a block-adaptive entropy coder [RD-1]. On the other hand, the CCSDS-123 standard [RD-2] describes a lossless compressor focused on multispectral and hyperspectral images, consisting of a predictor and an entropy coder. Two different options are offered for the latter: sample-adaptive and block-adaptive coding, which corresponds to the CCSDS-121 encoder stage [RD-1]. The input of both compressors is a multispectral or hyperspectral image, which is a three-dimensional array of integer sample values. The output is an encoded bitstream from which the input samples can be fully recovered.

2.2.1 Prediction

When enabled, the CCSDS121-IP pre-processing stage attempts to remove the correlation among data samples (predictor) and reformat them into a more suitable probability distribution (mapper). The proposed predictor in the CCSDS 121 standard is a Unit-Delay predictor, where the previous sample x_{i-1} is used as an estimator $\hat{x}_i$ of the current sample x_i. Then, the prediction error Δ_i with respect to the current sample is computed, and it is mapped into a non-negative integer δ_i, known as the mapped prediction residual.

In the CCSDS123-IP, the prediction of a sample, $s_{z,y,x}$, is performed in a three-dimensional neighbourhood of previously processed samples, as shown in Figure 2-1. The predicted sample $\hat{s}_{z,y,x}$ is computed using the previously processed neighbouring samples of $s_{z,y,x}$ in the current band as well as in P previous bands. P, is a user-defined parameter that can range from 0, for which no information from previous bands is utilized, up to 15.

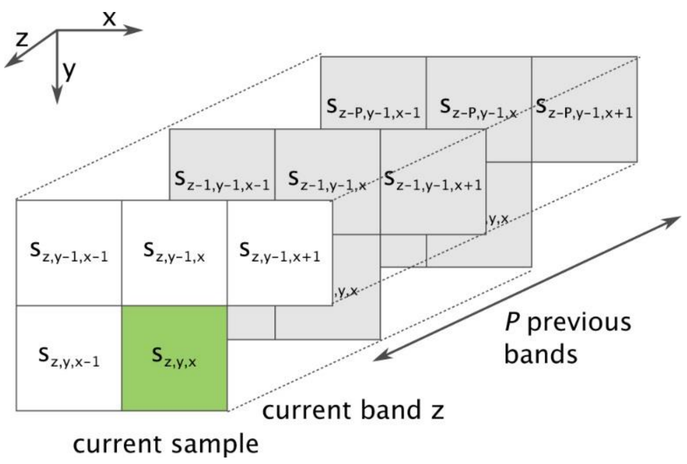


Figure 2-1: Three-dimensional neighbourhood used for prediction in the CCSDS 123 standard.

First, a local sum, $\sigma_{z,y,x}$, of the neighbouring samples of $s_{z,y,x}$ in the current band is computed. The local sums are used to calculate the central local differences values, $d_{z,y,x}$ and the directional local differences: $d^N_{z,y,x}$, $d^W_{z,y,x}$ and $d^{NW}_{z,y,x}$. The central local differences in the P previous bands, together with the 3 directional local differences constitute a vector $U_{z,y,x}$ whose elements are computed according to the selected configuration. The number of elements of the local differences vector is Cz, where $Cz = P$ when reduced prediction is selected and $Cz = P + 3$ for full prediction.

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

2.2.2 Entropy coding

The CCSDS-123 standard allows the selection between a sample-adaptive entropy coder and a block-adaptive entropy coder that are described next. Under the sample-adaptive entropy coding approach, the mapped prediction residuals δ_i are encoded using a Golomb power-of-two variable-length binary codeword. The codes are adaptively selected based on statistics, which consist of a running sum -accumulator- of mapped residuals in the spectral band, and a counter. The accumulator and the counter are updated after each sample is encoded and reset periodically according to an interval set by a user-defined parameter.

The block-adaptive entropy coder utilizes the Rice coder defined in the CCSDS-121 standard [RD-1]. In Rice's coding, several algorithms are concurrently applied to a block of J consecutive pre-processed samples, as shown in Figure 2-2.

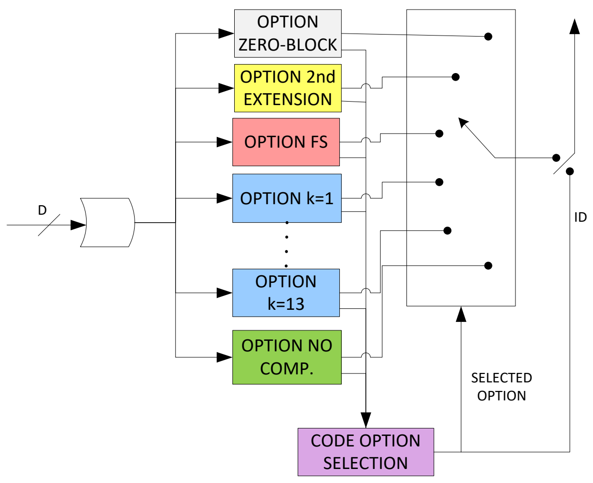


Figure 2-2: Block-adaptive entropy coder concept.

The code selector selects the coding option that minimizes the number of bits (including ID bits) used to encode a block of J samples. The ID bit sequence specifies which option was used to encode the accompanying block of samples.

2.3 ESA ITT AO/1-8032/14/NL/AK: SHyLoC project extension (SHyLoC-e) objectives and requirements

The work presented in this report follows the statement of work established by the European Space Agency in the scope of the ESA Express Procurement (EXPRO) entitled: "Extension of the SHyLoC IP Cores:

Improving Lossless Compression for Space Application" [RD-3]. The main requirements and objectives are summarized hereafter.

The main goal of this EXPRO consists of extending the SHyLoC compression IP cores [RD-9] in order to make it capable of achieving higher performance in terms of both compression efficiency and throughput. The extensions, detailed in the following subsections, shall be in compliance of the lossless compression algorithms defined by the CCSDS standards number 121 [RD-1] and 123 [RD-2].

The correct behaviour of the IP cores shall be demonstrated by performing extensive simulations targeting real and synthetic hyperspectral images, and cross-validating the results with relevant golden-reference implementations.

The produced extensions to the IP cores shall be implemented using the VHDL language, guaranteeing maximum reusability and minimising the IP-Core user's need of technical support. The extensions shall be provided with the already existing IP Core in a single database, and they should be technology agnostic.

2.3.1 Extensions of functionality in the CCSDS 121 IP Core

The current implementation of the CCSDS 121 IP [RD-9] core includes the block-adaptive entropy coder defined by the standard. When the CCSDS 121 IP is used as an independent IP, the compression efficiency can be improved by including a pre-processing stage. However, the only pre-processor included in the SHyLoC IP is the one defined in the CCSDS 123 algorithm, which specifically targets hyperspectral and multispectral data and has a high complexity [RD-2]. The CCSDS 121 standard foresees the inclusion of a lightweight pre-processor [RD-1].

An objective of this work is extending the functionality of the CCSDS 121 IP core by adding the pre-processor defined in the CCSDS121 standard [RD-1], allowing it to achieve higher compression ratios when used as standalone IP. Additionally, the IP shall be extended to include a reference sample in the bitstream and the necessary headers to enable its standalone use. The resulting implementation shall be cross-verified against the compression tools developed at ESA [RD-5]. Correctness of the implementation shall be demonstrated using the reference test vectors available in [RD-10].

2.3.2 Extensions of functionality in the CCSDS 123 IP Core

The CCSDS 123 algorithm performs the prediction based on an adaptive linear method. For each sample, in the prediction calculation, a *weight vector* ($W_z(t) = [\omega_z^{(1)}(t), \omega_z^{(2)}(t) \,..\, \omega_z^{(P_z^*)}(t)]$) is multiplied by a local differences vector. Such weight vector is updated based on the prediction error. The number of elements in the vector is C_z. The initial weight vector, $W_z(1)$, for each spectral band z might consist of the default values defined in Section 4.6.3.2 of [AD-1]; or custom values selected by the user, as defined in Section 4.6.3.3. of [AD-1].

When the default weight initialization is selected, the initial weight vector $W_z(1)$ is the same for all bands in the image. Each element in the vector is assigned a pre-defined value. When custom weight initialization is used, for each spectral band z, the initial weight vector $W_z(1)$ is assigned using a user-specified weight initialization vector Λ_z, consisting of C_z signed Q −bit integer components.

SHyLoC includes currently the default weight initialization option only, due to the inherent complexity of the custom weight initialization implementation. This work aims at analysing the possible trade-offs in order to extend SHyLoC to include custom weight initialization in a way that the current performance in terms of hardware occupancy and throughput is not compromised. Once the best architectural solution is found, a specifications document must be prepared in order to be used as a guide for its future implementation.

In addition, the CCSDS 123 IP Core offers different architectural solutions, aiming at making it feasible to achieve the best compromise between performance and hardware occupancy for the possible sample arrangements that can be found in hyperspectral data (BIP, BSQ and BIL) [RD-7].
For BIP order, two solutions were proposed for the storage of intermediate results during the compression process. One of the solutions uses only internal FPGA memory and the other (BIP-Mem) stores the intermediate results in an external memory that is accessed through the AHB bus. The implementation of the solution with internal memory only is many times unfeasible for some target devices, as the amount of memory required scales up with the size of the image to be compressed. Nevertheless, it achieves a very high throughput due to the inherent data parallelism found in BIP order. When external memory is used, in BIP-Mem, this throughput is drastically reduced (almost to 50%), because the AHB bus capabilities are not efficiently exploited in the current implementation.

The use of AHB bus in the BIP-Mem architecture shall be optimized, by appropriately using burst transaction and read/write interleaving, so that the performance decrease of the BIP-Mem architecture is reduced. The same solution shall be then applied to the BIL architecture, in order to offer an option with access to external memory also. The resulting implementation shall be verified by running the testbenches devised for the SHyLoC-e IP [AD-5].

2.4 SHyLoC-e VHDL description

SHyLoC-e consists of two IP cores: the CCSDS-121 and the CCSDS-123, which are described in VHDL. They are capable of work independently as well as jointly, allowing different implementation schemes as shown in the Figure 2-3. In order to do so, they have compatible interfaces, comprising input and output data interfaces, an AHB slave interface for configuration and a control interface to inform about the status of the compression. Since the CCSDS-123 predictor needs to store a fair amount of intermediate values during the compression, an AHB master interface is included to allow the storage of these samples in a memory external to the FPGA.

The IP cores are fully configurable, reflecting the versatility of the standards. Configuration is implemented by means of VHDL constants and registers. The constants allow the user to tailor the implemented IP core to the selected values, enabling the generation of a lower area/memory occupation version of the IP core by setting limits to the run-time configuration values that might be selected. Additionally, run-time configuration might be enabled or disabled by setting a VHDL constant. When run-time configuration is disabled, the IP core is configured at compile-time using the aforementioned set of constants. On the other hand, when run-time configuration is enabled, the configuration values are read from memory-mapped registers. Both IP cores include a configuration engine that reads the received configuration values from the AHB slave interface, checks that the configuration is valid and broadcasts the values to the rest of the components of the design.

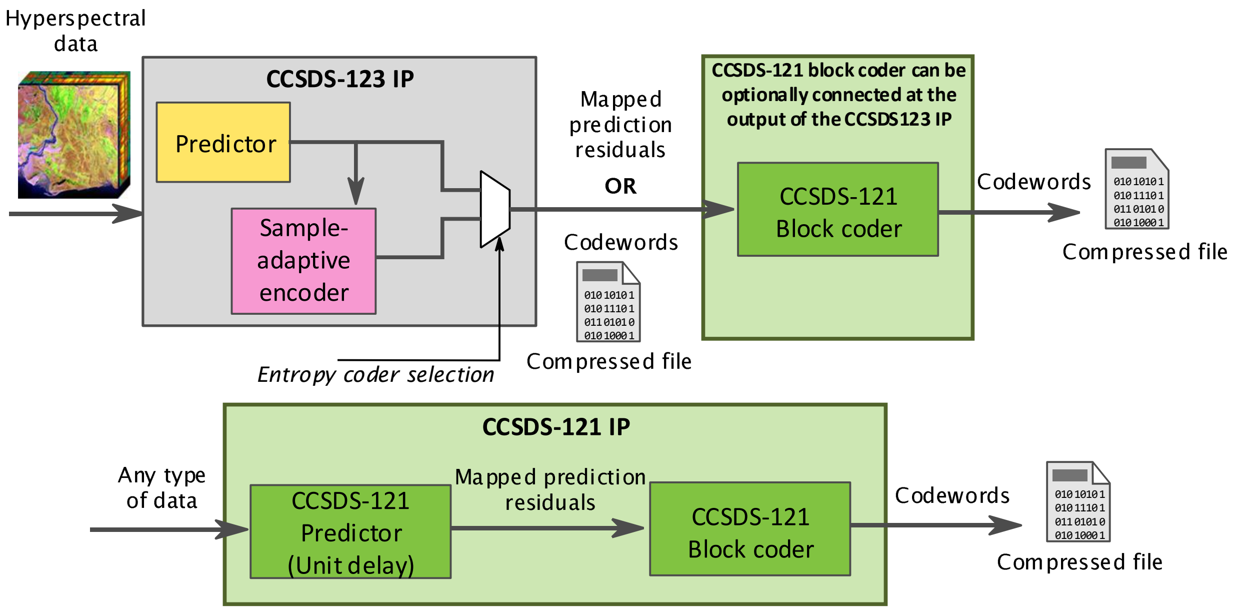


Figure 2-3 Shyloc-e IP cores and connectivity between them

2.4.1 The CCSDS-121 IP core

The VHDL description of the CCSDS-121 IP core includes, besides the described interfaces and configuration engine, a set of components that perform all the necessary steps in the data compression flow. More specifically, the prediction of incoming samples is performed by the predictor module, while the block-coder is in charge of the entropy coding of blocks of samples (which may be pre-processed or not, depending on whether the pre-processor is present or not). In addition, the block-coder receives the runtime configuration from the AHB bus and disseminates the configuration to the predictor module (if present). The schematic is shown in the Figure 2-4.

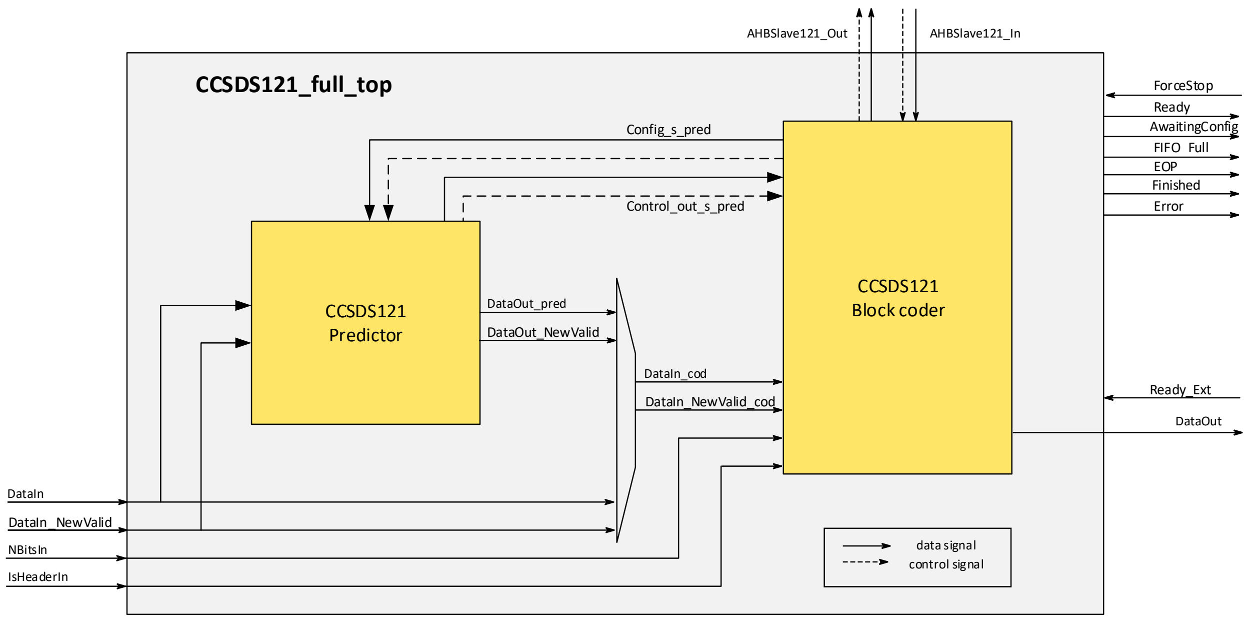


Figure 2-4 General view of the CCSDS-121 IP

The predictor includes a unit-delay predictor which computes the predicted samples based on the input data, and a mapper module which maps the prediction residuals. These steps can be bypassed by the unit control in order to periodically introduce reference samples in the compressed data. The predictor module is optional. If present, the data input of the block coder module is connected to the output of the predictor, and the input samples can be either signed or unsigned. Otherwise, the IP input samples are directly connected to the block-coder inputs, which are treated as unsigned samples.

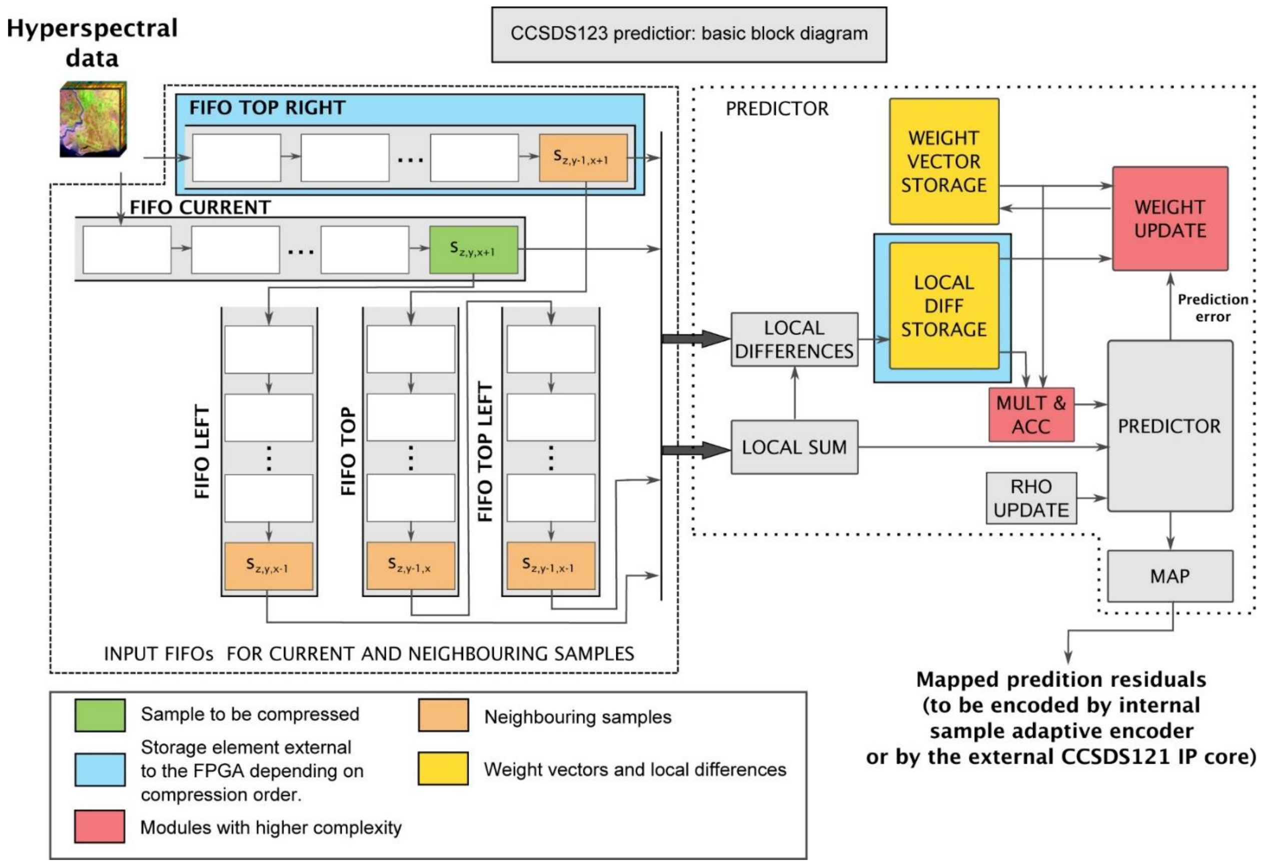
With respect to the block-coder, the input samples to be encoded δ_i are stored in a FIFO until a block of J samples is completed. The length of all encoding options is then calculated and stored in accumulators, whose value is then compared to select the smallest, taking also into account the possibility that all blocks are zero. Once a compression option is chosen, the input samples are encoded. While the options are calculated, the mapped residuals and the γ_i values computed for the second-extension option are stored in FIFOs, making it possible to parallelize the selection of the encoding option with the generation of the codeword. Finally, an output bit packer is used to create words of the size of the output buffer. When the unit-delay predictor is present, the block-coder periodically introduces reference samples according to the CCSDS-121 standard [AD-2], with a periodicity determined by the current configuration.

The size of the FIFOs of the design and the bit width of the registers and accumulators are optimized according to the size of the user-selected size of the block (J), the dynamic range of the input data (D) and the amount of encoding options to be evaluated.

2.4.2 The CCSDS-123 IP core

The CCSDS-123 IP core defines a configuration engine, a component performing the prediction, a sample-adaptive encoder, and a dispatcher. The dispatcher selects which value is transferred to the output depending on the user-selected encoding option (sample-adaptive or block-adaptive). When the block-adaptive option is selected, the output of the compressor are the mapped residuals that will be subsequently encoded by the CCSDS-121 IP core, otherwise, the output are the sample-adaptive compressed codewords.
Taking into account the most common sample orderings provided by multispectral and hyperspectral sensors, three different VHDL architectures are described for the CCSDS-123 predictor, corresponding to the BIP, BSQ and BIL compression orders. The basic schematic of the predictor is shown in Figure 2-5. The input samples to be compressed are first arranged in a set of FIFOs, as determined by the compression order, in such a way that the already compressed samples become the neighbouring samples of the pixels subsequently processed. The amount of samples stored in these FIFOs depends on the compression order, as specified in [RD-7]. For the BIP and BIL architectures, the user is offered the possibility of allocating the FIFO labelled as FIFO_TOP_RIGHT in an external memory, due to the amount of memory required.

The CCSDS-123 predictor contains different blocks which represent the main operations that need to be performed. Storage elements are needed for the local differences and weight vectors. Depending on the selected architecture, the local differences vector might be stored using internal memory or a memory placed outside the FPGA. The modules performing the most computationally demanding operations are marked in red. These operations are scheduled in parallel in the architectures in which the data dependencies do not impose a strong limitation in throughput; and serialized to reduce the resource occupancy when the opposite situation takes place.



Hyperspectral data
CCSDS123 predictior: basic block diagram
FIFO TOP RIGHT
$s_{z,y-1,x+1}$
FIFO CURRENT
$s_{z,y,x+1}$
FIFO LEFT
FIFO TOP
FIFO TOP LEFT
$s_{z,y,x-1}$
$s_{z,y-1,x}$
$s_{z,y-1,x-1}$
INPUT FIFOs FOR CURRENT AND NEIGHBOURING SAMPLES
PREDICTOR
WEIGHT VECTOR STORAGE
WEIGHT UPDATE
LOCAL DIFF STORAGE
Prediction error
LOCAL DIFFERENCES
MULT & ACC
PREDICTOR
LOCAL SUM
RHO UPDATE
MAP
Mapped predition residuals (to be encoded by internal sample adaptive encoder or by the external CCSDS121 IP core)
Sample to be compressed
Neighbouring samples
Storage element external to the FPGA depending on compression order.
Weight vectors and local differences
Modules with higher complexity

Figure 2-5: Basic schematic of the CCSDS123 predictor, showing the modules with a potentially highest complexity; and the storage elements that will potentially need to be stored outside the FPGA.

2.4.2.1 BIP and BIP-mem architectures

The architecture described for BIP takes into account that, provided enough samples in the spectral dimension, the predictor has the possibility of accepting one compressed sample per clock cycle. This is hence the compression order with the highest possible throughput. The user is offered the possibility of using an external memory to store the FIFO_TOP_RIGHT (BIP-mem architecture), or using only memory that is internal to the FPGA (BIP architecture). The former communicate with the external memory using the AHB master interface. Such interface has been extended in order to manage burst transactions of configurable length, with the aim of improving the performance bottleneck in the BIP-mem architecture.

The local differences vector is stored in the internal FPGA memory. The multiply and accumulation operations needed for the computation of the dot product of the local differences and weight vectors are performed using the structure depicted in Figure 2-6, which makes it possible to calculate and obtain a dot product result every clock cycle. The weight vectors are updated in parallel using instances of weight update units as shown in Figure 2-7. The amount of instances of multipliers and accumulators used for the dot product and the amount of weight vector units are calculated based on the maximum P value selected by the user by the compile-time parameter P_{MAX} .

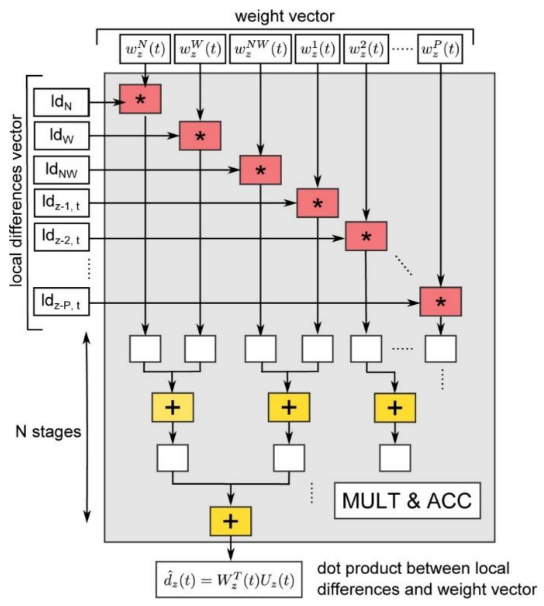


Figure 2-6: Multiply and accumulate structure to perform the dot product in BIP and BIL.

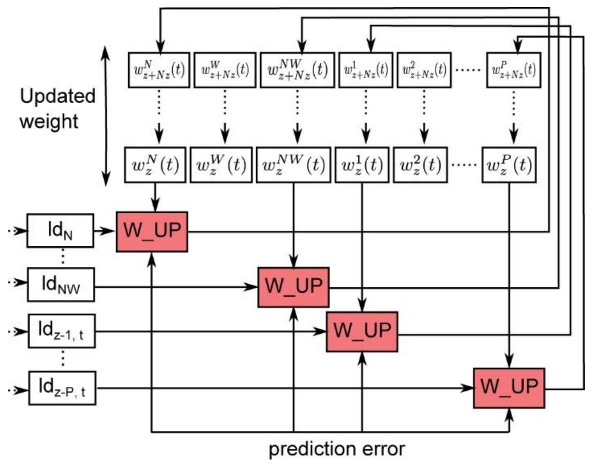


Figure 2-7: Weight vector update in BIP and BIL

2.4.2.2 BSQ architecture

The main differences between the BSQ and the BIP architecture lies in the allocation of the local differences vector in an external memory, and the scheduling of the multiply and accumulate operations, which are performed serially.

In BSQ, it is necessary to store a complete vector of local differences per sample during the compression of a band. This amount of storage makes it necessary to place the local differences values outside the FPGA. The AHB master interface is used to transfer the samples to and from an external memory. The memory addresses are calculated by the IP core as shown in Figure 2-8, in such a way that the memory locations are appropriately reused when available. One local difference needs to be stored per sample, and P values need to be read.

The data dependencies in BSQ place an important limitation in the throughput. Considering that the local differences vectors need to be retrieved from the external memory, we observed that there is no clear advantage in having the multiply and accumulate operations and the weight vector update performed with the structures depicted in Figure 2-6 and Figure 2-7. Conversely, this operation and the weight vector update are serialized in order to reduce the resource utilization. The rest of the predictor uses the same components as the BIP architecture.

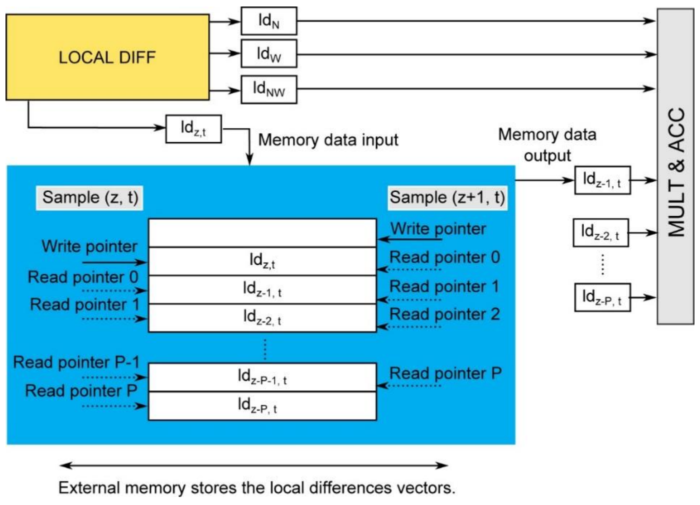


Figure 2-8: Storage of local differences vectors for each sample in a band in an external memory for BSQ order.

2.4.2.3 BIL and BIL-mem architectures

Finally, BIL inherits most of the components from the BIP architecture. The main difference resides in the local difference vector storage. In BIL, it is necessary to store one vector per sample in a line of pixels. This storage is placed inside the FPGA. The structures to compute the dot product and weight update operations are the sample as in BIP.

A specific scheduling is devised for BIL in order to ensure that the maximum possible throughput is achieved in both situations, when compressing the samples in a line, where we find the same data dependencies as in BSQ, and when compressing the last sample of a line and the first of the next line. In the latter, the data dependencies we find are the same as in BIP.

Similarly to the BIP-mem architecture, a variation of the BIL architecture which uses an external memory to store some intermediate samples has been designed. This new architecture, named BIL-mem, includes an AHB master interface which supports burst transactions of configurable length, in order to communicate with the external memory.

2.4.3 The full SHyLoC-e (CCSDS-121 + CCSDS-123 IP cores)

The CCSDS-121 and CCSDS-123 IP cores have been designed in such a way that they can be combined. In this scenario, the CCSDS-121 block coder acts as an external entropy coder of the CCSDS-123 predictor. In order to do so, both IP cores have compatible interfaces, as shown in Figure 2-9. When the IPs are working together, the output data interface of the CCSDS-123 IP core is connected to the input data interface of the CCSDS-121 block coder. For this configuration, the CCSDS-123 IP core includes an additional control interface that is meant to be connected to the control interface of the CCSDS-121 block coder. Moreover, in order to transfer the header values from the CCSDS-123 IP to the CCSDS-121 IP, signals are included to mark the output data as a header and to specify the amount of valid bits in that case.

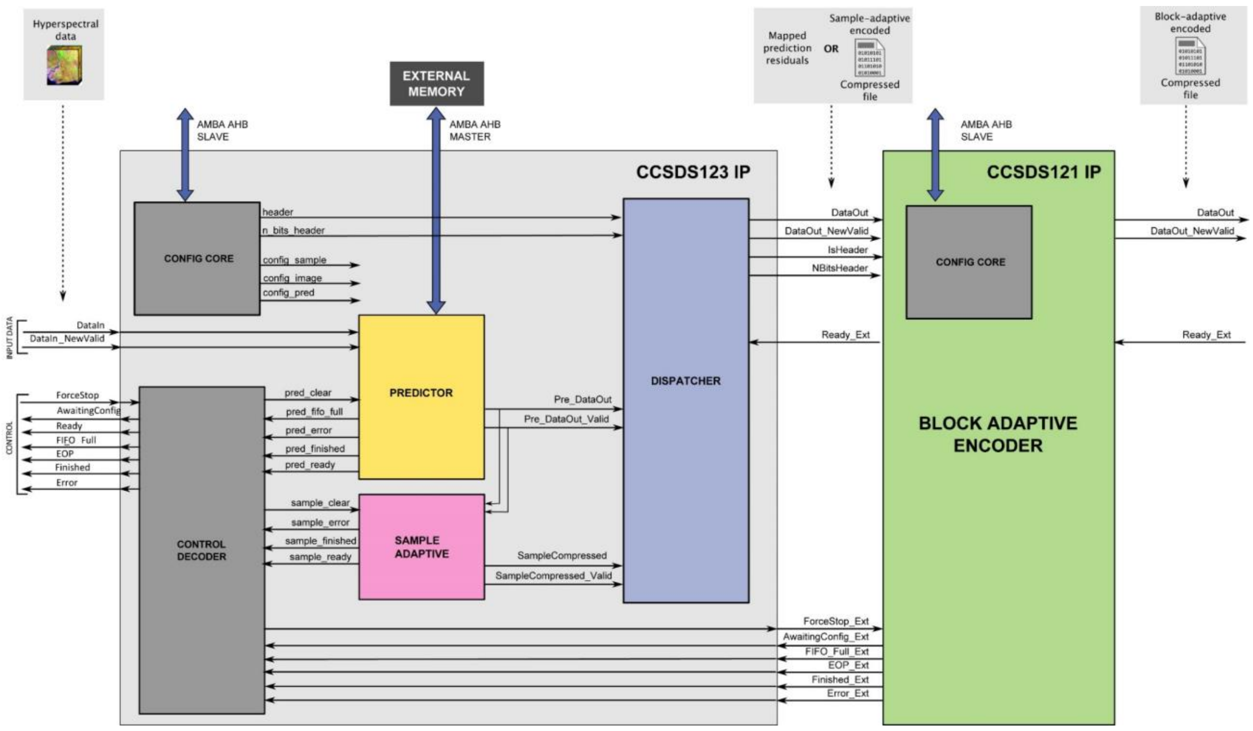


Figure 2-9: Basic SHyLoC-e IP cores: interfaces, main blocks and connection between them.

2.5 Verification and validation

The CCSDS-121 and CCSDS-123 IP cores have been extensively verified against reference software tools from ESA [RD-5], proving that, provided the same inputs, the IP cores and software counterparts produce the same exact compressed files.

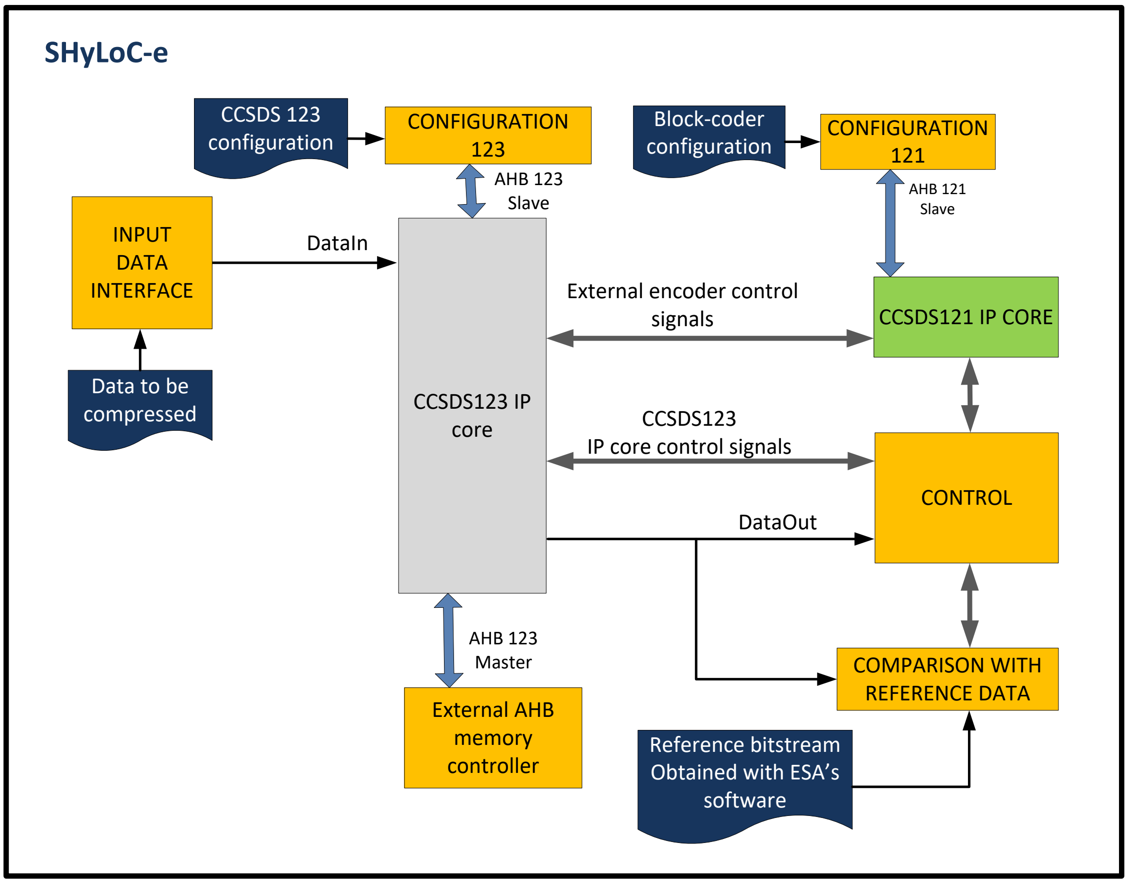


Figure 2-10: SHyLoC-e testbench and IP cores under verification

In particular, the verification dataset of the CCSDS121 IP combines 29 real and synthetic hyperspectral images with different number of samples, dynamic range and sample order; 10 sets of compile-time configuration parameters and 14 sets of run-time configuration parameters. In total 86 simulations have been performed. All simulations have been run with QuestaSim and automated with scripts.

With respect to the verification of the CCSDS123 IP, the dataset consists of 18 multispectral and hyperspectral images from different sensors, as well as some synthetic images. The set covers a broad range of image sizes and different data representations and samples ordering. In addition, 22 sets of configuration parameters have been defined, 11 in compile-time and other 11 in run-time, including some configurations which make use of the CCSDS121 block coder. In total, 108 tests have been designed, which have been run with QuestaSim and automated with scripts.

2.6 Technology mapping

The designed IP cores were described in VHDL and they are technology independent, i.e. no technology specific macrocells are instantiated. They have been successfully mapped to the following FPGA technologies:

- Xilinx Virtex 5 & 5QR
- Microsemi ProASIC3E, RTAX4000 and RTG4

- NanoXplore NG-MEDIUM and NG-LARGE

In the following sections, we present a summary of the synthesis results obtained for the Xilinx Virtex5 FX130 FPGA, Microsemi RTG4 150 and NanoXplore NG-MEDIUM. The complete technology mapping results can be consulted in the IP core's Datasheet [RD-7].

2.6.1 Synthesis results of the CCSDS-121 IP core

We have performed the synthesis by selecting a set of baseline compile-time parameters. In particular, the number of block in a samples, *J* is set to 16, the dynamic range of the input samples *D* is set to 32, and the size of the output buffer is set to 64 bits. Besides, the unit-delay predictor is included. Run-time configuration is enabled. Results can be observed in Table 2-1, Table 2-2 and Table 2-3.

TABLE 2-1: CCSDS121- SYNTHESIS RESULTS ON VIRTEX5 FX130

Resources	Used
Block RAM	0 (298)
DSP48s	5 (320)
Registers	2291 (81920)
LUTs	7660 (81920)
Max. Frequency (MHz)	81.4

TABLE 2-2: CCSDS121- SYNTHESIS RESULTS ON RTG4 150

Resources	Used
Carry Cells	2865 (151824)
Sequential Cells	1918 (151824)
Block RAM (RAM64x18_RT)	19 (209)
DSP Blocks	5 (462)
LUTs	11055 (151824)
Maximum Frequency (MHz)	46.0

TABLE 2-3: CCSDS121- SYNTHESIS RESULTS ON NANOXPLORE NG-MEDIUM

Resources	Used
Carry Cells	5660 (8064)
Block RAM (48Kb)	20 (56)
DSP48s	2 (112)
Registers	2670 (32256)
LUTs	10309 (32256)
Max. Frequency (MHz)	21.9

The results show the feasibility of the implementation in both FPGA targets. The throughput reaches 81.4 MSamples per second for the Virtex5 FPGA. The core is able to accept one sample per clock cycle after it is configured.

2.6.2 Synthesis results of the CCSDS-123 IP core

We have performed the synthesis of the CCSDS-123 IP core when configured for the compression of the images in Table 2-4. In these cases, the runtime configuration is disabled, and the IP core is tailored to a set of compile-time configuration values and the specific spatial and spectral dimension of the image to be compressed. Additionally, we have performed the synthesis of a version of the IP core that accepts runtime configuration of the compression parameters. In this case, we set the maximum size of the image that can be compressed to $N_x = 512, N_y = 1024, N_z = 256$. The P value is set to 3 in all experiments.

Synthesis results on the Virtex5 FX130, on the Microsemi RTG4 150 and on the NanoXplore NG-MEDIUM are shown in Table 2-5, Table 2-6 and Table 2-7, respectively.

TABLE 2-4: CCSDS123 IP – ACQUISITION SCENARIOS FOR MAPPING

IMAGE	EN_RUNCFG	Nx_GEN	Ny_GEN	Nz_GEN	D_GEN
MULTISPECTRAL (LANDSAT)	0	1024	1024	6	8
HYPERSPECTRAL (AVIRIS)	0	512	680	224	16
ULTRASPECTRAL (AIRS)	0	90	135	1501	14
RUNTIME CONFIG	1	512 (512)*	680 (1024)*	224 (256)*	16

(*) The depth of all FIFOs in the design is constrained to a power of two.

TABLE 2-5: CCSDS123 IP - IMPLEMENTATION PERFORMANCE ON VIRTEX5 (XC5VFX130T)

Parameters	Total Resources	MULTISPECTRAL (LANDSAT)				
		BIP	BIP-MEM	BSQ	BIL	BIL-MEM
I/O	840	317	317	317	317	317
BUFGs	32	1	2	2	1	2
Block RAMs	298	2	0	1	5	3
DSP48	320	8	7	5	7	7
Registers	81920	2309	3116	2900	2703	3497
LUTs	81920	3651	4269	4634	4218	5185
Maximum Frequency (Clk_AHB) (MHz)			156.1	128.3		119.2
Maximum Frequency (Clk_S) (MHz)		111.9	136.1	106.0	126.3	113.7
Parameters	Total Resources	HYPERSPECTRAL (AVIRIS)				
		BIP	BIP-MEM	BSQ	BIL	BIL-MEM
I/O	840	325	325	325	325	325
BUFGs	32	1	2	2	1	2
Block RAMs	298	74	10	1	74	10
DSP48	320	10	10	5	10	10

Registers	81920	2902	3326	3277	3049	3902
LUTs	81920	4707	5332	5778	5248	6257
Maximum Frequency (Clk_AHB) (MHz)			161.6	129.2		125.0
Maximum Frequency (Clk_S) (MHz)		113.7	113.2	112.9	113.7	123.2
Parameters	**Total Resources**	**ULTRASPECTRAL (AIRS)**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
I/O	840	323	323	323	323	323
BUFGs	32	1	2	2	1	2
Block RAMs	298	123	11	1	123	11
DSP48	320	8	10	6	10	10
Registers	81920	2781	3256	3238	2918	3783
LUTs	81920	4600	4967	5409	4925	5897
Maximum Frequency (Clk_AHB) (MHz)			164.8	139.2		119.5
Maximum Frequency (Clk_S) (MHz)		110.7	112.7	110.9	114.5	112.7
Parameters	**Total Resources**	**RUNTIME CONFIG**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
I/O	840	325	325	325	325	325
BUFGs	32	2	2	2	2	2
Block RAMs	298	74	10	1	74	10
DSP48	320	10	14	9	12	15
Registers	81920	3695	4223	4145	3843	4838
LUTs	81920	5918	6325	6768	6350	7330
Maximum Frequency (Clk_AHB) (MHz)		231.4	156.7	57.0	209.0	118.2
Maximum Frequency (Clk_S) (MHz)		112.4	111.6	79.3	104.4	104.7

TABLE 2-6: CCSDS123 IP - IMPLEMENTATION PERFORMANCE ON RTG4 (RTG4 150)

Parameters	**Total Resources**	**MULTISPECTRAL (LANDSAT)**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
IO Cells	720	190	212	212	190	212
Carry Cells	151824	2017	228	2489	2179	2420
Sequential Cells	151824	2485	3081	2800	2755	3413
Block RAMs (1Kx18 + 64x18)	209	4 + 31	0 + 33	1 + 25	10 + 38	6+40

DSP Blocks	462	7	7	7	7	7
LUTs	151824	5317	6198	6276	5639	6983
Maximum Frequency (Clk_AHB) (MHz)			101.5	81.7		72.0
Maximum Frequency (Clk_S) (MHz)		72.8	83.6	80.4	74.2	69.6
Parameters	**Total Resources**	**HYPERSPECTRAL (AVIRIS)**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
IO Cells	720	198	220	236	198	220
Carry Cells	151824	3604	3299	3044	3764	3376
Sequential Cells	151824	3052	3713	3177	3454	4128
Block RAMs (1Kx18 + 64x18)	209 + 210	129 + 62	1 + 64	1 + 36	135 + 65	7+67
DSP Blocks	462	13	13	11	13	13
LUTs	151824	7935	7985	7382	8175	8743
Maximum Frequency (Clk_AHB) (MHz)			95.9	85.2		76.3
Maximum Frequency (Clk_S) (MHz)		64.6	79.9	80.8	56.6	76.2
Parameters	**Total Resources**	**ULTRASPECTRAL (AIRS)**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
IO Cells	720	196	218	234	196	218
Carry Cells	151824	5092	4162	2954	4125	3290
Sequential Cells	151824	2994	3615	3079	3172	3844
Block RAMs (1Kx18 + 64x18)	209 + 210	266+212	10+214	0 + 30	275 + 30	19+32
DSP Blocks	462	7	7	6	7	7
LUTs	151824	9661	8878	7182	8774	8278
Maximum Frequency (Clk_AHB) (MHz)			84.8	79.5		82.5
Maximum Frequency (Clk_S) (MHz)		69.2	69.4	79.5	69.3	76.6
Parameters	**Total Resources**	**RUNTIME CONFIG**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
IO Cells	720	252	272	288	252	272
Carry Cells	151824	3960	3665	3447	4078	3756
Sequential Cells	151824	3641	4397	3789	3977	4787
Block RAMs (1Kx18 + 64x18)	209 + 210	129 + 64	1 + 66	1 + 38	135 + 67	7+69
DSP Blocks	462	16	16	14	16	16

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada

LUTs	151824	10027	10268	9648	10293	10877
Maximum Frequency (Clk_AHB) (MHz)		132.8	96.3	85.3	132.8	76.3
Maximum Frequency (Clk_S) (MHz)		66.1	79.6	74.2	73.9	74.2

TABLE 2-7: CCSDS123 IP - IMPLEMENTATION PERFORMANCE ON NG-MEDIUM NX1H35S

Parameters	**Total Resources**	**MULTISPECTRAL (LANDSAT)**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
Carry cells	8064	2360	2778	2936	2743	3114
Registers	32256	2687	3417	3041	3083	3816
Block RAMs (48Kb)	56	33	34	27	43	44
DSP Blocks	112	6	6	4	6	6
LUTs	32256	3526	4922	4300	3806	5094
Maximum Frequency (Clk_AHB) (MHz)			31.9	39.5		31.5
Maximum Frequency (Clk_S) (MHz)		41.1	37.2	40.6	39.1	39.0
Parameters	**Total Resources**	**HYPERSPECTRAL (AVIRIS)**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
Carry cells	8064	3353	3834	3646	3678	4128
Registers	32256	3148	3895	3308	3542	4291
Block RAMs (48Kb)	56	95	34	27	105	44
DSP Blocks	112	6	6	4	6	6
LUTs	32256	4263	5636	4896	4663	5866
Maximum Frequency (Clk_AHB) (MHz)			35.1	50.6		29.2
Maximum Frequency (Clk_S) (MHz)			40.1	38.0		33.3
Parameters	**Total Resources**	**ULTRASPECTRAL (AIRS)**				
		BIP	**BIP-MEM**	**BSQ**	**BIL**	**BIL-MEM**
Carry cells	8064	3333	3857	3562	3634	4126
Registers	32256	3170	3906	3240	3507	4246
Block RAMs (48Kb)	56	143	34	27	153	44
DSP Blocks	112	6	6	4	6	6
LUTs	32256	4118	5664	4815	4381	5735
Maximum Frequency (Clk_AHB) (MHz)			30.8	42.0		25.2
Maximum Frequency (Clk_S) (MHz)			37.3	40.6		31.2

Parameters	Total Resources	RUNTIME CONFIG				
		BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Carry cells	8064	3855	4440	4249	4198	4729
Registers	32256	3665	4518	3897	4023	4888
Block RAMs (48Kb)	56	97	36	29	107	46
DSP Blocks	112	8	8	6	8	8
LUTs	32256	6004	7251	6544	6221	7517
Maximum Frequency (Clk_AHB) (MHz)			35.1	36.5		28.2
Maximum Frequency (Clk_S) (MHz)			31.7	32.8		27.5

The results show how the requirements vary depending on the selected architecture and the selected image size. All the configurations tested can be implemented on the Virtex5 FPGA. The maximum throughput (136.1 MSamples per second) is achieved for the BIP-Mem architecture, which is capable of compressing one sample every clock cycle. With this result, we can conclude that the existing penalty in terms of throughput in the original SHyLoC when the BIP-Mem architecture is implemented has been solved. In general terms, the BSQ architecture has the lowest resource usage and also the lowest throughput, since some operations have been scheduled serially. Finally, BIL shows the same resource usage as BIP, but with a lower throughput due to the data dependencies present when compressing the samples in a line. DSP usage is low for all configurations, due to the limited amount of complex mathematical operations present in the standard.

The results are very similar for the RTG4. However, we observe that for the AIRS image, the amount of Block RAM needed exceeds what is available on the FPGA for the BIP and BIL architectures. We note the high amount of bands present in AIRS, 1501 and that the depth of all FIFOs in the design is set to be a power of two. This issue might be overcome in the future with optimizations and technology mapping, using technology-specific cells. Besides, we hightlight that throughput results for BIL order have been improved introducing the BIL-Mem architecture that uses an external memory interface, reducing too the usage of Block RAM and make the implementation feasible also for bigger images.

Finally, the results for NG-MEDIUM are worse in comparison with the other two FPGA devices because it is still an immature technology. In this case, the memory resources are a critical feature, making unfeasible to implement the BIP and BIL architectures for any of the images used for the mapping step. However, for BIP-Mem and BIL-Mem architectures we consider throughput results correct, taking into account the timing features of other implementations with similar complexity over this technology, available in the state-of-the-art.

2.7 Conclusions

In this work, we have presented the VHDL description of the modifications applied to two IP cores, which perform lossless compression as specified by the CCSDS-121 and CCSDS-123 standards, in order to include additional functionality and to have better performance in comparison with the original version. The cores might work independently as well as jointly, offering simple plug-and-play compatible interfaces.

The CCSDS-121 and CCSDS-123 IP cores are technology independent and configurable at compile-time and runtime. The compile-time parameters make it possible to tailor the IP cores in order to reduce their complexity if needed, in order to fit in a specific FPGA target.

The IP cores have been successfully mapped to 7 FPGA targets: Xilinx Virtex 5 & 5QR; Microsemi ProASIC3E, RTAX4000 and RTG4; and NanoXplore NG-MEDIUM and NG-LARGE.

Synthesis results have been presented for the Virtex5 FX130, RTG4 150 FPGA and NanoXplore NG-MEDIUM, with a maximum compression throughput of 140 MSamples per second. The cores exhibit a low LUTs and DSPs usage, showing a maximum occupancy of 7% of the LUTs of the Virtex5 and 13% of the RTG.