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

IUMA/1410/AO8032

D8 – Summary Report

By

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

13 March 2018

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

DISCLAIMER

The work associated with this report has been carried out in accordance with the highest technical standards and TRPAO8032 partners have endeavoured to achieve the degree of accuracy and reliability appropriate to the work in question. However, since the partners have no control over the use to which the information contained within the report is to be put by any other party, any other such party shall be deemed to have satisfied itself as to the suitability and reliability of the information in relation to any particular use, purpose or application.

Under no circumstances will any of the partners, their servants, employees or agents accept any liability whatsoever arising out of any error or inaccuracy contained in this report (or any further consolidation, summary, publication or dissemination of the information contained within this report) and/or the connected work and disclaim all liability for any loss, damage, expenses, claims or infringement of third party rights.

DOCUMENT HISTORY

Date	Version	Author	Description	Status
25-05-2017	1.0	Lucana Santos	Summary report	Ongoing
25-05-2017	1.1	Lucana Santos	Reduced contents	Ongoing
25-05-2017	1.2	Lucana Santos	Format problems	Ongoing

LIST OF AUTHORS

Partner	Authors
IUMA	Lucana Santos

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 (Summary Report) of the ESA Contract No. 4000113182/15/NL/LF entitled CCSDS Lossless Compression IP-Core Space applications.

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, May 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 ITT AO/1-8032/14/NL/AK, CCSDS Lossless Compression IP-Core Space Applications, Proposal, IUMA-TELETEL, October 2014

[AD-5] ECSS-Q-ST-60-02C- "ASIC and FPGA development": http://www.ecss.nl

[AD-6] TRP AO8032 – Deliverable D6, Verification and Validation Plan, IUMA, November 2016

[AD-7] TRP AO8032 – Deliverable D4, Requirements Specifications, IUMA, November 2016

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 recommmended standard CCSDS 123.0-B-1, The Consultative Committee for Space Data Systems, 2012.

[RD-3] "IEEE standard for standard SystemC language reference manual," IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005), pp. 1–638, Jan 2012.

[RD-4] A. Gómez, L. Santos, R. Sarmiento, and L. Fossati, "Modelling in SystemC the CCSDS standard for satellite multispectral and hyperspectral data compression," in Proceedings of 2016 ESA workshop on Onboard Payload Data Compression (OBPDC), 2016

[RD-5] Santos, L., Gómez, A., Hernández-Fernández, P. and Sarmiento, R., 2016, October. "SystemC modelling of lossless compression IP cores for space applications". Design and Architectures for Signal and Image Processing (DASIP), 2016 Conference on (pp. 65-72). IEEE.

[RD-6] Appendix 1 to ESA ITT AO/1-8032/14/NL/AK, Statement of Work, CCSDS Lossless Compression IP-Cores for Space Applications, European Space Agency, 2014.

[RD-7] European Space Agency. (2016) ESA IP Cores. [Online]. Available: http://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/About_ESA_IP_Cores

[RD-8] 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-9] 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-10] TRP AO8032 – Deliverable D10, IP cores Datasheet, IUMA, July 2017

[RD-11] L. Santos, A. Gómez, R. Sarmiento, L. Fossati and D. Merodio, "," in Proceedings of 2016 ESA workshop on Onboard Payload Data Compression (OBPDC), 2016

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 exact original data can be 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 focuses on defining 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) and consists 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 (bock-adaptive encoder).

Despite the availability of algorithms, efficiently computing the compression in the technologies available on a satellite is an important challenge, due to the limitations imposed by the space environment. Most of the times, the algorithms are implemented on FPGAs or ASICs in order to optimize their performance and make them low power. 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 performs 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 the various trade-offs between data reduction and potential complexity that arise from the selection of the compression parameters. It is well known that hardware architecture exploration may require numerous iterations and therefore is not practical from the VHDL description of the IP cores. In this sense, Electronic System Leven (ESL) [RD-2] arises as an advantageous paradigm in order to generate, in a relatively short time, the specifications that lead to efficient implementations of the CCSDS-123 and CCSDS-121 algorithms on the FPGA targets that are currently interesting in the space sector. ESL makes it possible to perform design optimization at an early stage quickly and cost-effective, as well as hardware validation against software before RTL is implemented. Therefore, as part of this work, the CCSDS-123 and CCSDS-121 IP cores are modelled in SystemC [RD-4][RD-5], before they are described in VHDL, exploring the design space and considering the impact in throughput and complexity of the user-defined parameters allowed by the algorithms.

2.2 Overview of the CCSDS lossless compression algorithms

The CCSDS-123 standard [RD-2] describes a lossless compressor 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 a previous standard issued by the CCSDS, namely the CCSDS-121 [RD-1]. The input of the compressor 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 recovered exactly.

2.2.1 Prediction

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 which can range from 0, for which no information from previous bands is utilized, up to 15.

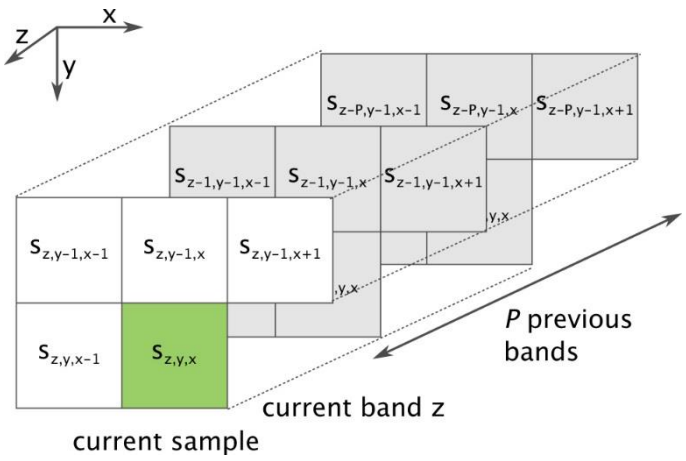

Figure 2-1: Three-dimensional neighbourhood used for prediction in the CCSDS 123 standard.

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 counter are causally 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.

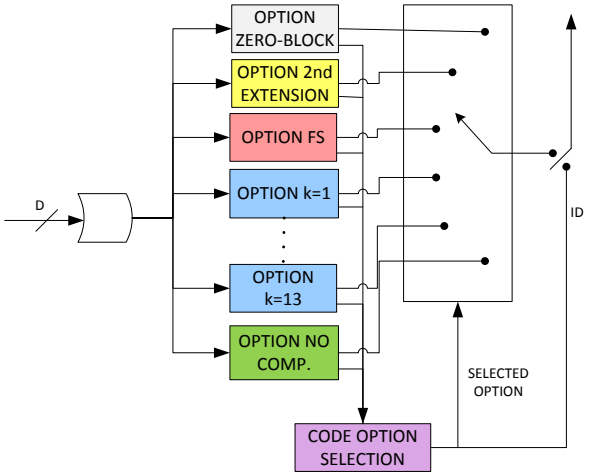

Figure 2-2: Block-adaptive entropy coder concept.

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: TRP project objectives and requirements

The work presented in this report follows the statement for work established by the European Space Agency in the scope of the commissioned Technology Research Project (TRP) entitled: "CCSDS IP cores for space applications" [RD-6]. The main requirements and objectives are summarized hereafter.

The main goal of this TRP project is to design a two new reusable IP cores compliant with the CCSDS-123 and CCSDS-121 standards that will be part of ESA's IP cores library [RD-7]. Since the two IP cores may work jointly, we devise a compression system called SHyLoC, which is presented in Figure 2-3. The CCSDS-123 and CCSDS-121 IPs have been designed to have compatible interfaces so that they can work separately as well as jointly, with the CCSDS-121 performing the entropy coding of the CCSDS-123, as established by the standards.

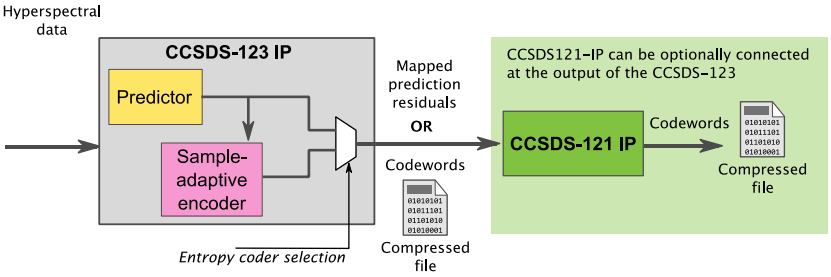

Figure 2-3: SHyLoC concept: separate IP cores with compatible interfaces that can be combined into a single compression entity.

In order to achieve real-time performance, SHyLoC shall be able to receive and compress samples at a rate of 1 Gbps. Additionally, it shall be capable of accepting in any of the possible compression orders: band-sequential (BSQ), band-interleaved per pixel (BIP) or band-interleaved per line (BIL). The compression parameters shall be selectable at compile-time as well as at run-time, in such a way that the configuration can be changed without re-synthesizing the core if it is desired. For this purpose, a slave AMBA AHB interface shall be provided to enable access to the IP cores memory mapped registers. Finally, SHyLoC shall be synthesizable in FPGA devices representative of space-grade hardware.

An example of the final FPGA instantiation of the IP cores is depicted in Figure 2-4. In this system, the samples are received directly from the sensor to the FPGA that is configured with the LEON2FT processor and the IP cores. Additionally, a memory element is included, that may be used to store the intermediate values of the compression.

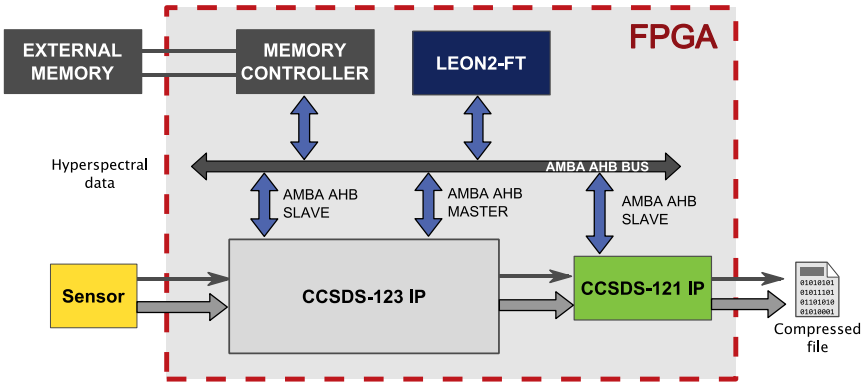

Figure 2-4: Possible final implementation of SHyLoC in an embedded system that includes a LEON processor and a memory element.

2.4 SHyLoC SystemC model

Previously to the description of the IP cores in VHDL, we perform the modelling of SHyLoC in SystemC [RD-4][RD-5], in order to ensure that the implementation of the IP cores meets the specifications and requirements derived from the aforementioned TRP activity. The SystemC models are described in a way that allows exploring possible HW/SW co-design solutions in heterogeneous platforms, and serve to design the hardware architecture, showing the influence of the compression parameters and the compression order in terms of storage requirements, complexity of the mathematical operations, data

dependencies that can set boundaries in throughput, etc. The IP cores described in SystemC exhibit the same interfaces and behaviour as the VHDL counterparts.

The SystemC models of the CCSDS-121 and CCSDS-123 IP include both ad-hoc interfaces -used to receive the samples to be compressed and output the compressed files- and standard AMBA AHB interfaces -to receive the runtime configuration parameters-. The CCSDS-123 IP includes additional control and data signals for the easy plug-and play connection of the CCSDS-121 compressor model.

2.4.1 CCSDS-121 SystemC model

The SystemC description of the CCSDS-121 IP cores includes the modules displayed in Figure 2-5 and listed and described below:

- *ahb slave config121*: implements the necessary TLM functions to read the configuration registers from the AMBA AHB Slave. It validates the configuration and transfers it to the interface module.
- *interface*: receives the configuration sends it to the rest of the modules.
- *io interface*: receives the configuration and the input data, generates the control signals and prepares the output data.
- *compute_len*: computes the length of all possible encoding options for each received block.
- *zero_coder*: counts the amount of zero blocks and encodes them.
- *select_option*: selects the encoding option that yields the shortest codeword.
- *fs_coder*: encodes the block of samples with the selected option.
- *packer*: fits the codeword in the proper words (with the same size of the output buffer).
- *header_gen*: generates the header.
- *block_counter*: simple counter of blocks received.

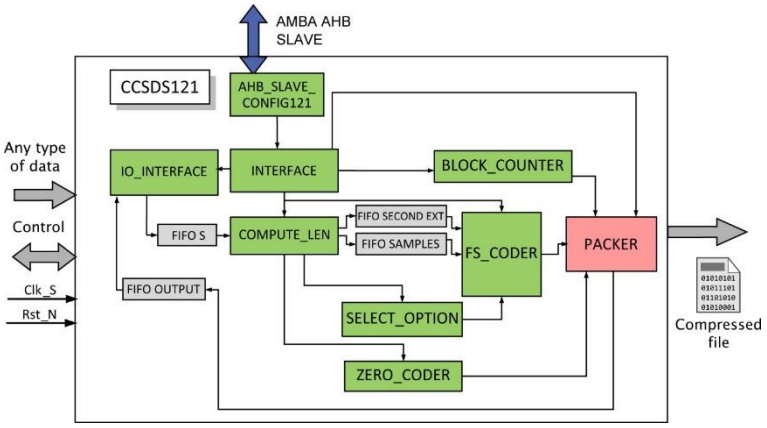

Figure 2-5: Modules of the CCSDS-121 SystemC model.

Additionally, memory elements have been included: a FIFO to store the input samples, the necessary FIFOs to store intermediate values while samples are being compressed; and one last FIFO for the output bitstream.

2.4.2 CCSDS-123 SystemC model

The components of the SystemC description are shown in Figure 2-1, and described below:

- *ahb slave config123*: implements the functions to read and validate the configuration registers from the AMBA AHB Slave.
- *interface*: receives AMBA configuration parameters and prepares the input samples to be compressed.
- *opcode*: manages the input FIFOs that store the received samples and neighbours, arranging them according to the selected compression order (BSQ, BIP or BIL).
- *localsum*: computes the local sum of the neighbouring samples.
- *localdiff*: creates and stores the local differences vector.
- *predictor*: computes the prediction performing the dot product of the localdiff and weight vectors.
- *updated_weights*: initializes and updates the weight vector from the local differences and the prediction error.
- *map*: maps the prediction residuals.
- *sample-adaptive*: encodes the mapped prediction residuals.
- *packer*: packs output bitstream according to the bit width of the output buffer.

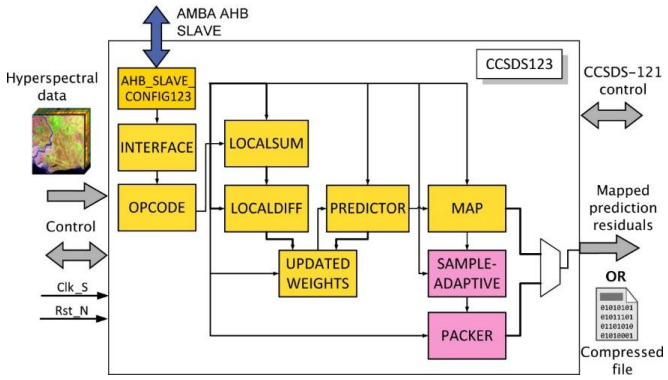

Figure 2-6: Modules of the CCSDS-123 SystemC model.

2.4.3 Conclusions extracted from the SystemC models

Once it was proven that, provided the same inputs, the SystemC models produced the same exact compressed files than the software implementations from ESA [RD-8] and UAB [RD-9], different experiments have been carried out with the SystemC model. The main purpose of these experiments is to obtain a first approximation to the final requirements for a future RTL implementation of the compression IPs in terms of metrics like complexity, storage and throughput. These experiments involve various hyperspectral images in the three possible orders: BSQ, BIL and BIP; and several combinations of compile-time and run-time parameters.

Regarding the CCSDS-121 IP core, the simulations confirmed that, in principle, there are no data dependencies that limit its possible throughput; hence, a carefully designed pipeline should be able to make it possible to accept a sample to be compressed every clock cycle.

On the other hand, for the CCSDS-123 we were able to observe the impact on complexity of parameter P. We also concluded that it does not make sense to try to optimize the designed hardware architecture for any possible P value, which can range between 0 and 15. Conversely, it is more reasonable to offer the user the possibility to choose, with a compile-time configuration parameter, the maximum selectable P value, to reduce the complexity of the implementation.

The SystemC description revealed that real-time compression cannot be achieved with a common hardware architecture that supports run-time selection of the compression order. Although in BIP order it was possible for the CCSDS-123 compressor to accept one sample per clock cycle, in BSQ and BIL order, the existing data dependencies forced the rate of input samples to be lower. We concluded that different architectures need to be devised for each compression order, BSQ, BIP and BIL. The user might select the desired order and hence the architecture to be implemented at compile-time.

2.5 SHyLoC VHDL description

SHyLoC consists of two IP cores: the CCSDS-121 and the CCSDS-123, which are described in VHDL. They have compatible interfaces, as shown in Figure 2-7, comprising input and output data interfaces, an AHB slave interface for configuration and a control interface to inform about the status of the compression. 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. The CCSDS-123 IP core includes an additional control interface that is meant to be connected to the control interface of the CCSDS-121 IP core. Moreover, in order to appropriately 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. 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.

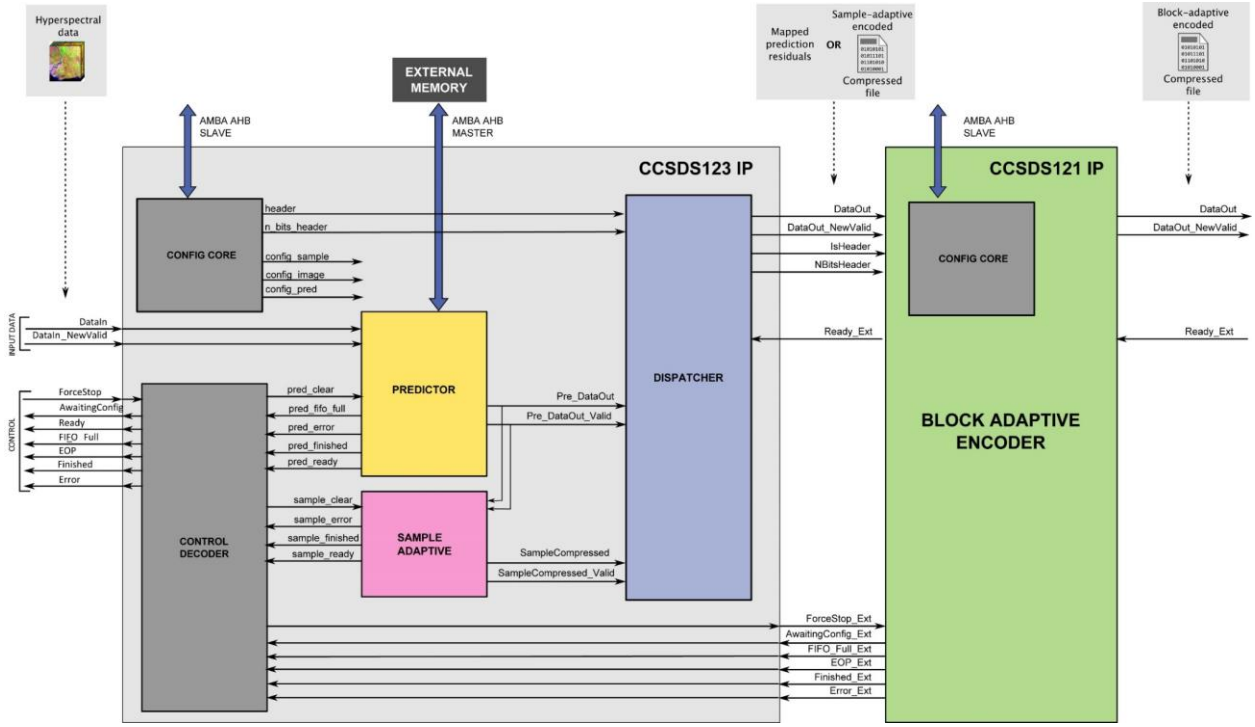

Figure 2-7: Basic SHyLoC IP cores: interfaces, main blocks and connection between them.

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.

2.5.1 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 the encoding of blocks of samples. 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 the option is known, 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.

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.

2.5.2 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.

As concluded by the design space exploration performed with the SystemC model, 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 2-8. 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 it was shown in Table 1. For the BIP architecture, 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.

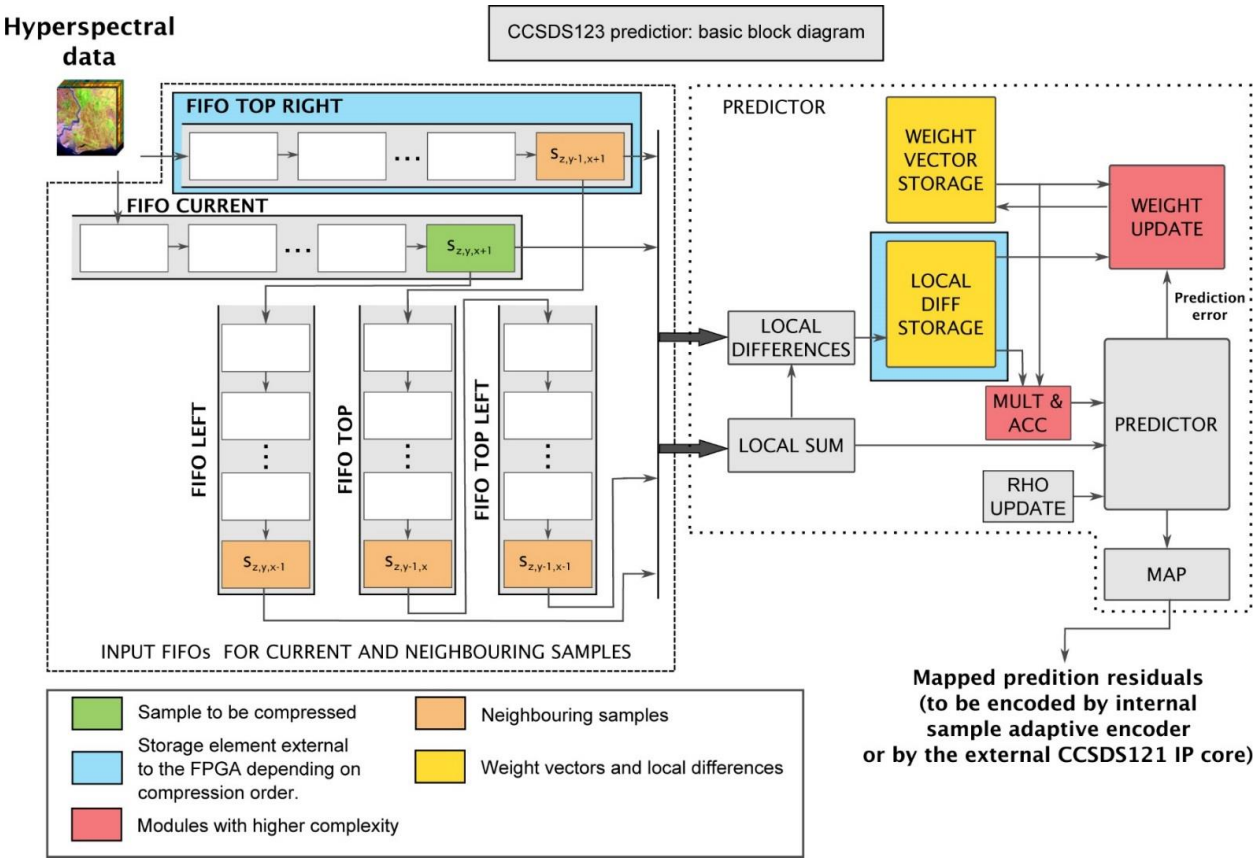

Figure 2-8: Basic schematic of the CCSDS123 predictor, showing the modules with a potentially highest complexity; and the storage elements that will potentially need to be stored outside the FPGA.

2.5.2.1 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.

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 2-9, 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 2-10. 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} .

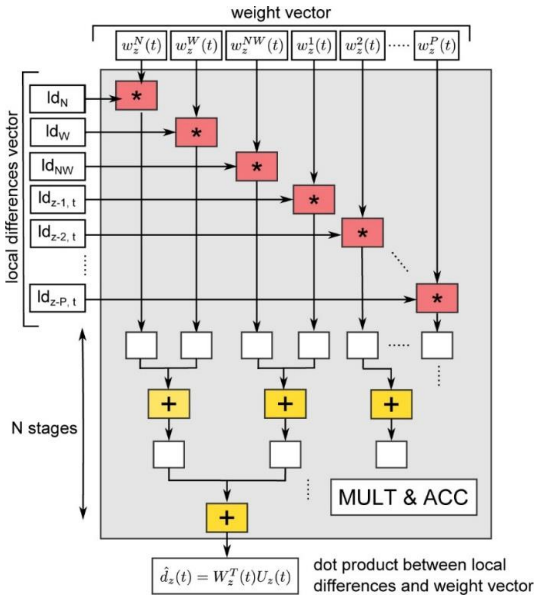

Figure 2-9: Multiply and accumulate structure to perform the dot product in BIP and BIL.

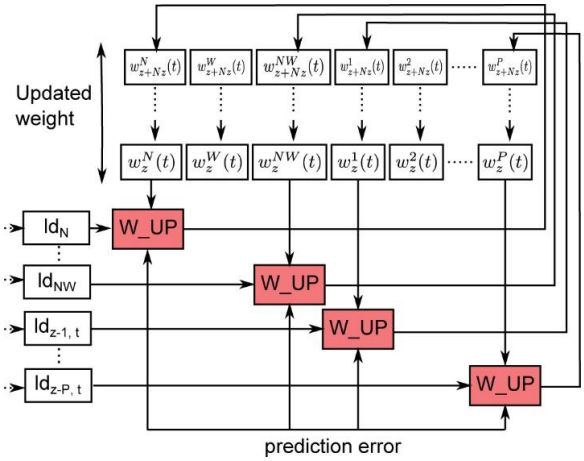

Figure 2-10: Weight vector update in BIP and BIL

2.5.2.2 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 2-11, in 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 2-9 and Figure 2-10. 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.

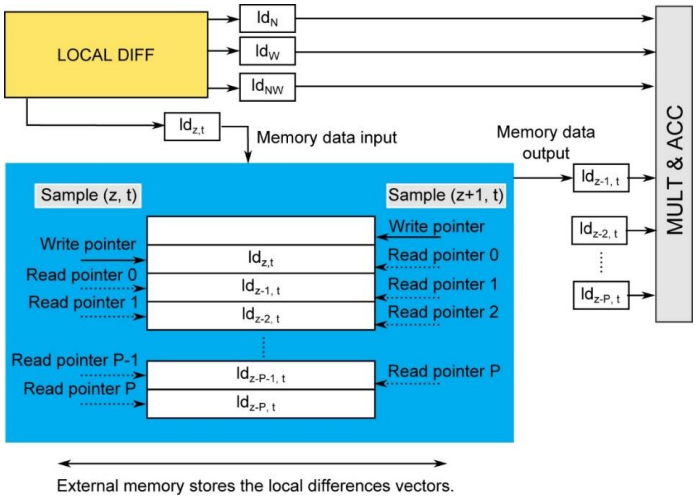

Figure 2-11: Storage of local differences vectors for each sample in a band in an external memory for BSQ order.

2.5.2.3 BIL architecture

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.

2.6 Verification and validation

The CCSDS-121 and CCSDS-123 IP cores have been extensively verified reference software implementations from ESA [RD-8] and UAB [RD-9], proving that, provided the same inputs, the IP cores and software counterparts produce the same exact compressed files.

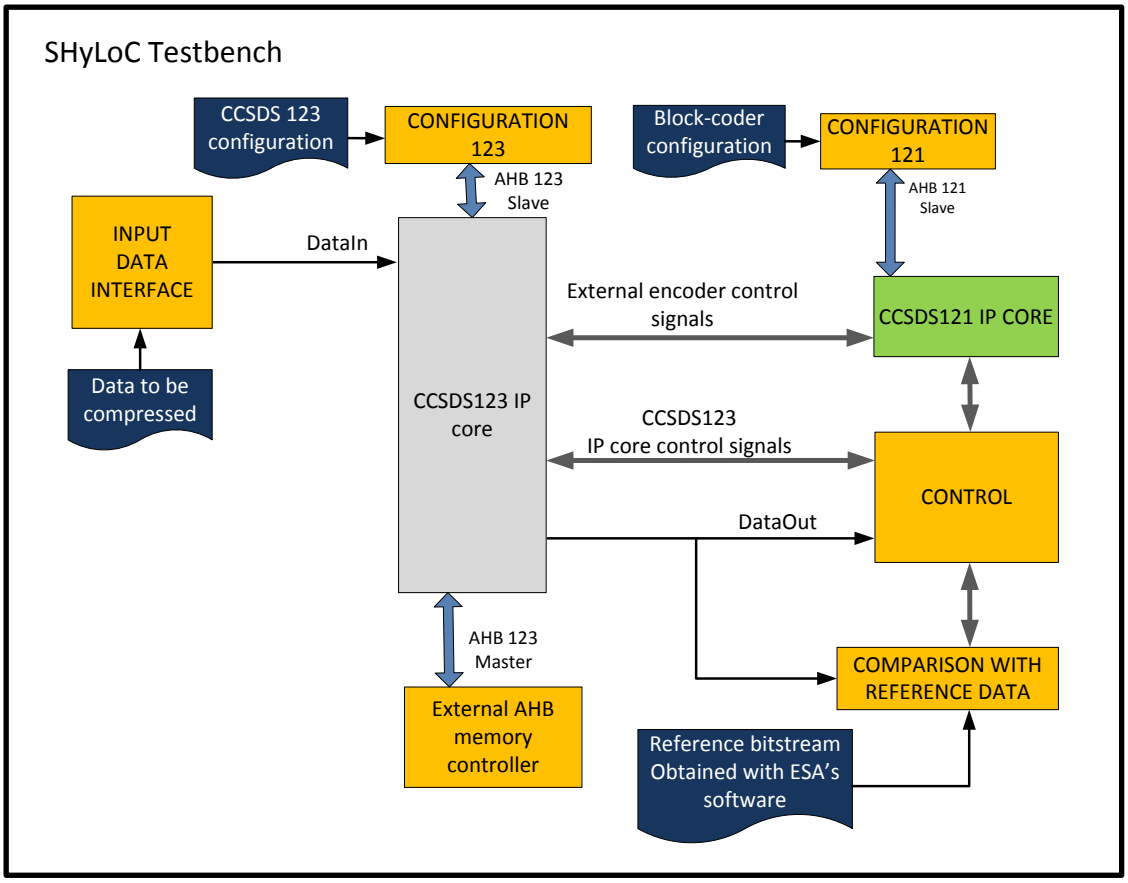

Figure 2-12: SHyLoC testbench and IP cores under verification

In particular, the verification dataset combines 35 real and synthetic hyperspectral images with different number of samples, dynamic range and sample order; 10 sets of compile-time configuration parameters and 10 sets of run-time configuration parameters. In total 121 simulations have been performed. All simulations have been run with QuestaSim and automated with scripts.

Additionally, 17 validation tests have been performed on a demonstrator that includes both the CCSDS-121 and CCSDS-123 IP cores. The maximum throughput achieved during the execution of the validation tests has been 1 Gbps. The architecture of the proposed demonstrator is shown in Figure 2-13. It consists of two parts:

- **Compression Core Board**: An FPGA based board with SpW interfaces, which hosts the developed cores and the adaptation logic required for the validation and demonstrations. The board is based on PLDA's XpressV6 has the following main features:
    - Xilinx Virtex-6 LX240T or LX550T FPGA (LX240T version will be purchased for the specific project)
    - Up to 2x4GB DDR2 SDRAM
    - Extension connector with up to 168 signals
    - PCIe core with multi-channel DMA host interface

- **EGSE**: An EGSE based on the iSAFT Simulator connected to the Compression Core board through a SpW link which validates the core's functionality, by stimulating the board and retrieving the results through RMAP transactions over a SpW link.

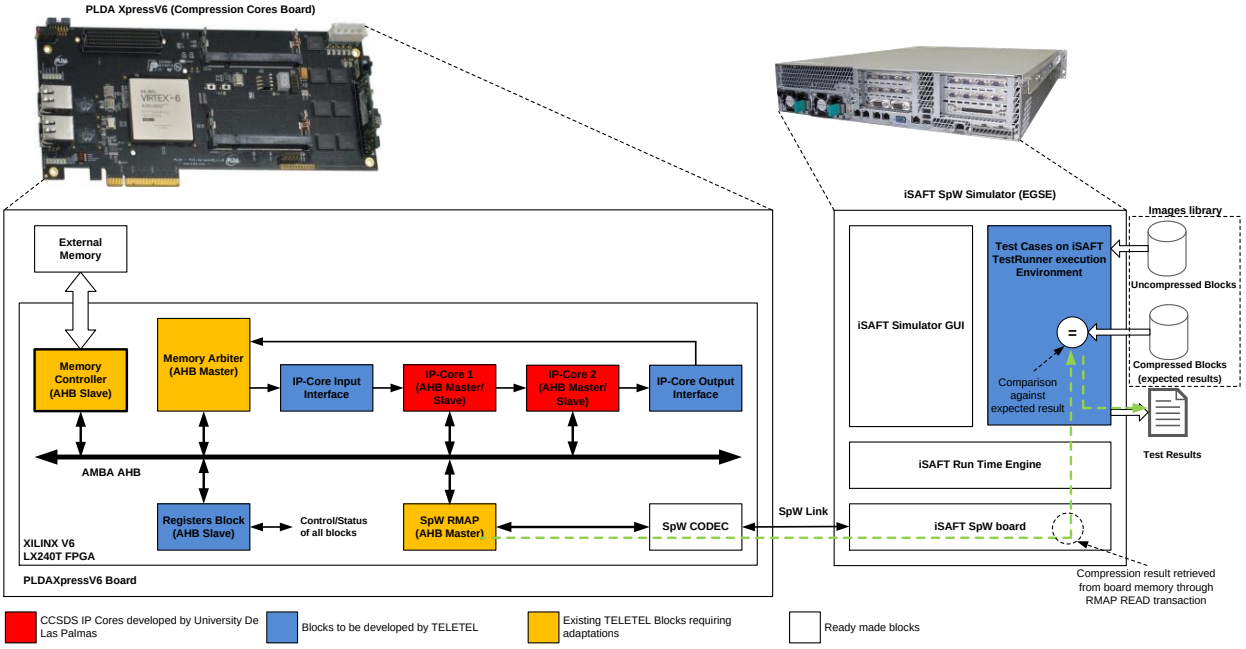

Figure 2-13: Demonstrator architecture.

2.7 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 and ASIC technologies:

- FPGA:
  - Xilinx Virtex 5 & 5QR
  - Microsemi ProASIC3E, ProASIC3L, RTAX2000, RTAX4000 and RTG4
- ASIC DARE 180 nm

In the following sections, we present a summary of the synthesis results obtained for the Xilinx Virtex5 FX130 FPGA, Microsemi RTG4 150 and DARE 180 nm. The complete technology mapping results can be consulted in the IP core's Datasheet [RD-10].

2.7.1 Synthesis results of the CCSDS-121 IP core

We perform the synthesis 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 16, and the size

of the output buffer is set to 32 bits. 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	3 (320)
LUTs	3510 (3%)
Max. Frequency (MHz)	118

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

Resources	Used
Carry Cells	792 (0.52%)
Sequential Cells	1347 (0.89%)
Block RAM (RAM64x18_RT)	11 (209)
DSP Blocks	4 (462)
LUTs	5419 (3.57%)
Maximum Frequency (MHz)	42.6

TABLE 2-3: CCSDS121- SYNTHESIS RESULTS ON DARE 180 NM (*)

Resource	Usage
Area (mm^2)	3.28
Gates (kilo)	63.02

(*) Memories smaller than 64 words are mapped to FFs/Combinatorial logic.

The results show feasibility of the implementation in both FPGA targets. The throughput reaches 118 MSamples per second for the Virtex5 FPGA. The core is able to accept one sample per clock cycle after it is configured.

2.7.2 Synthesis results of the CCSDS-123 IP core

We perform 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 perform 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 $Nx = 512$, $Ny = 1024$. The P value is set to 3 in all experiments.

Synthesis results on the Virtex5 FX130 and on the Microsemi RTG4 150 are shown in Table 2-1 and Table 2-2 respectively. The results for DARE 180 nm can be seen in Figure 2-14 and Figure 2-15.

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
I/O	840	317	317	317	317
BUFGs	32	1	2	2	1
Block RAMs	298	123	11	1	123
DSP48	320	8	8	6	9
Registers	81920	2753	3510	3101	2924
LUTs	81920	4564	5272	5264	5042
Maximum Frequency (Clk_AHB) (MHz)			150.4	150.4	
Maximum Frequency (Clk_S) (MHz)		154.3	130.3	113.4	152.5
Parameters	**Total Resources**	**HYPERSPECTRAL (AVIRIS)**			
		BIP	**BIP-MEM**	**BSQ**	**BIL**
I/O	840	325	325	325	325
BUFGs	32	2	2	2	2
Block RAMs	298	74	10	1	74
DSP48	320	10	15	9	12
Registers	81920	3658	4187	3899	3745
LUTs	81920	5750	6486	6574	6211
Maximum Frequency (Clk_AHB) (MHz)			188.8	154.4	
Maximum Frequency (Clk_S) (MHz)		130.9	128.8	132.4	130.9
Parameters	**Total Resources**	**ULTRASPECTRAL (AIRS)**			
		BIP	**BIP-MEM**	**BSQ**	**BIL**
I/O	840	323	323	323	323
BUFGs	32	1	2	2	1

Block RAMs	298	123	11	1	123
DSP48	320	8	8	6	9
Registers	81920	2753	3510	3101	2924
LUTs	81920	4564	5272	5264	5042
Maximum Frequency (Clk_AHB) (MHz)			186.8	132.2	
Maximum Frequency (Clk_S) (MHz)		131.6	129.7	109.6	131.6
Parameters	**Total Resources**	**RUNTIME CONFIG**			
		BIP	**BIP-MEM**	**BSQ**	**BIL**
I/O	840	325	325	325	325
BUFGs	32	2	2	2	2
Block RAMs	298	74	10	1	74
DSP48	320	10	15	9	12
Registers	81920	3658	4187	3899	3745
LUTs	81920	5750	6486	6574	6211
Maximum Frequency (Clk_AHB) (MHz)		313.6	182.7	136.1	313.6
Maximum Frequency (Clk_S) (MHz)		142.6	124.2	109.9	121.1

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

Parameters	**Total Resources**	**MULTISPECTRAL (LANDSAT)**			
		BIP	**BIP-MEM**	**BSQ**	**BIL**
IO Cells	720	190	212	212	190
Carry Cells	151824	1934	2289	2361	2046
Sequential Cells	151824	2384	3016	2788	2757
Block RAMs (1Kx18 + 64x18)	209	4 + 31	0 + 33	1 + 25	10 + 38
DSP Blocks	462	7	7	7	7
LUTs	151824	4934	6104	5918	5378
Maximum Frequency (Clk_AHB) (MHz)			102.8	91.8	
Maximum Frequency (Clk_S) (MHz)		80.7	82.4	72.6	79.8
Parameters	**Total Resources**	**HYPERSPECTRAL (AVIRIS)**			
		BIP	**BIP-MEM**	**BSQ**	**BIL**
IO Cells	720	198	220	236	198
Carry Cells	151824	2777	3157	2914	2797

Sequential Cells	151824	3052	3741	3177	3454
Block RAMs (1Kx18 + 64x18)	209 + 210	129 + 62	1 + 64	1 + 36	135 + 65
DSP Blocks	462	13	13	11	13
LUTs	151824	7263	7658	6987	7552
Maximum Frequency (Clk_AHB) (MHz)			108.0	85.4	
Maximum Frequency (Clk_S) (MHz)		62.3	74.0	68.9	62.4
Parameters	**Total Resources**	**ULTRASPECTRAL (AIRS)**			
		BIP	**BIP-MEM**	**BSQ**	**BIL**
IO Cells	720	196	218	234	196
Carry Cells	151824	2653	3034	2708	2682
Sequential Cells	151824	2869	3552	3047	3178
Block RAMs (1Kx18 + 64x18)	209 + 210	278 + 20	22 + 22	0 + 30	275 + 30
DSP Blocks	462	7	7	6	7
LUTs	151824	7442	7167	6864	7763
Maximum Frequency (Clk_AHB) (MHz)			99.4	85.6	
Maximum Frequency (Clk_S) (MHz)		57.2	76.1	70.2	59.0
Parameters	**Total Resources**	**RUNTIME CONFIG**			
		BIP	**BIP-MEM**	**BSQ**	**BIL**
IO Cells	720	252	272	288	252
Carry Cells	151824	2950	3421	3119	3005
Sequential Cells	151824	3652	4426	3825	3971
Block RAMs (1Kx18 + 64x18)	209 + 210	129 + 64	1 + 66	1 + 38	135 + 67
DSP Blocks	462	16	16	14	16
LUTs	151824	9391	9825	9251	9617
Maximum Frequency (Clk_AHB) (MHz)		108.8	103.6	78.3	109.6
Maximum Frequency (Clk_S) (MHz)		58.6	65.0	56.0	62.4

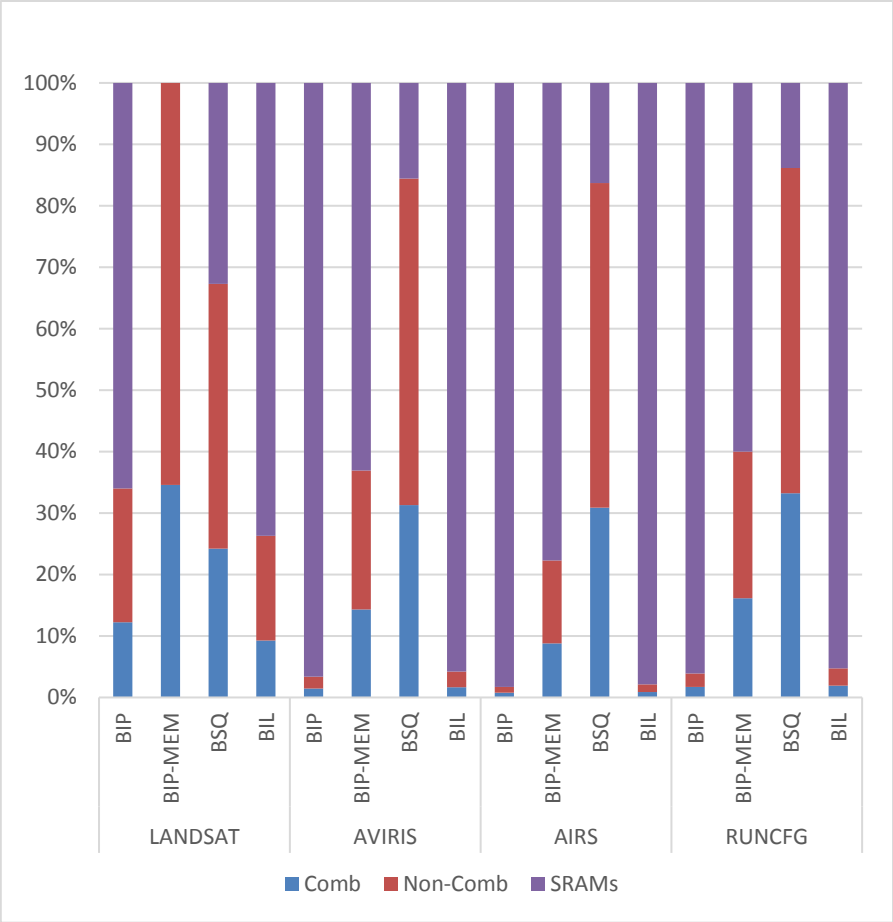

Figure 2-14: CCSDS-123 DARE 180nm mapping: area distribution (*)

(*) There is an overhead in memory usage do to port bit width alignment to match the memories aspects ratios from the DARE library that were available for this project.

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 (140 MSamples per second) is achieved for the BIP architecture, which is capable of compressing one sample every clock cycle. The BIP-mem architecture can also accept samples at a rate of one per clock cycle, however, a penalty in throughput is observed due to the use of an external memory. On the other hand, BIP-mem exhibits, as expected, much lower use of Block RAMs than BIP. The BSQ architecture has the lowest resource usage and also the lowest throughput, since some operations have been scheduled in 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.

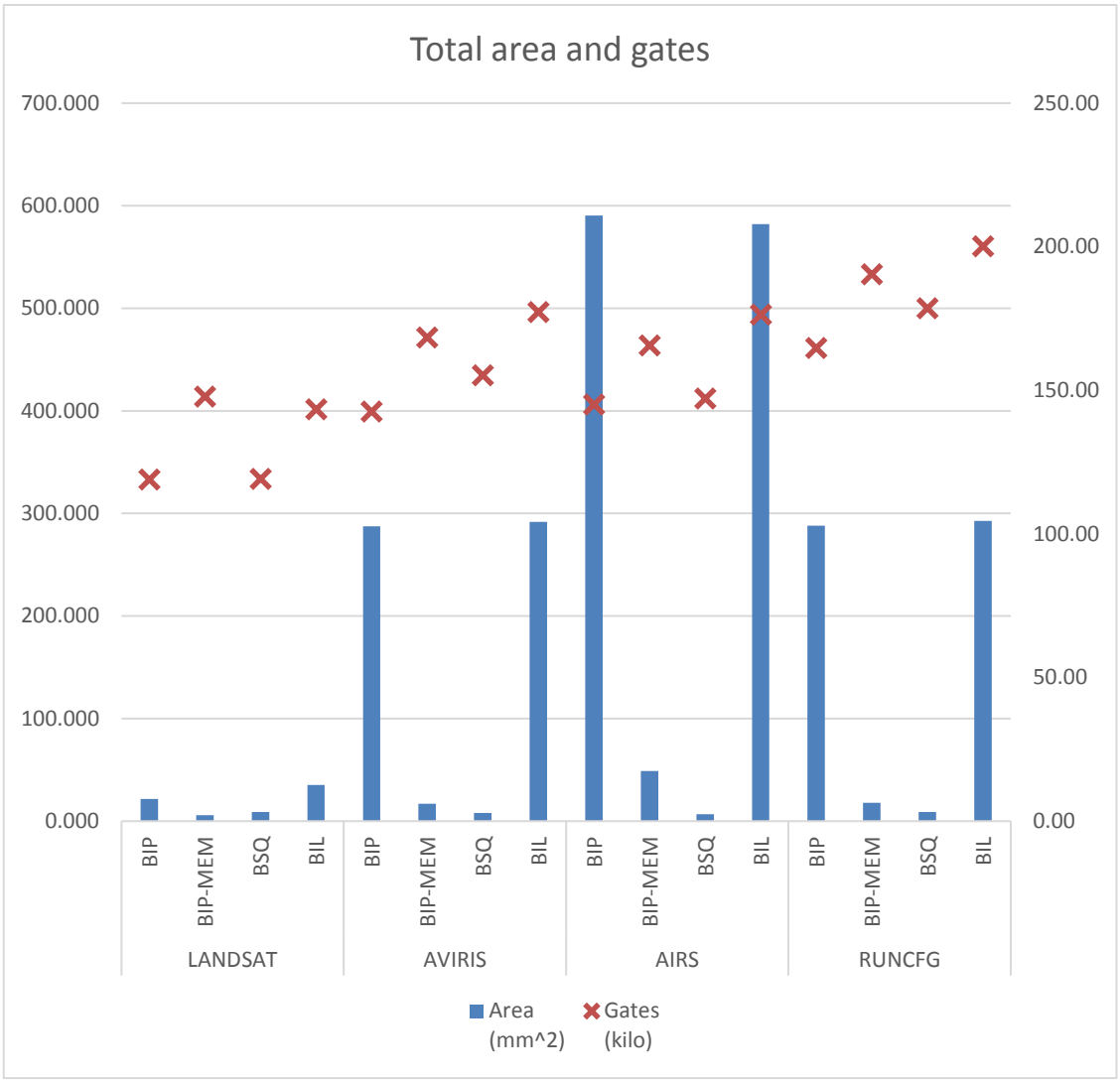

Figure 2-15: CCSDS-123 DARE 180nm total gates and area

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. Finally, the results point out that the versatility of the IP core might be further improved by adding an external memory interface in the BIL architecture, in order to reduce the usage of Block RAM and make the implementation feasible also for bigger images.

2.8 Conclusions

In this work, we have presented the SystemC modelling, hardware architecture design, VHDL description and implementation of two IP cores, which perform lossless compression as specified by the the CCSDS-121 and CCSDS-123 standards. 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, ProASIC3L, RTAX2000, RTAX4000 and RTG4 and to the DARE 180 nm standard cell library.

Synthesis results have been presented for the Virtex5 FX130, RTG4 150 FPGA and DARE 180 nm, 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. The developed demonstrator validates the design and shows a throughput of up to 1 Gbps.