Doctorado en Tecnologías de Telecomunicación e Ingeniería Computacional

A modular approach for the compression of payload information on-board satellites

Yubal Barrios Alfaro

Las Palmas de Gran Canaria, May 2022

Abstract

The integration on-board satellites of high-resolution sensors, such as spectroscopic or video sensors, is becoming common in the space industry. These sensors provide a large amount of information about the observed scene that is difficult to handle on-board, because satellites have limited computational and storage resources. For these reasons, on-board data compression is becoming mandatory for future space missions integrating this kind of high-resolution sensors. At the same time, compression algorithms must meet some requirements specifically imposed by the space environment, such as low complexity, low power consumption or robustness to errors due to radiation, among others. This makes challenging both to develop specific compression techniques for space missions and also to implement them in electronic devices that must be qualified to work in the outer space.

This Thesis proposes new modular hardware solutions for the compression of generic data and hyperspectral images on-board satellites. The goal is to provide several alternatives of prediction-based preprocessors and entropy coders, which are the two main stages of a compressor, that can be combined by the user. In this way, the optimal solution for the final application can be selected depending on certain requirements, such as target compression ratio, data quality after decompression, computational performance, hardware occupancy or robustness against radiation-induced failures. The implementation of the different prediction-based preprocessors and entropy coding alternatives is carried out on FPGAs, which are becoming popular in the space industry due to their high computational capabilities, low power consumption and the possibility of being reconfigured during the mission lifetime, either to add new functionality or to repair errors caused by radiation. Implementation results are provided for two recently available radiation tolerant FPGA technologies, the BRAVE and Kintex UltraScale families, manufactured by NanoXplore and Xilinx, respectively. Two different design methodologies have been combined, RTL (using VHDL) and HLS, choosing one or the other depending on certain constraints, such as computational performance requirements, logic resource utilization or design time restrictions, among others.

Concerning algorithms, this Thesis focuses on the compression algorithms proposed by CCSDS, an international organism in charge of publishing standards to be used in on-board processing systems for space missions. For generic data compression, the CCSDS 121.0-B-3 lossless compression standard is proposed, while the CCSDS 123.0-B-1 and CCSDS 123.0-B-2 standards are considered for the lossless and near-lossless compression,

respectively, of multi- and hyperspectral images. These compression standards have been studied in detail, in order to detect existing data dependencies between internal tasks and how to solve them. In this way, it is possible to parallelize certain computations of the algorithm, which are then executed concurrently thanks to the possibilities offered by FPGAs.

A fully compliant solution with the CCSDS 121.0-B-3 standard (i.e., unit-delay predictor plus block-adaptive entropy encoder), has been developed in VHDL and successfully mapped on the Xilinx Kintex UltraScale XCKU040 FPGA, obtaining a throughput of 176.3 and 163.7 MSamples/s when the input samples have a bit-depth of 8 and 16 bits, respectively. This performance enables real-time processing for spectroscopic sensors currently in orbit and for those planned for launch in the short to medium term. This computational performance is achieved with a low area footprint, using only 4.2% of the LUTs available on the device and with a minimum memory consumption (around 0.8%).

Regarding the CCSDS 123.0-B-1 lossless standard, it has also been completely developed in VHDL and is comprised by both the 3D predictor defined in the standard and the sample-adaptive encoder. Different architectures have been developed for the prediction stage, taking into account data dependencies in the processing of each possible order of the input image, with the aim of maximising the throughput without significantly affecting logic resource utilization. When the input order is Band-Interleaved by Pixel (BIP), the only one that allows processing of one sample per clock cycle, a maximum throughput of 151.6 MSamples/s is obtained for the XCKU040 FPGA, when compressing scenes acquired by AVIRIS, which have a precision of 16 bits per pixel. Logic resource consumption is considerably low (around 3% of the total LUTs available on the device), while the memory consumption depends directly on the image size (e.g., 12 % of BRAMs of the XCKU040 FPGA are used for AVIRIS scenes).

The CCSDS 123.0-B-2 near-lossless algorithm has been fully developed in HLS to quickly perform a design space exploration of the different predictor architectures, while at the same time the existing restrictions in terms of available design time are overcome. A design space exploration has been carried out to know which parts of the algorithm are the most critical in terms of achieved compression ratio and hardware occupancy. The impact of the different configuration parameters defined by the standard on the performance of the hardware implementation has also been studied. This solution consumes the 7.2% of LUTs and 14.2% of BRAMs available on the Kintex UltraScale XCKU040. A tailored version of this approach was made for the CHIME instrument, composed of just the

CCSDS 123.0-B-2 predictor features that allow the highest possible compression ratio, and the block-adaptive encoder. This approach, although it is not capable of providing real-time compression, has an acceptable hardware footprint (the 6.0% of LUTs and 28.5% of BRAMs available on the Xilinx XCKU040 FPGA). As it can be seen from the results obtained, the proposed implementations are feasible to be successfully implemented on hardware available on-board satellites.

Finally, as part of this Thesis, the application of the CCSDS 123.0-B-2 algorithm for monochromatic and RGB video compression has been evaluated, demonstrating the versatility of the proposed modular compression solution, capable of compressing both three-dimensional images and video sequences by using a single processing core. This solution provides acceptable compression ratios for a real Remote Sensing scenario, obtaining a video quality after decompression high enough to not perceive significant losses from a visual point of view.

In conclusion, this work presents efficient compression solutions capable of handling different types of data, including one-dimensional data, hyperspectral imagery and video sequences, which can be embarked on-board satellites in future space missions. In this way, flexible and high-performance compression approaches are provided, all with a low use of computational and memory resources.

Resumen

La integración a bordo de satélites de sensores de alta resolución, como pueden ser sensores hiperspectrales o de vídeo, es cada vez más común por parte de la industria espacial. Estos sensores proporcionan una gran cantidad de información acerca de la escena observada que es difícil de manejar a bordo, debido a que los satélites cuentan con recursos tanto computacionales como de almacenamiento limitados. Por estas razones, la compresión a bordo de satélites se está convirtiendo en una necesidad para futuras misiones espaciales que integren sensores de alta resolución. Al mismo tiempo, los algoritmos de compresión deben cumplir una serie de requisitos impuestos específicamente por el entorno espacial, como puede ser una baja complejidad, un reducido consumo de potencia o robustez frente a errores debido a la radiación, entre otros. Todo esto hace que suponga un reto tanto el desarrollo de técnicas de compresión específicas para misiones espaciales como su implementación en dispositivos electrónicos que deben estar calificados para trabajar en el espacio.

En esta Tesis se proponen nuevas soluciones hardware modulares para la compresión de datos genéricos e imágenes hiperespectrales a bordo de satélites. La idea fundamental radica en proporcionar varias alternativas de preprocesadores basados en predicción y de codificadores entrópicos, que son las dos etapas básicas de que consta un compresor, pudiendo ser combinadas por el usuario. De esta forma, se podrá seleccionar la opción óptima para la aplicación final dependiendo de determinados requisitos, como puede ser compresión objetivo, calidad de los datos después de la decompresión, prestaciones temporales, ocupación en términos de área, o robustez frente a fallos provocados por la radiación. Concretamente, dichas implementaciones se llevan a cabo en FPGAs, dispositivos electrónicos que están adquiriendo cada vez más interés por parte de la industria espacial debido a sus altas capacidades computacionales, reducido consumo de potencia y posibilidad de ser reconfiguradas durante el ciclo de vida de la misión, bien para añadir nueva funcionalidad o para reparar errores producidos por la radiación. En este sentido, cabe destacar que se proporcionan resultados para dos tecnologías FPGA tolerantes a la radiación recientemente disponibles en el mercado, como son las familias BRAVE y Kintex UltraScale, proporcionadas por los fabricantes NanoXplore y Xilinx, respectivamente. Dos diferentes metodologías de diseño han sido combinadas, RTL (utilizando VHDL) y HLS, eligiendo una u otra en función de ciertas restricciones, como pueden ser requisitos de rendimiento computacional, de utilización de recursos lógicos o limitaciones en el tiempo de diseño, entre otras.

En lo que a algoritmos se refiere, se presta especial atención a los estándares de compresión propuestos por el CCSDS, organismo encargado de publicar normas para ser empleadas en los sistemas de procesamiento a bordo de misiones espaciales. Para la compresión de datos genéricos, se propone el empleo del estándar de compresión sin pérdidas CCSDS 121.0-B-3, mientras que los estándares CCSDS 123.0-B-1 y 123.0-B-2 se consideran para la compresión de imágenes multi- e hiperspectrales sin o con pérdidas, respectivamente. Ambos estándares de compresión han sido profundamente estudiados, para poder detectar las dependencias de datos existentes entre tareas internas y cómo solucionarlas. De este modo, se consigue paralelizar ciertos cómputos del algoritmo, que son ejecutados concurrentemente gracias a las posibilidades ofrecidas por las FPGAs.

Se ha desarrollado en VHDL una solución conforme al estándar CCSDS 121.0-B-3 (*unit-delay predictor* más *block-adaptive encoder*), mapeado satisfactoriamente en la FPGA Kintex UltraScale XCKU040 de Xilinx, obteniendo un *throughput* de 176.3 y 163.7 MSamples/s cuando las muestras de entrada tienen una precisión de 8 y 16 bits, respectivamente. Estas prestaciones temporales permiten un procesamiento en tiempo real para los sensores hiperespectrales que se encuentran actualmente en órbita y para los que está previsto lanzar a corto o medio plazo. Este rendimiento computacional está unido a una baja ocupación de área, usando únicamente un 4.2% de las LUTs disponibles en el dispositivo y con un consumo de memoria prácticamente nulo (alrededor del 0.8%).

Respecto al estándar CCSDS 123.0-B-1, también ha sido completamente desarrollado en VHDL y está compuesto tanto por el predictor 3D definido en el estándar, como por el *sample-adaptive encoder*. Diferentes arquitecturas han sido desarrolladas para la etapa de predicción, teniendo en cuenta las dependencias de datos existentes en el procesamiento de cada orden posible de la imagen de entrada, con el objetivo de maximizar el *throughput* sin que la ocupación de recursos lógicos se vea significativamente afectada. Cuando el orden de entrada es *Band-Interleaved by Pixel* (BIP), el único que permite un procesamiento de una muestra por ciculo de reloj, se obtiene un *throughput* máximo de 151.6 MSamples/s para la FPGA XCKU040, cuando se comprimen escenas adquiridas por AVIRIS, que tienen una precisión de 16 bits por píxel. La utilización de recursos lógicos es considerablemente baja (alrededor del 3% del total de LUTs disponible en el dispositivo), mientras que el consumo de memoria depende directamente del tamaño de la imagen (por ejemplo, se utiliza el 12% de BRAMs de la FPGA XCKU040 para escenas adquiridas por AVIRIS).

El algoritmo descrito en el estándar CCSDS 123.0-B-2 se ha desarrollado completamente en HLS para realizar una exploración rápida del espacio de diseño y estudiar las diferentes

arquitecturas posibles del predictor, mientras que al mismo tiempo se evitan las restricciones existentes en cuanto a tiempo de diseño disponible. Se ha realizado una exploración del espacio de diseño para identificar qué partes del algoritmo son más críticas en términos de ratio de compresión y utilización del hardware. También se ha estudiado el impacto de los diferentes parámetros de configuración definidos en el estándar en el rendimiento de la implementación hardware. Esta solución consume el 7% de LUTs y el 14.2% de BRAM disponibles en la Kintex UltraScale XCKU040. Se ha realizado una versión a medida de esta solución para el instrumento del programa espacial CHIME, compuesta por la funcionalidad del predictor propuesto en el estándar CCSDS 123.0-B-2 que permitía alcanzar un mayor ratio de compresión, y el *block-adaptive encoder*. Esta solución, aunque no es capaz de proporcionar compresión en tiempo real, tiene una ocupación de hardware comedida (el 6% de LUTs y el 28.5% de BRAMs disponibles en la FPGA XCKU040 de Xilinx). Como se puede comprobar a partir de los resultados obtenidos, las implementaciones propuestas son viables para ser implementadas satisfactoriamente en hardware disponible a bordo de satélites.

Adicionalmente, se ha evaluado el uso del algoritmo CCSDS 123.0-B-2 para la compresión de vídeo monocromático y RGB, obteniendo de esta forma una solución de compresión versátil, capaz de comprimir tanto imágenes tridimensionales como secuencias de vídeo obteniendo un único núcleo de procesamiento. Esta solución proporciona ratios de compresión aceptables para un escenario real de teledetección, obteniendo una calidad de vídeo después de la decompresión lo suficientemente alta para no percibir pérdidas significativas desde un punto de vista visual.

En definitiva, este trabajo proporciona soluciones de compresión eficientes capaces de manejar diferentes tipos de datos, incluyendo datos unidimensionales, imágenes hiperespectrales y vídeo, que puedan ser embarcadas a bordo de satélites en futuras misiones espaciales. De esta forma, se dota a la misión espacial de técnicas de compresión flexibles y de alto rendimiento, y todo ello con un bajo uso de recursos computacionales y de memoria.

Acknowledgements

The development of this Thesis has not been an easy journey and it would have been practically impossible to finish it without the support of countless people who have been part of my professional and personal life during the last four years. First of all, I would like to express my gratitude to my supervisors, Prof. Roberto Sarmiento and Dr. Lucana Santos, who have supported and guided me during this research, always providing constructive comments from their experience in the field. During these four years, I have been part of the Institute for Applied Microelectronics (IUMA), research centre associated to the University of Las Palmas de Gran Canaria (ULPGC). As part of the IUMA team, I have involved in different European and ESA-funded projects, which have contributed to support with funding my PhD, whose research topics are clearly connected to the work done in those projects.

I also thank the European Space Agency for offering me the opportunity to make an international research stay in their facilities at the European Space Research and Technology Centre (ESTEC) located in Noordwijk, the Netherlands. I would like to thank all the TEC-EDM section for their hospitality during those three months and for sharing with me a valuable knowledge about the status of the European space industry in the short- and near-future. I am also grateful to people from Thales Alenia Space in Spain and in France, whose I have collaborated in different space programs during the past four years.

I cannot forget to thank my colleagues of the DSI Lab for easing the coexistence in the professional environment and also for the enriching discussions, the innumerable anecdotes out of work and of course for their patience in some stressful moments. If I had the possibility to come back to the beginning of my PhD to choose my teammates, needless to say that I would choose them.

Finally, but not less important, I am infinitely grateful to the most important people of my life, whose are of course my parents, for their generous support and the way they educate me, which has made me the person that I am. Thanks also to Laura for being my life partner and for always believing in me.

Contents

List of Figures

List of Tables

Abbreviations

AAT Arbitrary Affine Transform

ADC Analog-to-Digital Converter

AHB Advanced High-performance Bus

AIRS Atmospheric Infrared Sounder

AMBA Advanced Microcontroller Bus Architecture

ASI Italian Space Agency

ASIC Application Specific Integrated Circuit

AVC Advanced Video Coding

AVIRIS Airbone Visible/Infrared Imaging Spectrometer

AXI Advanced eXtensible Interface

BIL Band Interleaved by Line

BIP Band Interleaved by Pixel

bpp bits per pixel

BRAM Block RAM

BRAVE Big Re-programmable Array for Versatile Environment

BSQ Band Sequential

CABAC Context-Adaptive Binary Arithmetic Coding

CALIC Context-based, Adaptive, Lossless Image Codec

CAVLC Context-Adaptive Variable Length

CCSDS Consultative Committee for Space Data Systems

CHIME Copernicus Hyperspectral Imaging Mission for Environment

CNES National Centre for Space Studies

COTS Commercial off-the-Shell

CPU Central Processing Unit

CR Compression Ratio

CTU Coding Tree Unit

CU Coding Unit

DCT Discrete Cosine Transform

DFF D-type Flip Flop

DMR Double Modular Redundancy

DPCM Differential Pulse Code Modulation

DSP Digital Signal Processing

DST Discrete Sine Transform

DUT Design Under Test

DWT Discrete Wavelet Transform

EDA Electronic Design Automation

EDAC Error Detection And Correction

EnMAP Environmental Mapping and Analysis Program

EO Earth Observation

EOP End of Processing

ESA European Space Agency

ESTEC European Research and Technology Centre

EZW Embedded Zero-tree Wavelet

FAPEC Fully Adaptive Prediction Error Coder

FF Flip-Flop

FIFO First-In First-Out

FL Fast Lossless

FLEX Fast Lossless EXtended

FPGA Field Programmable Gate Array

FPS Frames Per Second

FS Fundamental Sequence

FSM Finite-State Machine

GFLOPS Giga-Floating-point Operations per Second

GPO2 Golomb-power-of-2

GPU Graphics Processing Unit

HDL Hardware Description Language

HEVC High Efficiency Video Coding

HISUI Hyperspectral Imager Suite

HPSC High-Performance Spaceflight Computing

HSI HyperSpectral Imaging

HyspIRI Hyperspectral Infrared Imager

IAA Instituto de Astrofísica de Andalucía

IAC Instituto de Astrofísica de Canarias

I/O Input and Output

IP Intellectual Property

ISA Instruction Set Architecture

ISS International Space Statione

IUMA Institute for Applied Microelectronics

IWT Integer Wavelet Transform

KLT Karhunen-Loeve Transform

LCE Lossy Compression for Exomars

LCPLC Low-Complexity Predictive Lossy Compression

LCTF Liquid Crystal Tunable Filter

LEO Low-Earth Orbit

LMS Least Mean Square

LSB Less Significant Bit

LUT Look-Up Table

LZW Lempel-Ziv-Welch

MAC Multiply and Accumulate

MB Macroblock

MBU Multiple Bit Upset

MMU Mass Memory Unit

MPSoC Multi-Processor System-on-Chip

MODIS Moderate Resolution Imaging Spectroradiometer

MSB Most Significant Bit

MSE Mean Squared Error

MSI Multispectral Instrument

MSS Multispectral Scanner System

NASA National Aeronautics and Space Administration

NGMP Next Generation Multipurpose Processor

NRE Non-Recurring Engineering

PCA Principal Component Analysis

PFCU Processing, Formatting and Control Unit

PLL Phase-Locked Loop

POT Pairwise Orthogonal Transform

PRDC Payload Rice Data Compressor

PSNR Peak-Signal to Noise Ratio

RAM Random Access Memory

RHBD Radiation-Hardened by Design

ROI Region of Interest

SAO Sample Adaptive Offset

SEE Single Event Effect

SEFI Single Event Functional Interrupt

SEU Single Event Upset

SHALOM Spaceborne Hyperspectral Applicative Land and Ocean Mission

SHyLoC Hyperspectral Lossless Compressor for space applications

SNR Signal to Noise Ratio

SoC System-on-Chip

SPARC Scalable Processor ARChitecture

SRAM Static RAM

TID Total Ionizing Dose

TMR Triple Modular Redundancy

TTM Time-To-Market

UAB Universitat Autònoma de Barcelona

ULPGC University of Las Palmas de Gran Canaria

VCL Video Coding Layer

VHDL VHSIC Hardware Description Language

WICOM Wavelet Image Compression Module

WPP Wavefront Parallel Processing

Chapter 1

Introduction

This Chapter presents the motivation and objectives of the Thesis, which is focused on providing efficient hardware implementations for on-board data compression. The main goal of the work is to develop different modular solutions to compress generic data, hyperspectral images and video sequences on-board satellites using configurable hardware devices and keeping in mind the main constraints existing on space missions (computational resources, storage capacity and power consumption, among others). This Chapter also provides some hints about the constraints to deal with when designing hardware for space. This information is required to understand the problems involved and the solutions provided.

1.1 Motivation

Remote Sensing applications have become very popular for the space industry during the last decades, since high-resolution sensors allow to obtain useful information for an important number of applications (e.g. environmental studies, climate, surveillance, characterization, monitoring, detection, etc.) [1–7]. These kind of sensors, commonly used in Earth Observation (EO) missions, are also gaining interest for space exploration, and are currently considered by space agencies to study the Moon or the Mars surface [8, 9].

Among all sensors, those with the capability to obtain rich spectral information have been used since the beginning because of their strengths for characterization and monitorization applications. A proof of this is the number of satellites with multi- or hyperspectral sensors on-board. The history starts with Landsat-1, launched by NASA in 1972, which can be considered the first modern satellite for EO [10]. Landsat-1 integrated a Multispectral Scanner System (MSS) that acquires four bands with wavelengths from 0.5 to 1.1 μm with a spatial resolution of 80 m. The Landsat program has launched a total of nine satellites and still continues. Landsat-9 has been recently launched on September 2021 and it will join Landsat-8 in orbit. Other important missions by NASA are the Earth-Observing One (EO-1) satellite, which embarked the Hyperion Imaging Spectrometer (decommissioned on March 2017) [11]; and the Hyperspectral Infrared Imager (HyspIRI) mission, aims to provide critical information on natural disasters and vegetation health [12]. HyspIRI was launched in 2020 and includes an imaging spectrometer measuring from the visible to short-wave infrared (i.e. from 380 nm to 2.5 μm) in 10 nm contiguous bands, with a spatial resolution of 60 m.

In Europe, the Copernicus program has the main goal of providing real-time dynamic monitoring of the environment. The space-based observation part of the program is responsibility of the European Space Agency (ESA), which has launched a family of six satellites, known as Sentinel [13]. Sentinel-2 integrates a Multispectral Instrument (MSI) that acquires 13 spectral channels: the first four bands are the three RGB bands and a Near Infrared (NIR) spectral channel, with a spatial resolution of 10 m; six bands, four of them in the Visible and Near Infrared (VNIR) and 2 Short Wavelengths Infrared (SWIR) bands (1.61 μm and 2.19 μm), with a spatial resolution of 20 m; and the last three bands for atmospheric correction (443 nm, 945 nm and 1374 nm), with a 60 m spatial resolution.

The Copernicus Hyperspectral Imaging Mission for Environment (CHIME) is introduced by ESA in the future Copernicus 2.0 program to provide routine hyperspectral observations

for the monitorization of natural resources, including applications such as sustainable agricultural and biodiversity management, soil properties characterization, sustainable mining practices and environment preservation [14]. CHIME shall provide continuous spectral data in VNIR and SWIR spectral domains, covering approximately 250 spectral channels between 400 nm and 2.5 μm. The sensor will be used to acquire hyperspectral images with a bit-depth of 16 bits per sample, what leads to input data rates up to 2 Gbps. This Thesis will provide an ad-hoc solution for compressing information acquired by the CHIME instrument (see Chapter 5).

European space agencies have also launched different EO satellites as part of their national programs. The National Centre for Space Studies (CNES) has launched seven satellites, known as the SPOT family. SPOT-1, launched in 1986, is considered the world's first Remote Sensing satellite. A new French mission, denoted as HypXIM [15], is expected to be launched in the time-frame 2022-2024, collecting a total of 210 spectral bands in the range 400-2500 nm (VNIR and SWIR regions), with a spectral and spatial resolution of 10 nm and 8 m, respectively. PRISMA is a mission fully funded by the Italian Space Agency (ASI) and launched in 2019 [16]. The Spaceborne Hyperspectral Applicative Land and Ocean Mission (SHALOM) is a joint mission by the Israeli and the Italian space agencies to develop a hyperspectral satellite that collects 275 bands with a spectral resolution of 10 nm and a spatial resolution of 10 m [17]. The Environmental Mapping and Analysis Program (EnMAP) is a German hyperspectral satellite mission for Earth monitoring and characterization, embarking a sensor that collects continuous spectral bands of 6-14 μm width in the range 420-2450 nm, with a spatial resolution of 30 m [18].

Finally other Agencies (as Japanese and Chinese) has launched a number of similar programs to the aforementioned ones [19–21]. Table 1.1 summarises the main features of the most relevant current and future space missions that embark hyperspectral sensors, including the number of bands collected and both the spatial and the spectral resolution.

Table 1.1 shows that high-resolution sensors are conceived for acquiring as many spectral bands as possible with a high spatial and spectral resolution. This extra information in the spectral domain is intended to obtain a high grade of information about the objective under analysis. A first solution could be to acquire the information on-board and send it to ground in a raw format to be then processed. However, the satellite downlink bandwidth with ground stations usually is limited, as is the memory storage capacity on-board. In this way, data compression emerges as the solution to dodge these constraints, reducing the data volume prior to send it to ground.

Table 1.1: Main features of current and future spacecraft with on-board hyperspectral sensors

Sensor	Number of bands	Spectral Range(nm)	Spectral Resolution(nm)	Spatial Resolution(m)	Launch
HyspIRI (USA)	210	380–2500	10	60	2020
CHIME (Europe)	220	400-2500	10	20-30	2027-2029
PRISMA (Italy)	240	400-2500	10-12	30	2019
SHALOM (Italy/Israel)	275	400-2500	10	10	2022
HypXIM (France)	210	400-2500	10	8	2022-2024
EnMAP (Germany)	$\tilde{2}00$	420-2450	6-14	30	2020
HISUI (Japan)	185	400-2500	10-12.5	30	2019
Gaofen-5 (China)	330	400-2500	5-10	30	2018
Zhuhai-1 (China)	166	450-2500	10-20	30	2019

On the other hand, the demand of video sensors embarked on satellites is also increasing in the last years since they provide extra information in the temporal domain. This interest will grow exponentially in the near-future, mainly for applications such as monitorization, including target detection and tracking, and exploration missions. Nowadays, some commercial satellites provide video products for potential applications on ground. For instance, SkySat is a constellation of 21 high-resolution imaging satellites (SkySat-1 was launched in 2013) at different orbits and altitudes. They are able to collect video using a panchromatic camera, which has a spatial resolution of approximately 1x2.5 km^2 and with a temporal duration between 30 and 120 seconds at 30 FPS [22]. The Jilin-1 Chinese satellite constellation also provides high-definition video imaging, optical and hyperspectral imagery [23]. High-definition RGB video is provided in real-time at 10 FPS, with a duration up to 120 seconds and a spatial resolution around 1 m, covering a scene of 19 km [24]. Academic institutions are also working in CubeSat missions comprised by Commercial Off-The-Shelf (COTS) components and including image and video sensors for EO. As an example, the High-Resolution Image and Video CubeSat (HiREV), currently under development by the Korea Aerospace Research Institute, will integrate a sensor for high-resolution (5 m per pixel) color image and video [25]. Moreover, multi- and hyperspectral video is expected on space missions in a short future, mainly due to the availability of the snapshot cameras.

In conclusion, it is expected that the new-generation imaging sensors boost in the next years both the spatial and the spectral resolution, even collecting temporal information for video applications. Nowadays, there is a huge amount of data being acquired on satellites and this amount will be much higher in the short future. As mentioned before, the limited downlink bandwidth in comparison with the size of the acquired data will constitute a bottleneck for missions that integrate this kind of sensors, preventing the direct

transmission of raw data to ground [26]. Therefore, performing on-board compression is becoming mandatory on space missions for Remote Sensing applications.

Data compression is still considered a challenge by the space industry. Very efficient and low-complexity algorithms and processing platforms will be required on-board to compress that huge amount of information in real-time. This has been a hot topic for the scientific community during the last decades, mainly because the computational restrictions present on-board satellites force the compression algorithms to meet certain design criteria. At the same time, it is very important to preserve the quality of the reconstructed data after decompression because it will be impossible to capture the same data twice.

This is the reason why the Consultative Committee for Space Data Systems (CCSDS), an international organization comprised by the main space agencies in the world to define a common procedure for developing space data and information systems, has published different compression standards. The target of these algorithms has different data nature (1D, 2D and 3D data) and compression techniques (i.e., lossless or lossy), but always meet the condition of a reduced algorithmic complexity. These standards pretend to establish a common framework for the development of on-board compression solutions and to have an universal solution for the decompression of the information on ground. Regarding one-dimensional data, image and hyperspectral image compression, the CCSDS has published the following Recommended Standards: a) CCSDS 121.0-B-3 (Lossless Data Compression) [27], b) CCSDS 122.0-B-2 (Image Data Compression) [28], c) CCSDS 122.1-B-1 (Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression) [29], d) CCSDS 123.0-B-1 (Lossless Multispectral and Hyperspectral Image Compression) [30] and e) CCSDS 123.0-B-2 (Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression) [31].

Although the algorithms proposed in these standards have been selected in order to be feasibly implemented on the hardware available on-board satellites, the development of these solutions is not trivial, and it is still challenging for both academic institutions and space companies. The most common solution nowadays is to include in the system an IP core (soft or hard, depending on the availability and mission requirements) that implements the specific compression technique needed on-board. If there is not an IP available, the solution is to perform the compression as a software application running on a general-purpose microprocessor, though in this case the performance goals are very hard to met.

The main goal of this Thesis is to provide efficient compression solutions to be implemented on space missions. The proposed solutions shall be able to manage data from different nature, including one-dimensional data, hyperspectral images and video (panchromatic and RGB). The proposed approaches have to achieve not only an acceptable compression ratio without degrading the quality of the compressed information, but they also have to take into account additional constraints that appear on-board satellites, such as throughput to meet real-time requirements; low area utilization, to fit well on the available hardware resources embarked on space missions; and low power consumption. The designed solutions will be developed following a modular approach, based on the fact that many of these standards have computational similarities. In particular, this Thesis will study the CCSDS 121.0-B-3, the CCSDS 123.0-B-1 and the CCSDS 123.0-B-2 standard, from which computational modules will be created that can be merged together to solve the problem of one-dimensional data compression, multispectral and hyperspectral image compression and video compression on-board satellites.

1.2 Application environment constraints

When dealing with on-board applications, there is a number of constraints to take into account. The spacecraft is considered as a platform and a payload. For EO satellites, the payload usually includes a hyperspectral sensor and/or a panchromatic/RGB camera, together with the electronic circuits needed to acquire, process and store the information. The platform consists in the satellite subsystems that supports the payload [32]. These subsystems (such as electrical power distribution; telemetry, tracking and command; thermal control; propulsion; etc.) are not covered in this Thesis. This Thesis will cover part of the on-board data handling subsystem and, particularly, will be part of the Processing, Formatting and Control Unit (PFCU) of the satellite, dealing with the compression of the information generated by the different sensors that can be embarked on it.

Usually on-board, depending of the mission, there are three different kind of sensors: Panchromatic, VNIR and SWIR. Apart from the optics and focal plane arrays for each sensor, the payload includes a specific processing element for each sensor, which provides the information in a specific formatted stream. This formatted data are stored in a Mass Memory Unit (MMU) or are directly processed on the fly. All the solutions provided in this Thesis are able to process the information on the fly, and make it ready either to be

sent to ground through the satellite downlink or to be stored in the MMU. Ancillary data, telemetry information or other type of data could be also necessary to send to ground. For the sake of saving downlink bandwidth, sometimes it is also necessary to compress this kind of one-dimensional information [33, 34].

Electronic circuits used in space applications are typically several generations behind of those used on ground, at least in terms of complexity and performance. However, this has changed in the last years with the availability of very high performance technologies and solutions, such as Field-Programmable Gate Arrays (FPGAs) or parallel processors. This fact is transferring to the design for space applications the problems intrinsically linked to the highly complex solutions of embedded systems with the aggravation of the problems related to radiation and thus the needed robustness. This section deals with all the related constraints that are important to consider in this Thesis. These topics may be technological, methodological or may be related to the applications themselves.

1.2.1 Hardware on-board satellites

The traditional way to introduce high performance circuits on a satellite have been Application-Specific Integrated Circuits (ASICs). An ASIC is a very efficient way to perform different mission tasks related to the payload. This is because ASIC solutions offer the best balance between performance and power consumption, since they are fully optimized for a specific application and the technology used for their manufacturing is thought for space applications (i.e., radiation tolerant). However, space is a very low volume application and hence it is difficult to reach the break-even point because of the high non-recurring engineering costs and manufacturing costs. The manufacturing time is also high and very sensitive to the possibility of requiring a re-spin. These limitations still exist even if a System-on-Chip (SoC) methodology (reusing already existing cores or IPs) is followed. With the possibility of using high performance on-board processors (e.g., LEON3 or LEON4), the use of ASICs in space applications have decreased.

In the other hand, FPGAs are increasing their presence in the last years as part of space payload processing circuits, because of their flexibility, high computational performance and low power consumption. A main advantage of FPGAs is also their reduced cost compared to ASICs [35], mainly recurring engineering costs and, to a lesser extent, non-recurring engineering costs. The inherent flexibility of the RAM-based FPGAs allows to change all or some parts of the functionality dynamically to adapt it to new requirements that can

appear during the mission lifetime or even if corruption takes place because of radiation effects. Besides, FPGAs encourage task parallelism, executing simultaneously different operations if no data dependencies are presented, providing an advantage compared to the sequential behaviour of embedded microprocessors.

Initially, the FPGAs used in the space sector were the ones fabricated with technologies inherently robust to radiation effects, mainly antifuse but also based on flash memories. Companies like Microsemi use this kind of programmable technology, which has a short development time to space applications, hence they have been widely used during the last decades. Lately, FPGAs based on SRAM technology are being predominant (the same than in ground applications) because of their flexibility and early adoption of more advanced manufacturing processes. In this way, Xilinx offers specific Radiation-Hardened by Design (RHBD) devices to the space industry, which are manufactured according to the special conditions that are present on a critical environment like the outer space (e.g., robustness against radiation or a wide range of temperature). These FPGAs are currently dominating the space market, and they have been integrated in the last years in different space payloads. Recently, NanoXplore, an European Company, has been designing and manufacturing a new family of re-programmable devices also based on SRAM technology, known as BRAVE. BRAVE FPGAs, which are supported by ESA and other European space agencies, offer a family of radiation-hardened reprogrammable devices integrally developed in Europe [36], avoiding dependencies with non-European technologies that have dominated the space market during the last decades. Figure 1.1 shows the first device of the family on the market, known as NG-MEDIUM.

Different hardening techniques are combined in space-graded FPGAs, in order to preserve a proper behaviour under radiation effects. Some of these approaches are Error Detection And Correction (EDAC) for dedicated memory blocks, Triple Modular Redundancy (TMR)

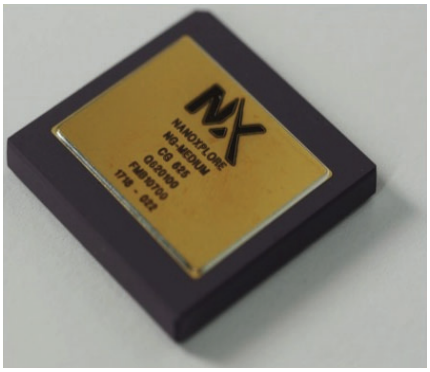


Figure 1.1: NanoXplore NG-MEDIUM FPGA

flip-flops, Double Modular Redundancy (DMR) clock-tree, or a background scrubber to preserve the integrity of the FPGA memory configuration.

The FPGA market was higher than 6.2 billion dollars in 2021, but the FPGA space segment just represents the 3.7% (0.23 billion dollars, mainly for Xilinx and Microsemi devices) [37]. Therefore, the availability of FPGAs in the ground segment is very wide and FPGAs have much lower cost and higher computational capabilities in comparison with RHBD FPGAs. This is the reason why during the last years COTS FPGAs (mainly SRAM-based) are gaining interest for space applications (specially as for short-time and low-orbit missions). Space qualification process takes a long time (commonly years) and it is costly, implying radiation campaigns in specialised facilities. Sometimes, the mission design process cannot wait this long time to decide the final device to be included in the hardware processing payload, so a decision must be made taking into account mission requirements (e.g., orbit, lifetime, etc.).

Nevertheless, radiation can cause bit flips in SRAM-based COTS FPGAs configuration memory, which may derive in a modification of the design behaviour or even in a progressive malfunction of the whole device, until reconfiguration is performed [38]. Mitigation techniques must be introduced in SRAM-based COTS FPGAs to be used on space missions, since they are not inherently robust to radiation. Some techniques, such as scrubbing and hardware redundancy at different levels, are implemented as part of the mapped design onto the FPGA to prevent or mitigate radiation effects and to preserve both data integrity and system functionality. FPGAs can be also combined with microprocessors in the same die, forming heterogeneous SoCs, which increase system capabilities in terms of computational performance. To demonstrate their viability for short-time scientific applications, the space industry is testing in-orbit these SoCs in SmallSats for Low-Earth Orbit (LEO) missions, where the radiation effects are reduced [39].

The compression solutions presented in this Thesis target the main space-qualified FPGAs available in the market. Hence, the design of these solutions will be technology-agnostic, when possible, but architectural approaches developed for each compression solution will be focused on efficiently exploit the strengths of these FPGAs, providing high performance and reduced hardware occupancy. Special emphasis will be done on designs with NanoXplore BRAVE family, since it is the European approach to FPGA space market and is widely supported by ESA. In addition, it is remarkable that the implementation results provided for the different compression approaches presented throughout this Thesis are the first in the state-of-the-art for this novel family of devices.

1.2.2 Design methodology

The FPGA design flow for the ESA missions has been addressed mainly by the ECSS-Q-ST-60-02C *ASIC and FPGA development* regulation [40]. This norm is based on a Register-Transfer Level (RTL) methodology with a classical design approach using Hardware Description Languages (HDL), such as VHDL or Verilog, to describe a synchronous digital circuit [41]. The gate-level equivalent netlist is obtained by performing a logic synthesis using the appropriate EDA tools. Logic synthesis preserves the architectural details of the RTL description, at the same time it tries to meet the required target performance. A relevant aspect of the RTL descriptions is that they could be done as technology-independent, getting rid of information about specific technology or components libraries. In addition, the designer can provide information to the logic synthesis tool, commonly denoted as synthesis directives, that can help to achieve the desired performance, minimizing synthesis and place-and-route iterations.

Using the RTL methodology, the characterization and assessment of the final performance of the circuits is not a difficult task. However, verification is a huge problem specially when the design complexity increases. Further problems are also the design time, the impact of possible design re-spins and the emerging interest of using heterogeneous SoCs, which extend system capabilities in terms of computational performance.

For this reason, early modelling the functionality of the electronic system at a higher level of abstraction is a must for many ESA projects, allowing to speed up the design time as well as the verification, and reducing the re-design costs of complex electronics systems. ESA has tried to support this functionality creating specific virtual platforms (e.g. ESA SoCRocket Virtual Platform [42]), which allow to validate the design at early stages of the development flow.

The High-Level Synthesis (HLS) design methodology, whose workflow is summarised in Figure 1.2, provides an alternative to model complex electronic designs at a higher level of abstraction than RTL. HLS starts from an algorithmic model of the functionality in a high-level language, either with cycle-accurate information (i.e., using SystemC) or without it (e.g., by employing hardware-friendly C/C++ descriptions), introducing an extra layer in the design description hierarchy, to subsequently perform a transformation of the algorithmic behaviour to its micro-architecture, comprised by a datapath and its control unit represented at RTL [43]. Such transformation is automatically done by an HLS tool, which guarantees the correctness of the hardware implementation and should

match the design constraints to achieve the goals in terms of timing, resources and power consumption.

Main advantages of the HLS methodology are that verification is accelerated (although formal verification methods are still not common in HLS tools) and the easy re-spin. An extra reduction of the design time can also be achieved, if existing software descriptions are reused as starting point for the HLS model. Different optimizations can be specified by the user in the existing loops in the source code to accelerate throughput of the obtained solution, such as introducing pipelining or unrolling directives [44]. Moreover, this design flow considers the possibility of manually refining certain parts of the generated RTL model, if particular targets in terms of clock-cycling accuracy are required.

HLS also offers much more flexibility to perform an exhaustive design space exploration, evaluating different architectures at early stages of the development flow, which is specially relevant for highly-parameterizable algorithms [45]. At the same time, HLS allows an easier verification of the different design modules, since they can be treated as isolated units, which are verified independently by using software-based testbenches. These latter features make HLS models an interesting option for prototyping purposes [46], demonstrating

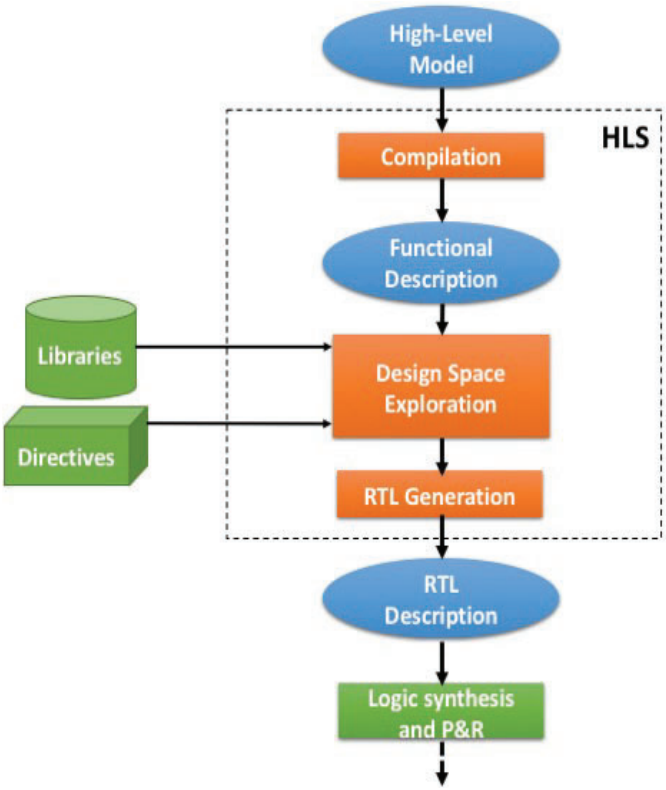


FIGURE 1.2: Overview of the HLS design methodology

the viability of hardware implementations under high timing restrictions and providing a preliminary logic resources utilization.

Currently, there is a wide range of HLS tools available in the market, including company-specific and technology-agnostic alternatives. In the first group, Xilinx Vitis HLS (previously known as Vivado HLS) and Intel-Altera HLS Compiler are the main options, which are fully optimized when targeting FPGA devices from these companies. On the other hand, Mentor Catapult HLS or the open-source Bambu tool, developed at the Politecnico di Milano, provide support for FPGAs from different manufacturers, loading the necessary component models for the target technology [47]. In addition, the well-known mathematical suite MATLAB-Simulink provides a plug-in to generate equivalent RTL descriptions, denoted as HDL Coder, which provides a reasonable performance for low-complexity algorithms.

According to [48], where the authors perform a deep survey by analysing most of the available HLS-related articles in well-known scientific databases, the average development time using an HLS strategy is a third of the equivalent purely RTL description, requiring the latter more lines of codes to implement the same functionality. The results in terms of performance are clearly better for RTL, though differences in terms of maximum clock frequency are minimum. The HLS approaches also consume more logic resources, withy a penalty in average about 41%.

In this Thesis, both RTL and HLS design methodologies have been combined to develop the different stages of the proposed modular on-board compression solutions, attending to the necessities of each particular implementation. The factors that will take into account to decide which design methodology is used in each case are the available development time and the target throughput, imposed by the space mission requirements.

1.2.3 Hyperspectral sensors and image formats

Data acquisition by Remote Sensing hyperspectral instruments is done by collecting the reflected light of the observed scene with an array of sensors, where each sensor records the information of one pixel of the image. This information is not only in the spatial domain, but also considers the spectral reflectance for that given pixel. Thus, hyperspectral instruments must take into account both the spatial and the spectral dimensions, in addition to the temporal domain, since acquiring a full hyperspectral cube generally requires a scanning in a time-frame.

Although there are a variety of methods to acquire the spectral information of each pixel at different wavelengths, two of them are clearly distinguished: dispersive spectrometers and spectral filters. The first type commonly uses a prism that split the light reflected into the different wavelengths, which is coupled to the sensor that collects the pixel spatial information [49]. The dispersive methods are classified in two main types: whiskbroom and pushbroom. Whiskbroom scanners are comprised by a unique detector that receive the light from the prism divided in different wavelengths. A whiskbroom spectrometer acquires the information of one pixel with all the wavelengths at each time. The acquisition of the complete image requires a scanning in two directions. The first one in the cross-track direction to scan all the pixels that form the swath width and the other one in the along-track direction (the satellite flight direction). On the other hand, pushbroom sensors are formed by an array of detectors able to collect one line of pixels at a time in the along-track direction [50]. The overview of both scanning types is shown in Figure 1.3.

The selection of the appropriate hyperspectral scanner is closely related to the mission requirements. Whiskbroom scanners are simpler, since they are comprised by only one detector, but they require of a rotation mirror to scan all the swath width. They have longer swath width compared to a pushbroom one but will have higher requirements in the detector in regards to the capturing time. Pushbroom scanners require a correct calibration between adjacent pixels of one line (they are scanned by different detectors). In recent missions, pushbroom scanners are preferred because it limit the number of moving parts and have higher light integration times.

The second option of spectrometers are based on adding spectral filters before the detector. Spectral filters based hyperspectral sensors are less popular and appear recently. The main idea is to add one or more spectral filters to transmit the selected spectral bands to the detector. The tunable filter could be fixed as simple as a wheel of filters or a linear variable filter (LVF). In both cases, every wheel part of the LVF transmits only a specific wavelength, absorbing the rest. Each filter in a wheel of filters should be changed in a mechanical way to conform the whole hyperspectral cube by multiple exposures to the same scene. A recent alternative is the use of electronically tunable filters, such as Liquid Crystal Tunable Filter (LCTF) or Acousto-optic tunable filter (AOTF), comprised by a unique optical filter whose sensibility to a wavelength is electronically/acoustically controlled by applying a concrete frequency.

An additional alternative is a snapshot imager, which captures the whole scene in the spatial and the spectral domain from a “shot” (i.e., simultaneously), not requiring spatial

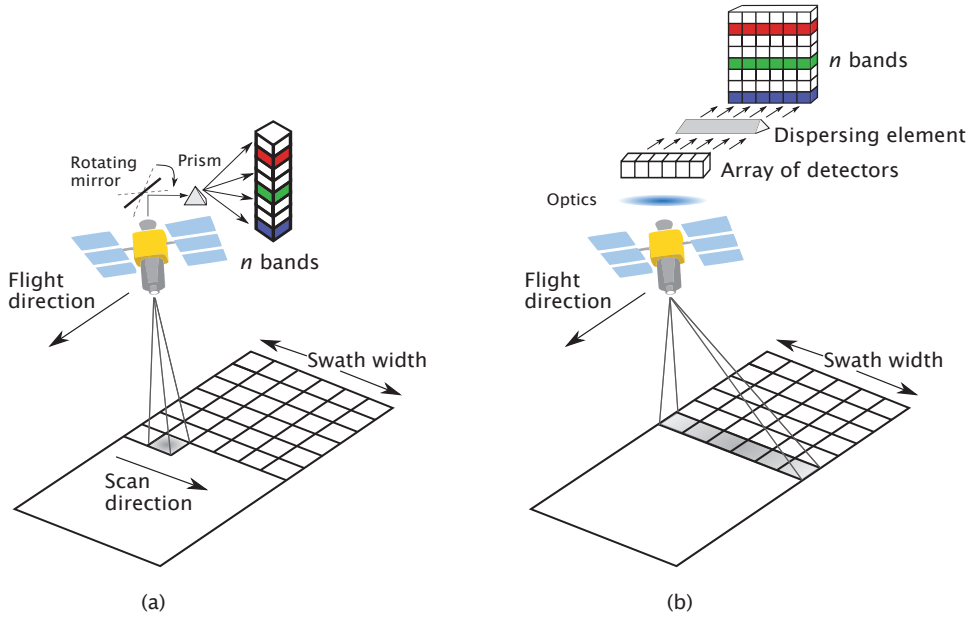


FIGURE 1.3: Hyperspectral scanners. a) Whiskbroom; b) Pushbroom (extracted from [51])

displacements to compose the whole hyperspectral cube [52]. However, it acquires just a reduced number of spectral channels, compared to the whiskbroom and the pushbroom scanners. The sensors employed by snapshot imagers are comprised by a Bayer filter to obtain the different spectral channels after demosaicing techniques [53].

Taking into account the different sensors in the market there are three possible arrangements in the acquired images: a) Band-Sequential (BSQ) order, where the samples in a spectral channel are processed before moving to the next one, typical of the spectrometers based on spectral filters; b) Band-Interleaved-by-Pixel (BIP), where a pixel is processed with all its spectral information (i.e., the different wavelengths) before handling the next one, typical of the whiskbroom spectrometers; and c) Band-Interleaved-by-Line (BIL), in which a complete line of samples in the spatial domain is processed before starting the next band, typical of the pushbroom spectrometers. These samples arrangements are visually summarised in Figure 1.4. The solutions to be provided in this Thesis should work with all kind of image arrangements, with no impact in the system performance. In some cases this can lead to specific architectures for each different sensor or to data input reordering. System storage requirements is also a problem to be deal with because some orders will be more memory demanding than others. This adds extraordinary complexity to obtain efficient and general solutions.

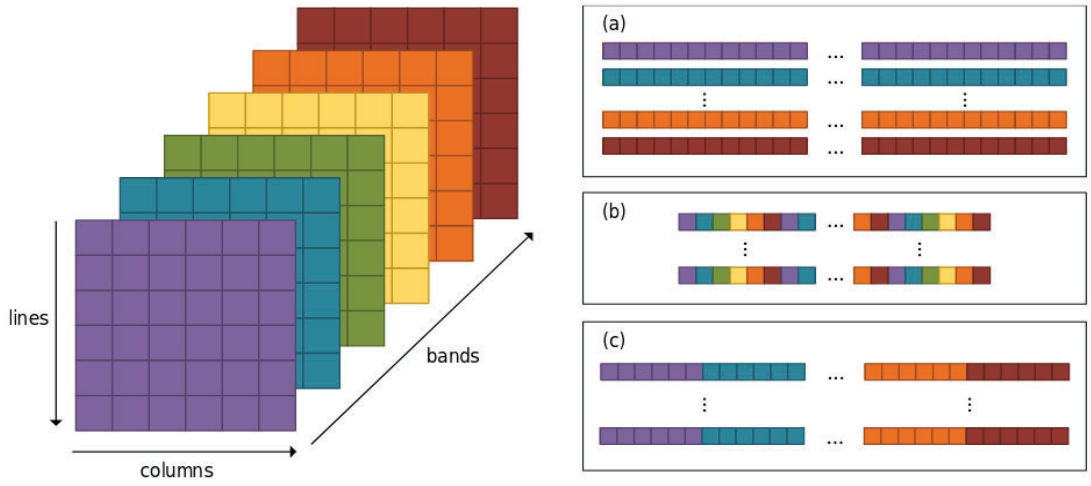


FIGURE 1.4: Samples arrangement in hyperspectral cubes. a) BSQ; b) BIP; c) BIL

1.2.4 Real-time video acquisition on satellites

A traditional approach to acquire video is the use of a panchromatic camera, which only provides gray-scale information (i.e., brightness of the scene, without color) in the spatial and the temporal domain, but not spectral data. Panchromatic information is commonly wide in the visible range of the spectrum, providing both a high spatial resolution and Signal-to-Noise Ratio (SNR). The resulting image/video is generated by using the total light acquired by the sensor, without dividing it into different wavelengths.

Hybrid approaches are also used for video acquisition, combining panchromatic and RGB/multispectral cameras through pansharpening fusion methods to take advantage of the strengths of both devices (high spatial and spectral resolution, respectively) [54, 55].

Video is then obtained by taking consecutive shots of the same scene or leveraging of the satellite movement in its orbit to record a wider scene, which is useful for target detection, object tracking or monitorization of natural phenomena. The size of the scene acquired by the camera in the spatial domain depends on both the sensor size, the focal length and, consequently, the Field of View (FoV), having a wider FoV when the focal length is shorter. The acquisition cadence is measured by using the Frames Per Second (FPS) metric, considering as a frame an individual scene acquisition. Once the video has been fully collected, it must be also preprocessed in a similar way as hyperspectral images, but also dealing with specific situations that only appears in this kind of data, such as temporal correlation or the parallax effect [56].

1.3 On-board data compression

Compression of both hyperspectral images and video sequences is considered an efficient technique for data reduction, since the acquired scenes contain redundant and correlated information that can be removed (i.e., adjacent spectral channels and consecutive frames, respectively). The main goal of compression techniques is to represent the acquired data with the minimum amount of bits, without preventing its reconstruction after the decompression process on ground.

Compression techniques can be either lossless or lossy. Lossless compression preserves all the information presented in the original data, which can be fully recovered during the decompression process. For this reason, lossless compression results interesting for the scientific community since it maintains the fidelity of the data acquired by the sensor. However, the compression ratio (CR, see Equation 1.1) of lossless compression techniques is typically limited from 2 to 4, depending on the data correlation. On the other hand, lossy compression yields higher compression ratios than lossless techniques by introducing losses in the compression chain. As a consequence, the recovered information is not identical to the the data captured by the sensor [57]. Lossy compression is a key enabler for deep-space missions with high-resolution sensors on-board, in order to exchange information with ground.

In an intermediate point we find near-lossless compression, which achieves higher compression ratios than lossless techniques without reaching the levels of lossy compression. Near-lossless compression is a good trade-off between compression ratios achieved, image quality and algorithm complexity. Both lossy and near-lossless approaches can provide an accurate control of the losses introduced. This is achieved by introducing a rate control method, commonly based on a feedback loop, in the the processing chain. This rate control determines the quantization step to be used to process the next group of input samples, taking into account the compression ratio reached for the previous one and the compression target specified by the user.

Regarding the nature of the compression algorithm, two main techniques clearly stand out in the state-of-the-art: prediction-based and transform-based approaches. Compression methods based on transforms reduce the redundant information present in the input data by transforming the information from the spatial domain to an alternative representation, such as the spectral dimension, in order to efficiently decorrelate the data. The transform step is typically followed by a quantization and encoding stages. The quantizer is included

since transform-based approaches are generally used for lossy compression, though some reversible transforms can also be used for lossless compression. Transform-based methods are widely used in ground applications, because of the high compression ratios they achieve. On the other hand, prediction-based solutions are based on a calculation of the current pixel value like a weighted sum of the pixels in the spatial, spectral or temporal neighbourhood. Although the compression ratio achieved is worse than transform-based algorithms, prediction-based approaches provide a good trade-off between compression performance and algorithm complexity. For this reason, they are used as alternative to transform-based in many applications that require low-complexity compression solutions, and specifically on-board satellites. Both compression methods are further explained in Section 2.3.

The compression efficiency, which depends on both the compression method and the nature of the input data, is measured by using the CR, which is the relationship, represented in bits, between the size of the acquired information (i.e., raw data) and the resulting size of the compressed bitstream (see Equation 1.1). Alternatively, the compression performance can be expressed in terms of bits per pixel (bpp) when referring to image and video compression. This metric is obtained following Equation 1.2, and it takes into account the total number of pixels of the image or video.

$$CR = \frac{\text{Size of the acquired data (bits)}}{\text{Size of the compressed data (bits)}} \tag{1.1}$$

$$bpp = \frac{\text{Size of the compressed data (bits)}}{N_{pixels}} \tag{1.2}$$

In addition to the CR, different metrics have been defined to evaluate the performance of the compression techniques, specially to analyse the impact of introducing losses when lossy methods are used. A widely used metric is the Rate-Distortion (RD) ratio, which is the relationship between the SNR or the Peak-Signal to Noise Ratio (PSNR) and the CR. Other extended metric is the Mean Squared Error (MSE), which measures the cumulative squared error between the compressed and the original data, so the lower the MSE value, the lower the error. Equations 1.3 and 1.4 reflect how the MSE and the PSNR are calculated, respectively:

$$MSE = \frac{\Sigma \mid s(t) - \hat{s}(t) \mid}{\text{Size of the acquired data (bits)}} \tag{1.3}$$

$$PSNR = 10 \cdot log_{10} \frac{(2^D - 1)^2}{MSE}, \tag{1.4}$$

where $s(t)$ represents the sample processed at time t, $\hat{s}(t)$ is its reconstructed equivalence and D the bit-depth of the input samples.

In space applications, other relevant aspect that should be taken into account is the robustness against errors during the compression or when data is transmitted to ground. Different strategies can be adopted to provide robustness to the compression process, such as data segmentation, avoiding error propagation from one portion of data to others in case of corruption. In this sense, only a data segment is lost, making possible to correctly reconstruct the rest of the compressed bitstream.

1.4 Thesis framework

This Thesis have an industrial oriented scope. The objectives, methodology, algorithms, architectures and constraints of each proposed solution are determined by current and future projects related to acquisition and processing of different kind of information on-board satellites.

The research on compression of hyperspectral images for on-board satellites started with the European Spatial Agency funded SHyLoC project, developed by the Group between 2015 and 2021 [58–60]. The goal of this project was to develop a pair of reusable IP Cores fully-compliant the CCSDS 121.0-B-3 and 123.0-B-1 lossless compression standards. The former corresponds to a a Lossless Data Compressor, while the latter is Lossless Multispectral and Hyperspectral Image Compression architecture. This project could be considered a landmark that impacted all later developments. The work presented in this Thesis is partially an extension of these early developments. Some of the interfaces, architectures or configurations are inherited from this project.

Other important project is CHIME [61], where the Group participated in the pre-development stage (phase A/B1) of the mission. In this stage, the first hardware implementation of the CCSDS 123.0-B-2 standard available in the state-of-the-art was proposed by following an HLS design flow, working together with a cloud detection algorithm [62]. Phases B2, C/D and E1 of this space program are currently under development, refining the

proposed CCSDS 123.0-B-2 compression architecture at RTL and adapting its performance for the specific mission constraints.

SHyLoC and CHIME projects have been done in collaboration, among others, with Thales Alenia Space in Spain (TASiS) and Thales Alenia Space in France (TASiF), branches of the European leader in satellite manufacturing.

The contributions presented in this Thesis regarding video compression on-board satellites have been developed in the context of the H2020 EU-funded *Video Imaging Demonstrator for Earth Observation* (VIDEO) project, which is focused on the development of a next-generation instrument for Earth observation [63]. This instrument will have the capability to perform high-resolution video monitoring on an extremely wide scene, with the purpose of recognizing and tracking objects.

The work presented in this Thesis is a continuation of the research developed in two previous theses of the Group. In [51], a solution was given to provide efficient lossy compression for hyperspectral images on-board satellites. In addition, a first implementation of the CCSDS 123.0-B-1 lossless compression standard on FPGA is proposed, prioritizing low hardware occupancy. Then, in [64] the HLS design methodology is introduced to quickly develop functional solutions for lossy compression of hyperspectral images on-board satellites. One of these solutions takes the CCSDS 123.0-B-1 lossless compression standard as starting point, introducing the necessary steps (i.e., a quantizer and a bit-rate control) to perform near-lossless compression.

As a general framework, other research of the Group should be mentioned regarding low-complexity hardware implementations of compression techniques, specifically targeting hyperspectral images acquired on-board satellites [65–67].

1.5 Objectives of the Thesis

As it has been mentioned throughout this chapter, next-generation sensors integrated in future space missions will acquire such quantity of data that on-board compression will be mandatory to maintain high acquisition data rates and overcome the restrictions in terms of available storage and downlink bandwidth. On-board compression requires not only efficient algorithms that provide the desired ratio and reconstructed quality, but also a suitable physical implementation on the available hardware that preserves the correct behaviour of the implemented approach, which should coexist with other functionality in the same payload.

We have seen that the quantity and types of sensors on-board a satellite are diverse and the solutions for data compression should be easily adapted or configurable to each particular mission. The limited on-board memory and resources will require compression of data on the fly (i.e., in real-time) and thus specific low-complexity algorithms for on-board compression. Although the solutions adopted are technology-agnostic, it will be demonstrated on FPGAs, since they are an interesting platform for current and future missions. This will imply the development of hardware architectures that exploit efficiently the capabilities of FPGAs.

The use of FPGAs will speed-up the design process of the payload processing functions in the PFCU. In this sense, the use of high-level methodologies will accelerate even more the design and verification process. Although the use of high-level methodologies is not extended on space applications, this Thesis will investigate the use of HLS when the performance and the resources consumption required by the solution is considered adequate for the target application.

The main objective of this Thesis is to provide modular solutions that can be adapted to compress data acquired by high-resolution sensors on next-generation space missions from different nature, including generic information, multi- and hyperspectral images, and video sequences.

The flexibility provided by this modular scheme also allows to change the functionality of the compression solution if new requirements appear during the mission lifetime. This is achieved by replacing the appropriate processing stages to reduce hardware occupancy and power consumption, to increase compression ratio or to accelerate throughput.

The specific objectives of this Thesis are detailed next:

- Analyze the solutions provided by the CCSDS for compression of one-dimensional data, 2D images, 3D multi- and hyperspectral images and video sequences. This analysis will determine the best solutions to be implemented on space applications, taking into account their complexity and the expected performance.
- Develop a design space exploration of the CCSDS algorithms object of study, providing architectural alternatives for their implementation, depending on the target requirements in the real scenario.
- Provide efficient and modular solutions for high-throughput and low-occupancy data compression on space-grade FPGAs. The focus is on the proposed CCSDS algorithms for one-dimensional data and multispectral/hyperspectral image compression due to their interest for future space missions: the CCSDS 121.0-B-3 [27] and the CCSDS 123 standard, including Issue 1 (only thought for lossless compression) [30] and the recent Issue 2 (extending the functionality for near-lossless compression) [31].
- Propose the use of different design methodologies, such as RTL and HLS, depending on the space mission constraints in terms of development time and performance. The obtained results will be studied to remark the weaknesses and strengths of the design methodologies followed.
- Propose different hardware configurations, depending on the compression requirements and the nature of the data to be compressed. Following this approach, multiple compression chains can be defined by reusing modules that perform the prediction and the entropy coding stages in an efficient way, taking into account data nature and the final application.
- Demonstrate that the solutions are viable for a number of scenarios when implemented on space-grade FPGAs. This will be done by performing an exhaustive verification and also validation on-chip.

1.6 Document structure

The present document is structured in six chapters, including this introductory one, dedicated to introduce the context of this Thesis and to present the main motivations and goals. The rest of the chapters are briefly described next.

1.6.1 Chapter 2: Satellite data compression algorithms and their hardware implementation

This chapter summarises the state-of-the art in the field of algorithms for data, 3D imaging and video compression, including also an study of their complexity to be implemented on hardware. Compression algorithms are detailed and classified according to their purpose and nature, and different hardware implementations are analysed. In this way, the work done in this Thesis is contextualised.

1.6.2 Chapter 3: FPGA implementations of prediction-based preprocessors for data decorrelation

In this chapter, different prediction-based approaches are presented as efficient architectures to be implemented on space-grade FPGAs to serve as spatial and/or spectral decorrelator of the data acquired. The alternatives under study are based on the CCSDS 121.0-B-3, 123.0-B-1 and 123.0-B-2 compression standards, providing an algorithmic background, an architectural description and preliminary synthesis results for each proposed solution.

1.6.3 Chapter 4: Low-complexity hardware solutions for entropy coding

Following a similar scheme that Chapter 3, this chapter describes different entropy coding alternatives that allow to reduce the bitstream size by representing prediction residuals with the lowest possible number of bits, ensuring at the same time a proper reconstruction on the decompression side. The proposed hardware implementations, which are the block-adaptive, sample-adaptive and hybrid encoders, are also based on the CCSDS 121.0-B-3, 123.0-B-1 and 123.0-B-2 compression standards, respectively. For each encoding option, the theoretical basis is provided, followed by a detailed description of the developed hardware implementation and some preliminary results in well-known space-grade FPGAs.

1.6.4 Chapter 5: Modular solutions for on-board data compression

This chapter is considered the core of this Thesis, since different modular solutions are presented for on-board data compression on real space missions, providing different approaches depending on target application requirements, such as acquired data nature, compression performance, hardware occupancy, throughput or development time. The proposed solutions combine a prediction-based preprocessor and an entropy coding stage, presented in previous chapters, to comprise a full compression chain. The different alternatives are compliant with the CCSDS standards and ensure a correct behaviour on hardware available on-board satellites. In addition to the compression chain structure, detailed results are provided to quantify the viability of the presented solutions in a real scenario. Finally, a performance assessment is included to analyse the possibility of using the CCSDS 123.0-B-2 compression algorithm to process panchromatic and RGB video in future space missions, obtaining a versatile solution that can compress different data depending on mission necessities.

1.6.5 Chapter 6: Conclusions

In the last chapter, all the contributions of this Thesis are summarised and the achieved goals are also remarked. In addition, further research lines are proposed, which arise as an extension of the presented work.

Chapter 2

Satellite data compression algorithms and their hardware implementation

This Chapter provides an overview on the main compression techniques employed in Remote Sensing applications and, concretely, in space missions. A description of available solutions in the state-of-the-art and CCSDS space standards for on-board data, hyperspectral image and video compression is provided, including the theoretical basis and the use in space missions, if applicable. Finally, hardware implementations of the analysed compression solutions are presented for different technologies, including space-qualified electronics and focusing on FPGAs.

2.1 Outline

Developing efficient compression solutions for space poses a challenge, since the employed algorithms must achieve the goal of reaching a high compression ratio together with an acceptable quality after decompression. Moreover, these algorithms need to have a low complexity to be executed on the available hardware resources on-board satellites and the meeting required timing performance for a given mission. This challenge has motivated academia and the space industry to conduct research on data, image and video compression techniques for space applications. A summary of this research can be found in the scientific literature elsewhere (e.g. [68, 69]).

The most relevant compression algorithms targeting space missions are described throughout this chapter, as well as other compression techniques commonly used on ground, which are potential candidates to be implemented on-board in future missions provided some adaptations. Compression techniques proposed by the CCSDS are also introduced. Since some of these standards are the core of the compression solutions proposed in this Thesis, they will be thoroughly described in Chapters 3 and 4.

In general, compression algorithms are computationally complex, so they need to be adapted in order to fit well in on-board processing systems. Therefore, in addition to the compression methods, the available on-board hardware must be analysed, taking into account certain constraints that appear in space missions, such as area, power consumption and robustness against radiation effects. For this reason, the most relevant hardware implementations of compression algorithms targeting space applications are also presented in this chapter, highlighting their main features in terms of performance, hardware occupancy and power consumption.

2.2 One-dimensional data compression algorithms

Different methods have been used on space missions to compress one-dimensional data from different nature, including telemetry, geolocation or spectrograph measurements [70]. In a nutshell, one-dimensional data compression algorithms are based on removing the redundant information acquired and/or produced by the satellite, and representing the decorrelated data with the minimal number of bits without preventing a correct reconstruction on ground.

Differential Pulse Code Modulation (DPCM) [71] is widely employed for lossy data compression. Although there are other alternatives to implement decorrelation stages on a data compression solution, DPCM is commonly used because it provides an acceptable compression performance with low algorithmic complexity. This solution is comprised by a linear predictor (i.e., estimating the value of the predicted sample by subtracting the value of previously preprocessed samples) for data decorrelation and a uniform quantizer. Lossless compression is also possible, if the quantizer is bypassed in the compression flow. Its main advantage is a reduced computational burden, and the fact that it is also possible to pipeline hardware implementations for parallelization purposes.

Regarding entropy coding stages, the use of coding redundancy techniques is also extended, with or without a preceding decorrelation stage in the compression chain. In this approach, a variable-length coding strategy is used, assigning the shortest codewords to the most frequent symbols, and vice versa. The most common entropy coding methods based on coding redundancy are the next:

- **Huffman coding**. Its basic idea is to exploit data statistics to assign variable-length codewords to an alphabet of symbols with a known probability distribution [72]. Huffman coding first constructs an optimal code tree, with the highest probability at the top and the lowest at the bottom, producing a node set with this information that is constantly updated with new entries (normally, the sum of two existing probabilities).

- **Golomb-Rice coding**. Golomb codes are based on the division between the input sample s and a tunable parameter, obtaining in this way a quotient q and a reminder r. The codeword is finally generated by concatenating q 'ones', followed by a stop bit and the r least significant bits of s. Rice codes work in a similar way, but using power-of-two values as tunable parameter for the division computation [73]. Rice coding is the basis of the CCSDS 121 universal lossless compression standard [27], where these codes are calculated simultaneously for a group of input samples, selecting the one that provides the shortest codeword. The use of Rice codes eases hardware implementations, since multiplication and division by a power-of-two can be easily implemented using logic shifts.

- **Run-Length Encoding**. This approach replaces chains of repeated consecutive bytes by a counter that represents the number of replays and the symbol under coding [74]. However, this method achieves low compression ratios, if the data to

be compressed do not contain long series of repeated symbols, though its simplicity eases its hardware implementation with a reduced memory consumption.

- **Arithmetic coding**. This technique achieves higher compression ratios than Huffman coding, at the expense of higher complexity. It can be seen as a simple clustering algorithm, dividing the interval of an input word, initially defined between 0 and 1, into smaller segments which represent the probability of the word symbols. Then, each new input symbol is assigned to an interval, repeating this procedure iteratively until reaching the last input word [75]. The intervals are narrowed with every symbol included, finally selecting any value inside the final interval to represent the input word [76].

A common one-dimensional data compression approach is to combine DPCM with a redudancy-based entropy coder stage in order to increase compression performance. For example, DPCM combined with a Huffman encoder (Figure 2.1) is one of the most common techniques used on-board satellites for data compression [70].

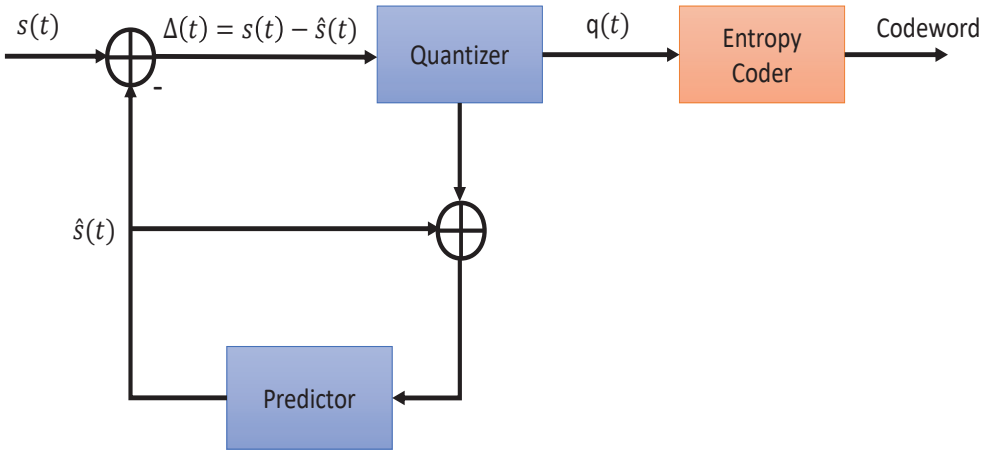


FIGURE 2.1: Overview of a DPCM architecture combined with a Huffman encoder

The Fully Adaptive Prediction Error Coder (FAPEC) [77] is a low-complexity data compressor specifically thought for space missions, which is able to achieve high compression ratios under undesirable situations, such as the presence of noise or outliers. It can be combined with a pre-processing stage to obtain higher performance. It is based on a segmentation strategy, working with the statistical analysis of small data blocks and providing in this way good adaptation to changing data statistics and robustness against outliers. FAPEC generates a variable-length code for each prediction error, taking into

account the coding tables previously created. It is an alternative to the Golomb-Rice coder with the maximum code length limited to less than twice the bit length of the input values.

Another example is the Context-based, Adaptive, Lossless Image Codec (CALIC) [78], which uses several modeling contexts to characterize a non-linear predictor that includes an error feedback mechanism for correcting itself, taking into account a given context previously employed.

Additionally, there are other well-known algorithms which can be used for one-dimensional data compression, such as the Lempel-Ziv-Welch (LZW) [79], which initializes a dictionary with 256 single characters and the string table of encoding. Then, data are read and matched with the existing entries in the dictionary until a mismatch takes place. The code of the entry which matches successfully is outputted, while the entry which mismatches is added to the dictionary. The number of entries increases rapidly, and so does also the matching probability. The dictionary is reset when more than 4096 entries have been stored during the encoding process. LZW presents certain advantages when compared with Huffman coding, since the dictionary is created on-the-fly at the same time the encoding is performed, what makes it suitable for applications with real-time constraints, at the expense of a high memory usage [80]. Other techniques, such as data subsampling, which quantizes the data by deleting part of it, encoding then the quantized residuals [70], can be also used for on-board one-dimensional data compression, depending on the specific constraints of the space mission.

2.3 Hyperspectral image compression

Currently, multiple solutions can be found in the specialized literature to compress hyperspectral images. The main feature of any hyperspectral compression technique is to exploit redundancies among adjacent samples in both the spatial and the spectral domains, with the goal of reducing the data size. In this sense, two different perspectives are viable: 2D coding, which only takes into account redundancies in the spatial or in the spectral dimension [81–85]; or 3D coding, which considers redundancies in both domains, providing higher compression ratios than the 2D methods. For this reason and their relevance for this Thesis work, only 3D algorithms will be analysed in the rest of this chapter. These algorithms can be classified as lossless, near-lossless and lossy techniques.

In general, a lossless hyperspectral image compression algorithm is comprised by two main stages: a spatial and/or spectral decorrelator and an entropy coder. In addition, a quantizer and even a bit-rate control are included, if near-lossless or lossy compression is required. Extra pre-processing stages, such as band reordering, can be incorporated to increase the correlation between consecutive bands, obtaining higher compression ratios at the expense of additional computational burden [86, 87].

It is also desirable that, in addition to a low complexity architecture to fit well with the hardware available on satellites, the compression algorithms have certain robustness against errors induced by radiation (e.g., bit flips in the FPGA configuration memory). Most times, a bit error in a corrupted compressed pixel causes a global malfunction during the decompression stage, since there are dependencies between consecutive image samples. For this reason, image compression algorithms should limit the error propagation, avoiding that an error in a certain pixel will be propagated to others, finally degrading the whole image. Techniques such as image segmentation can prevent that the quality of the reconstructed image is fully degraded. However, the obtained compression performance should be evaluated when compression is independently applied to each image segment. This analysis is needed to guarantee that this segmentation strategy does not imply a penalty in terms of CR, compared with the one obtained when compressing the full image in a single step.

Although there are different methods for hyperspectral image compression, such as vector quantization [88–91], or the more recent compressive sensing [92–95] and learning-based [96–99] techniques, algorithms based on prediction or transform as decorrelating stages prevail in the specialized literature. Relevant solutions into these categories are detailed below.

2.3.1 Prediction-based algorithms

Predictive algorithms are preferred for lossless compression, since they have lower complexity than transform-based approaches, providing at the same time acceptable compression ratios [100]. Besides, they can be extended to obtain low-complexity solutions to compress in a near-lossless to lossy range, by including a quantizer stage and/or a bit-rate control, which allows to adjust the compression ratio taking into account the target performance and the real-time statistics [101–103]. As entropy coder, current on-board implementations

normally employ a coding redundancy technique, such as Huffman and specially Rice coding, because of the provided trade-off between complexity and compression performance.

Among the predictive compression methods available in the state-of-the-art, the Fast Lossless (FL) algorithm [104] emerges as a reference for on-board lossless compression, since it is the basis of the CCSDS 123.0-B-1 standard [30]. This low-complexity solution calculates the predicted sample for the current pixel taking into account a vicinity in both the spatial and the spectral domain, adapting the prediction statistics after a sample is processed based on the prediction error. FL is based on the sign algorithm, a variant of the Least Mean Square (LMS) method [105]. Recently, the FL algorithm has been extended to perform near-lossless compression, denoting the new solution as Fast Lossless EXtended (FLEX) [106]. The FLEX algorithm, which is also the basis of the recent CCSDS 123.0-B-2 standard for hyperspectral image compression [31], introduces two new elements in the prediction loop: a uniform quantizer and the Sample Representative stage (also denoted as local decompressor), which approximately reconstructs the original samples after the quantization step, in the same way that the decompressor does. This solution allows to limit the maximum error introduced in the compression chain by some user-specified parameters. Other new features introduced regarding FL are a local sum mode which does not take into account the sample at the left of the current one in the same band, favouring pipelining in hardware implementations; and a new encoder, named hybrid encoder, which provides a better response for low-entropy data than the sample-adaptive and the block-adaptive encoders, defined in the CCSDS 123.0-B-1 [30] and the CCSDS 121.0-B-3 [27] standards, respectively.

Besides, some algorithms are proposed for hyperspectral image compression based on CALIC, previously defined in Section 2.2. 3D-CALIC [107] is an extended version of CALIC for working with multispectral images, switching between intra-band (i.e., working in the spatial dimension) and inter-band modes, exploding in the latter case the strong correlation between adjacent spectral bands. It was observed that the compression performance of the 3D-CALIC solution decreases when the number of spectral channels used for the prediction increases, not being suitable to compress hyperspectral images. A more recent version of CALIC, known as M-CALIC [108], improves the compression results obtained by the 3D-CALIC solution, using only inter-band compression by taking advantage of the high wavelength resolution of hyperspectral images, which increases correlation between consecutive bands. Since spatial decorrelation is not considered, M-CALIC also provides a simpler solution than 3D-CALIC, fitting better on limited hardware resources.

Predictive algorithms based on Look-Up Tables (LUTs) [109, 110] search try to identify in the previous band z a pixel with similar value to a spatial neighbouring of the sample under processing. Then, the sample in the same spatial position of the one found in the previous band is used to predict the value of the current sample. The name of the algorithm derives from the use of Look-Up Tables to accelerate the search. This method is interesting when input images are in BIL order, outperforming other compression standards in the state-of-the-art. However, it was demonstrated in [111] that the LUT-based algorithms work extremely well for calibrated data because they exploit artificial regularities that are introduced during the calibration. Thus, LUT-based methods do not work as well when they directly compress raw data or when new calibration methods are applied.

Mielikainen et. al. also proposes in [112] the use of clustered DPCM with adaptive prediction length, using the k-means clustering to assign each spectral vector (i.e., a band with all the pixels in the spatial dimension) to the cluster whose center is the nearest. Then, a linear prediction is applied to each cluster and finally the prediction residuals are encoded using an adaptive range encoder.

An alternative prediction-based scheme to compress hyperspectral images is proposed in [113], but using time-lapse HSI data. This approach, based on the FL algorithm, can be considered an intermediate point between hyperspectral image and video compression, since input data are consecutive hyperspectral cubes captured in a reduced time-frame, so the temporal correlation is also exploited to reduce data size. The efficiency of the proposed method depends on the acquisition time-frame, decreasing when temporal and/or spectral frames are weakly correlated.

2.3.2 Transform-based solutions

Transforms, which provide good performance in decorrelating in the the spectral domain, are normally combined with a spatial decorrelator to compress hyperspectral images, since they need to be extended to work with 3D data.

The Discrete Cosine Transform (DCT) is a technique widely used in image compression and included in some image and video standards, such as JPEG [114] and H.264 [115], respectively. The DCT is applied to an image in a block-by-block basis, representing it as a superposition of cosine functions at different frequencies, also denoted as DCT coefficients, which measure the contribution of the cosine function at those frequencies.

Larger and uniform blocks provide higher efficient coding and, consequently, higher compression performance, increasing at the same time the computational burden and the image degradation. Furthermore, the Discrete Wavelet Transform (DWT) and other Wavelet-based approaches emerge as a reference for image compression, which provide higher data reduction and image quality than the DCT by applying the transform to the whole image instead of following a block-by-block scheme [116]. Nonetheless, the DWT is more complex than the DCT from an implementation point of view, making difficult its implementation on-board satellites. As an example, the well-known JPEG2000 image compression standard employs a Wavelet transform, instead of the DCT used by its predecessor [117], deleting the blocking artifact and allowing the definition of ROIs (Regions of Interest), solving in this way the main disadvantages of the original JPEG.

Another alternative is the Karhunen-Loève Transform (KLT) [118], which provides best results in terms of rate-distortion among different transform-based alternatives [119]. Despite the suitable results presented by the KLT approach, it has some disadvantages, such as a high computational and memory demands, together with high implementation costs and its lack in terms of scalability, preventing its use for specific applications with strong constraints, like on-board compression. The Pairwise Orthogonal Transform (POT) derives from the KLT but reducing the complexity of the operations and still obtaining better results than the Wavelet-based approaches [120]. The Principal Component Analysis (PCA) has been also considered as spectral decorrelator for some hyperspectral image compression solutions [91, 121, 122]. The behaviour and complexity of the PCA is similar to the KLT (both represent the image as a sum of orthonormal and uncorrelated components, then exploiting the correlation among them), but the former preserves most of the data integrity after its reduction, if some assumptions are done in the model. Other examples of transform-based compression solutions are the HyperLCA algorithm [65], which focuses on low-complexity to fit well on limited hardware resources at the same time that high compression ratios are achieved; or those based on partitioning in hierarchical trees [85, 123–125].

The CCSDS has also defined the CCSDS 122.1-B-1 spectral preprocessing transform for 3D image compression standard [29], whose general overview is shown in Figure 2.2. This algorithm is able to work in both lossless and lossy compression modes, though it presents a better behaviour in the latter case. The standard defines four different spectral transforms as decorrelator, whose selection depends on the target sensor and the hardware processing capabilities. These alternatives are the identity transform, the most simple option that gets rid of the upshift stage; the Integer Wavelet Transform (IWT), a low-complexity option

that provides a compression improvement over the identity transform; the POT, which achieve higher compression ratios than the IWT, at the expense of higher implementation complexity; and the Arbitrary Affine Transform (AAT), which uses one-dimensional affine operations to obtain a transformed image.

The resulting image after the spectral transform is compressed by multiple instances of the CCSDS 122.1-B-1 2D encoder (one per transformed band), controlling the image quality and the compression performance via user-defined parameters. The outputs of the different encoder instances are finally grouped and interleaved to generate the compressed bitstream. Additionally, two extra stages, denoted as upshift and downshift (i.e., multiplication and division by power of two, respectively), are included to achieve a trade-off between compression performance and image quality, limiting at the same time the maximum bit-depth of the input image to the 2D encoders. It is recommended to not include these stages if lossless compression is selected, since they degrade the compression performance for this particular case [126].

Finally, an interesting research line is to evaluate the feasibility of employing video compression algorithms to compress hyperspectral images [127]. The main idea is to tune the most used video encoders to obtain a reduced set of options and the suitable parameter values to compress hyperspectral images in an efficient way, without incurring in a high complexity, keeping in mind the main constraints of the video compression algorithms to be implemented on hardware. As hyperspectral images are commonly used on-ground for scientific purposes, lossless compression is preferred to preserve all the data content.

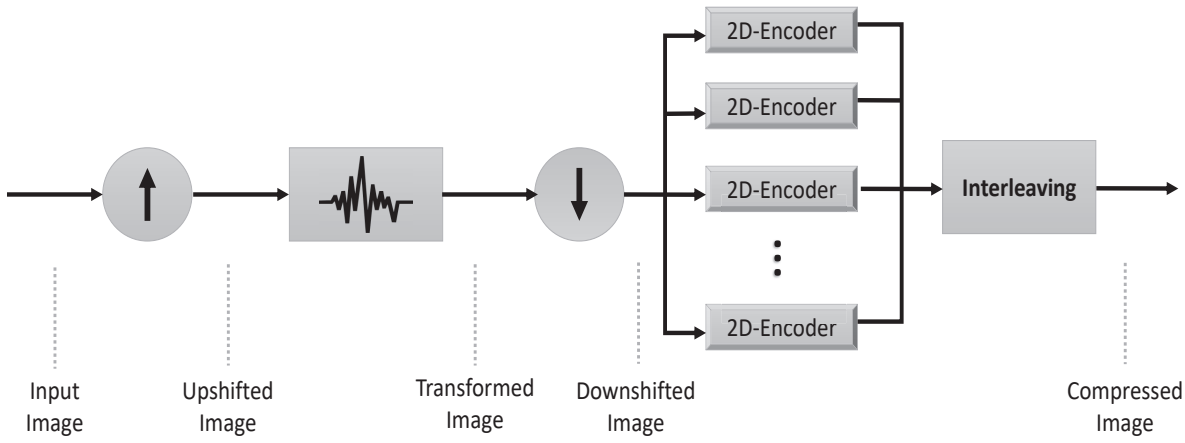


Figure 2.2: CCSDS 122.1-B-1 standard - Block diagram

However, video compression is generally lossy, reason why different works analyse the impact in post-processing applications, such as feature unmixing [128] or classification [129], of using video coders to compress hyperspectral data, demonstrating the viability of these developments and their good results in terms of compression ratio, but at the expense of a high architectural complexity.

2.4 Video compression

Video compression has been employed in the last decades for ground applications such as TV signal broadcasting or multimedia content streaming through Internet. Nevertheless, hardware resources embarked on satellites did not have the capability to manage and process the high volume of data produced by a video sensor in real-time, preventing its use in EO missions for applications such as monitorization. This is because commercial video compression algorithms are not low-complexity oriented and it is many times not feasible to implement them on-board satellites. Selecting a suitable video compression algorithm is essential to reduce the data acquired without degrading the video quality, but guaranteeing at the same time that unique constraints present on space missions (i.e., reduced computational burden, low power consumption, real-time capabilities) are met.

Currently, there is not a standard solution specifically developed for on-board satellite video compression, beyond the adaptation of common commercial video encoders, such as the Advanced Video Coding (MPEG-4 AVC) or the most recent High Efficiency Video Coding (HEVC), commonly denoted as H.264 [115] and H.265 [130], respectively.

H.264 has been the reference among commercial video compression solutions during the last decades, providing a universal solution for a wide range of video applications. The standard defines different stages or layers, but taking into account the scope of this Thesis, we focus on the Video Coding Layer (VCL), the one responsible of processing the video content. The VCL follows a block-based coding approach, dividing a frame into macroblocks (MBs from this point forward) with associated luma (i.e., brightness) and chroma components, Cb and Cr, which are the color deviation from gray toward blue and red, respectively. H.264 allows to partition a frame into fixed-size and rectangular MBs, with a size of 16x16 samples for the luma component and 8x8 samples for each one of the chroma components.

The H.264 VCL presents the general structure shown in Figure 2.3. The standard exploits both the temporal correlation between consecutive frames and the spatial correlation

among adjacent samples in a frame by using inter- and intra-prediction stages, respectively, which are then encoded by applying a transform-based approach. Then, a quantization and a final entropy coding stage are applied.

The intra-frame prediction exploits the spatial correlation among adjacent MBs. The process is performed using neighbouring pixels of previously processed MBs at the left and/or above the MB under analysis, in contrast with previously issued video standards, which work in the transform domain. In the inter-prediction stage, the predicted value is obtained by displacing a concrete MB of the reference frame, specified by a motion vector. In H.264, motion compensation has a maximum precision of a quarter of distance between luma samples, representing an improvement regarding previous standards [131]. Motion vectors can also point outside the frame area, which is supported by repeating the edge samples and extrapolating the boundaries of the reference image. Multi-frame motion prediction is also supported, alllowing the use of more than one reference frame.

After the integer transform stage, a uniform quantization takes place by tuning the Q parameter, which increases the quantization step by a 12% each time it is increased by 1. The theoretical basis of these stages is deeply explained in [132]. The deblocking filter corrects the blocking effect, one of the main artifacts of the encoding process when

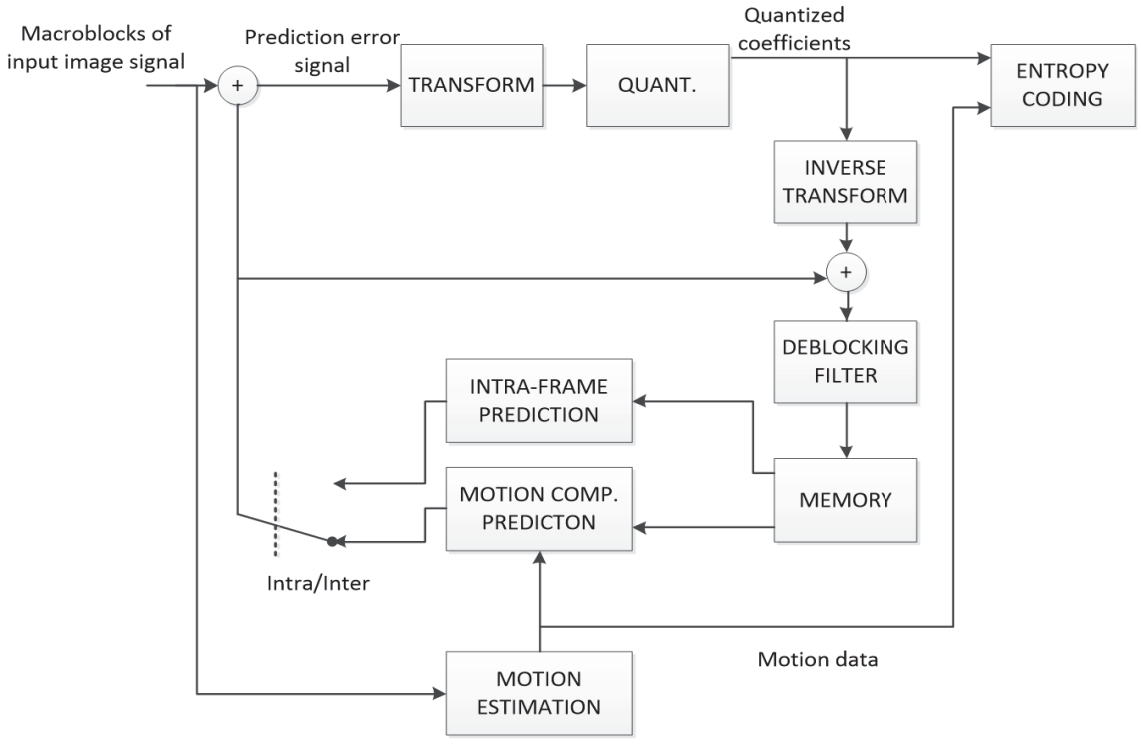


FIGURE 2.3: H.264 standard - Structure overview

the video sequence is reconstructed. This effect takes place because block edges are typically predicted and reconstructed with less accuracy, since adjacent MBs can content a drastically different information. Its main idea is to soften blocking artifacts if the absolute difference between samples near a block edge is large enough, considering different defined thresholds [131].

Regarding the entropy coding stage, H.264 proposes two sophisticated techniques to encode the prediction results (i.e., the prediction residual, the transform coefficients and the motion vectors): Context-Adaptive Variable Length (CAVLC), which maintains different VLC tables and switches among them to exploit inter-symbol redundancies and taking into account data statistics (i.e., previous elements which have been already coded); and Context-Adaptive Binary Arithmetic Coding (CABAC), which reaches higher compression ratios than CAVLC by using arithmetic coding that at the same time introduces more complexity for hardware implementations because of higher computational burden [133].

H.264 provides up to fourteen different profiles, which establish the set of options and features described in the standard that can be used. Globally, they can be summarised into four different categories: the Baseline profile, which supports all features included in the standard, except B/SP/SI slices, slice data partioning, CABAC and field coding; the Main profile includes, in addition to the features encompassed in the Baseline one, support for B slices, CABAC and field coding; the Extended profile, which supports all the aforementioned features, except CABAC; and the High profile, the last incorporated and the most efficient and powerful for high-definition and storage applications. The High profile comprises a set of new features, denoted as Fidelity Range Extensions (FRExt), that allow to extend the samples bit-depth (up to 12 bits) and the supported sampling formats, providing a profile for panchromatic video compression and even the possibility to define lossless compression. The standard also defines 15 different levels, which specify maximum frame size, video bit-rate and other features of the video encoding/decoding processes [131].

Its successor, the HEVC standard, commonly denoted as H.265 Recommendation [130], was developed not only to reduce the bit-rate about the 50% regarding H.264, but also to support efficient compression in high video resolution applications. The structure of HEVC is similar to H.264 but introducing some refinements in certain stages to improve compression performance. The MB is replaced by the Coding Tree Unit (CTU) as main coding unit, which can reach a maximum size of 64x64 luma samples, larger than the one permitted for MBs in H.264. CTUs can be also divided into smaller partitions along

the different stages which comprise the coding process [134]. Besides, the prediction can be performed in a larger range of block sizes for both inter- and intra-prediction stages. Regarding the transform stage, two-dimensional transforms are computed in the horizontal and vertical directions for 4x4, 8x8, 16x16 and 32x32 blocks of luma samples, deriving the elements of the transforms matrices by applying scaled DCT functions. In addition to the deblocking filter, a Sample Adaptive Offset (SAO) filter is included in the reconstruction loop, which modifies the value of the reconstructed samples by adding an offset, selected by the use of look-up tables. In this way, the samples reconstruction is optimized before writing them into the decoded frame buffer, which is then used by the motion compensation and estimation stages. Finally, HEVC only supports CABAC as entropy coder, but it has been enhanced to obtain high throughput [134].

An alternative to H.264 for video compression which has been widely analysed in the state-of-the-art is the use of the JPEG or JPEG2000 image standard algorithms but extended for video compression, version which is called Motion JPEG/JPEG2000 [135]. Using this technique, each frame is coded independently by intra-prediction, discarding the inter-prediction to exploit temporal redundancies in the video sequence. In this way, a simpler method in terms of implementation complexity is obtained in comparison with H.264 and H.265 specifications, since the inter-prediction stage is one of the most computationally intensive stages in video coding, at the expense of a reduction in the compression performance. The core of the Motion JPEG/JPEG2000 is comprised by a transform stage, a quantizer and a bit-plane encoder, inheriting the structure of the original image compression standards. The use of this algorithm is intended for applications with reduced local movement, such as videoconferencing, where the image is essentially static, reaching an acceptable compression ratio by exploiting only the spatial correlation among adjacent blocks of pixels in smooth areas.

The CCSDS has published some recommendations for on-board video interfaces, formats and transmission protocols, such as the CCSDS 766.1-B-2 for digital motion imagery [136], which also provides a set of video standards to choose from, depending on the target application. Regarding video compression, H.264 is recommended for real-time applications where live video is required on ground for monitoring purposes, and selecting one of the three available profiles depending on the required image quality. Its predecessor, MPEG-4, can be used for recording applications with a medium quality level. Motion JPEG2000 is also proposed for low bit-rate, high quality and high fidelity (frame-by-frame) applications.

2.5 Physical implementations for on-board data compression

As it was mentioned in Section 2.1, implementations of compression algorithms need to be evaluated to demonstrate their viability to work on-board satellites, taking into account the constraints of the embarked hardware in terms of available computational resources, power consumption or robustness against radiation. Although there is a trend in the use of commercial devices for certain short-time space missions, such as SmallSats for LEO missions [39], space-qualified technologies are preferred since they are radiation tolerant. Compression algorithms must meet some performance requirements when they are implemented on-board, such as throughput or real-time capabilities, which will depend on the specific space mission and the associated application.

On-board processing tasks could be executed as software, running onto the PFCU of the satellite, which is formed by a Central Processing Unit (CPU) that can be used for other purposes, such as control and monitorization. CPUs are not power efficient since they have a sequential behaviour, which at the same time could prevent to reach real-time processing capabilities. This latter feature is specially desirable for on-board compression techniques, since an on-the-fly compression of the collected data allows to reduce on-board storage demands. Because compression algorithms sometimes provide the possibility of apply parallelism at task level, other hardware devices have been adopted for their implementation on-board. In this sense, it is possible the use of customized ASICs or FPGAs, because of their strengths to work on-board satellites. Graphics Processing Unit (GPUs) are also a promising technology due to their inherent parallelization processing capabilities, as analysed in [51], implementations on these devices will not be taken into account since they are not currently qualified against radiation effects, though some different studies are underway in this area [137].

During the remainder of this Section, the technologies that are used on space missions to implement compression and other data processing are described in detail. Besides, the most relevant implementations existing in each category will be described.

2.5.1 Software-based

Implementations on CPUs are software-based and they are described in a high-level language, such as C/C++. The description of the desired functionality in a high-level language requires a short development time and a low cost, which is a clear advantage for space programs with restricted scheduling. Additional requirements are mandatory for software applications when they are developed as part of ESA space programs, as regulated by the ECSS-E-ST-40C *Software* guidelines [138]. CPU-based solutions present limitations in terms of power consumption and to offer real-time processing capabilities, because its intrinsic sequential nature. These restrictions bring to light specially when processing a high volume of data (e.g., hyperspectral images or video sequences). For these reasons, software versions of compression algorithms are not frequently implemented on-board satellites, except simple coding-based data compression techniques, which fit well on the hardware available on-board.

Some software implementations of data, image and video compression algorithms are currently available as open-source executable written in high-level languages, such as C/C++ or Python, or as part of third-party tool libraries. ESA provides the C source code of the CCSDS 121.0-B-2 and the 123.0-B-1 compression standards under request. In addition, these standards are offered by ESA, together with the CCSDS 122.0-B-1 image compression standard, under the WhiteDwarf GUI application, which provides the option of applying both the compression and the decompression flows [139]. The Universitat Autónoma of Barcelona (UAB) developed the Emporda software, an implementation of the CCSDS 123.0-B-1 lossless compression standard for multi- and hyperspectral images [140]. Recently, CNES is also providing a software version of the CCSDS 123.0-B-2 and 121.0-B-3 under non-commercial licence [141].

Regarding the physical platforms used to implement software processing approaches on-board satellites, the most extended microprocessor architecture in ESA programs is the LEON [142], a family of 32-bits processors that implement the Scalable Processor ARChitecture (SPARC) V8 Instruction Set Architecture (ISA). These cores are highly versatile, being them suitable for SoC designs, such as the Next Generation Multipurpose Processor (NGMP), a fault tolerant architecture that was implemented by Cobham Gaisler on the GR740 multiprocessor SoC, which is based on the LEON4-FT core [143]. Currently, LEON3 is the recommended architecture for space-qualified technologies, which is based on a deeper 7-stage pipeline and multi-processor support and it is the basis of different commercial products, such as the GR712RC, a dual-core LEON3-FT SPARC V8 processor

with advanced interface protocols and fault-tolerant capabilities [144]. The most recent developments, LEON4 and LEON5, are thought for their implementation in more powerful technologies, such as ASICs or FPGAs, and they include some improvements regarding their predecessors, such as wider internal buses, modified pipeline and support for a Level-2 cache, among other features.

The LEON family has been successfully validated for hyperspectral image compression when a multicore platform is used as target device. As an example, the work presented in [145] proposes a parallel implementation of the CCSDS 123.0-B-1 lossless compression standard on a quad-core LEON4-FT architecture, obtaining an overall speedup of $\times 3.71$ (using the 4 cores) when compared to its performance when running sequentially on a single core.

Nowadays, there is a trend in the space industry about the use of alternative microprocessor architectures, such as RISC-V, which provides an ISA under an open-source licence with support for three bit-widths, 32, 64 and 128 bits, and a variety of subsets for different applications, such as embedded systems or supercomputing. Its use for space applications has been addressed in [146]. Cobham Gaisler has recently developed the NOEL-V core, the first RISC-V synthesizable VHDL model for space missions, based on the 64-bits ISA subset. The well-known ARM microarchitecture, in which is based the CPUs of most of mobile devices, is being also evaluated to be integrated on-board satellites. For example, the High-Performance Spaceflight Computing (HPSC) [147] is a project funded by NASA that proposes a power-efficient and radiation-hardened ARM-based SoC design, based on ARM Cortex-A53 quad-core processors. This SoC also includes a high performance module for throughput-oriented applications and a real-time processing subsystem. Nanoxplore company also includes a hard-IP, based on the ARM-R5 processor, onto its NG-LARGE device [148]. Another alternative is the VORAGO VA10820, a low-power, radiation-hardened chip based on the ARM Cortex-M0 processor [149].

Alternatively, DSPs can be used to implement compression algorithms, since these devices are optimized to perform complex arithmetic operations, which are the core of image and video compression solutions. Some examples of space-qualified DSPs are the SMJ320C6701-SP provided by Texas Instruments [150], capable of working in floating-point precision and achieving a maximum performance of 1 GFLOPS by executing eight 32-bit instructions/cycle at 140 MHz, with a radiation hardness of 100 kRad Total Ionizing Dose (TID); and the RC64 device [151], a many-core solution thought for on-board image processing and based on an scalable architecture with up to 64 CEVA X1643 DSPs with floating-point extension,

which achieves a maximum performance of 40 GFLOPS working at 300 MHz, providing a radiation hardness of 300 kRad TID. Both hyperspectral image and video compression solutions implemented on DSPs are available in the state-of-the-art, though they generally target commercial DSPs. As an example, an H.264 encoder and a hyperspectral image compressor based on prediction, IWT and the Embedded Zero-tree Wavelet (EZW) are implemented on a TM320DM642 DSP in [152] and [153], respectively. In [154], another hyperspectral image solution based on a prediction stage that implements the spectral LOCO-I method combined with an arithmetic encoder is mapped on a TMS320C6416 DSP. Nonetheless, some space missions have included on-board 2D image compression implemented on DSPs. The TEAMSAT and Proba-1 missions supported by ESA are some examples, executing the JPEG algorithm based on the DCT on a DSP-TCS21020 [68], a radiation tolerant (100 kRad TID) 32-bits DSP that is able to achieve a throughput up to 45 MFLOPS.

2.5.2 Hardware-based

2.5.2.1 FPGAs

FPGAs are an interesting solution for the space industry, because of their reconfigurable capabilities together with a reduced development cost, in comparison with ASICs. Although antifuse-based RHBD FPGAs are clearly the most used FPGA technology on space missions nowadays, they are limited in terms of logic resources availability and maximum clock frequency, which is relevant to real-time data processing. In addition, antifused-based FPGAs are a one-time programmable technology, preventing its reconfigurability. For this reason, SRAM-based FPGAs are gaining interest for space missions with a high demand in terms of on-board data processing. SRAM-based FPGAs are some steps forward in terms of performance, power consumption and logic resources availability than other FPGA technologies, such as Flash- and antifuse-based, which have been exploited during lustrums to monetize the costs of device qualification for space missions. In addition, SRAM-based FPGAs can be reprogrammed more times than Flash-based FPGAs and even at a higher programmability speed. Nonetheless, these strengths are provided at the expense of a higher susceptibility against radiation effects, compared to antifuse- and Flash-based FPGAs.

There are several contributions to the field of FPGA implementations for on-board data and hyperspectral image compression, both on COTS and RHBD devices. Next, a summary of these implementations are provided, depending on the nature of the input data:

- **One-dimensional Data Compression**. The CCSDS 121.0-B-3 data compression standard, which is based on Rice coding, has been implemented on space-qualified FPGAs. The work presented in [155] implements the CCSDS 121.0-B-2 data compression standard on a XQR5VFX130 FPGA but enhancing the design with a 2D second-order prediction that overcomes the performance of the standard unit-delay predictor. This improved implementation reaches a throughput up to 205 MSamples/s, consuming around the 5% of slices and BRAMs available in the device.

  In [156], a low-complexity implementation of the FAPEC compressor is developed, targeting a Microsemi ProASIC3L M1A3P1000L FPGA, similar in terms of availability of logic resources to the space-qualified anti-fuse RTAX1000S FPGA from the same vendor. The implementation achieves a maximum throughput of 2 MSamples/s (16-bits input samples) working at 40 MHz, and consuming the 13% of available logic cells and the 9% of embedded memory, with a power consumption of approximately 20 mW.

  The compression system on-board the Chang'E-1, a Chinese mission for Moon Exploration, employs a DPCM scheme comprised by a differential predictive stage together with a bit-plane encoder. This solution is implemented on FPGA (device not specified) and it is able to work in both lossless and lossy modes, selecting the appropiate technique depending on the image content [68].

- **Hyperspectral Image Compression**. Implementations of the prediction-based CCSDS 123.0-B-1 lossless compression standard, which is based on the FL algorithm described in Section 2.3.1, abound in the literature, both in COTS and RHBD FPGAs. The fastest implementation of this algorithm available in the state-of-the-art using only one compression instance is the proposed in [157], where the authors implement a fine-grained pipeline in critical feedback loops of the BIP architecture, achieving up to 213 MSamples/s on a Xilinx Virtex-5 FX130T, the equivalent commercial device of the space-qualified Virtex-5 XQR5VFX130. In a more recent work [158], the authors also provide results on the next-generation Xilinx radiation tolerant Kintex UltraScale XQRKU060 FPGA, achieving a maximum throughput of 315 MSamples/s implementing the same architecture described in [157]. Other implementation target COTS devices, such as the one proposed in [159], which selects

a Xilinx Zynq-7020 MPSoC for the implementation, proposing a BIP architecture with different pipelining strategies to reduce the delay in the prediction module, and achieving a maximum throughput of 147 MSamples/s. Parallel implementations in which multiple instances of the compressor work simultaneously by using a shared memory, such as the ones proposed in [160] and [161], reach higher throughput at the expense of a considerable increment of the hardware occupancy. Preliminary implementations of the FLEX algorithm are also available, such as the one presented by Keymeulen et. al. [162], which reaches a throughput up to 61.51 MSamples/s in an AVIRIS flight compression experiment, when 15 compression instances are working simultaneously on a Xilinx Virtex-7 FPGA.

Current ESA missions, such as Proba-V or Sentinel-2 perform on-board compression on FPGA but focused on 2D transform-based algorithms. Proba-V mission includes an implementation of the CCSDS 122.0-B-1 image compression standard on a Microsemi RTAX2000S FPGA [163], which achieves a throughput of 90 Mbps by consuming the 48% of available cells and the 84% of internal RAM. The EnMap German mission also includes an implementation of the CCSDS 122 standard on an RTAX2000S device, though compression in this particular case is done in lossless mode following a band-by-band scheme [164]. A maximum throughput of 130 Mbps is achieved, using slightly higher hardware resources (54% of logic cells and 91% of available BRAMs) than the Proba-V implementation. Different studies have also evaluated the possibility of simplifying transform-based approaches, which are computationally cost, to perform on-board lossy compression on space applications and targeting FPGAs as implementation device. In [165], authors evaluate an optimized KLT to be implemented on hardware, with custom square root and division modules to reduce algorithm complexity. The design was evaluated on a Xilinx Spartan-6 FPGA, consuming 20% of the available logic cells and providing an output every 4 clock cycles. The complexity of the POT was also analysed in [166], demonstrating its feasibility to be implemented on a Microsemi RTAX2000S FPGA for different image sizes, reaching a throughput between 12.5 and 18.4 MSamples/s, depending on the selected configuration.

Implementations of near-lossless and lossy compression algorithms on FPGA are also available in the state-of-the-art, evaluating the architectural complexity and compression performance of these solutions to target space applications. The transform stage of the HyperLCA algorithm, which originally works with floating-point precision, is evaluated in [66] using integer precision, demonstrating its viability to

be implemented on hardware (concretely, a Xilinx radiation tolerant XQRKU060 FPGA), without incurring in a compression penalty compared to the floating-point version. In [167], JYPEC is presented, which is a lossy hyperspectral compression algorithm that combines PCA and JPEG2000. JYPEC has been evaluated on a Xilinx XC7VX690T FPGA, consuming the 5% of the available slices and a minimum memory footprint (0.4% of the BRAMs) and achieving the goal of real-time performance for the AVIRIS-NG sensor constraints (74 MSamples/s, considering 8-bits samples). Near-lossless alternatives, such as the implementation of the Lossy Compression for Exomars (LCE) algorithm presented in [168] or the one of the Low-Complexity Predictive Lossy Compression (LCPLC) solution developed in [169], also demonstrate their feasibility to be implemented on-board satellites onto a Xilinx Virtex-5 FPGA, combining a reduced logic resources consumption together with promising throughput results, even obtaining in the latter implementation real-time capabilities (up to 120 MSamples/s).

- **Video Compression**. Since video sensors are not extended on-board satellites yet, implementations of video encoders on FPGAs are limited to preliminary analysis to demonstrate their viability on future space missions, taking into account the intrinsic complexity of these solutions. It is also remarkable that existing implementations target commercial FPGAs, which have a high logic resources availability, bringing to light the high complexity of developing video encoding hardware solutions. H.264 has been completely evaluated (i.e., integrating all the modules that comprise the VCL) in [170], achieving a viable implementation of the Baseline Profile with a tailored VLC encoder on a Xilinx Virtex-6 FPGA, which satisfies the constraints of HDTV (1280x720 pixels at 60 fps), consuming the 89% of the available slices and the 22% of the embedded memory. An alternative implementation of the H.264 encoder is presented in [171], which is mapped on an Altera Arria II GX FPGA, consuming the 66% of the available LUTs and supporting 1080p at 60 fps. The complexity of the H.264 entropy encoders is analysed in [172] onto a Xilinx XC5VLX110T-3-FF1136 FPGA, providing also a hardware solution that allows to select between the two possible encoders defined by the standard. The whole coding solution consumes the 39% of the available slices and also internal DSPs and BRAMs in the case of CABAC, and it is able to work at 163 Mbin/s with a working frequency of 164 MHz. Another Double-Mode Binary Coder (i.e., CAVLC plus CABAC) is presented in [173], implementing the developed solution on an Altera Stratix II FPGA (3917 LUTs are consumed) and encoding with a rate up to 159 Mbin/s. Regarding HEVC,

studies have been focused on how to simplify the complexity of the intra-prediction stage (up to 33 directional modes, in comparison with the 9 modes available in H.264) to fit well in the logic resources available on an FPGA, without compromising the throughput and the compression performance. As an example, in [174] the authors propose an scalable architecture that checks a variable number of candidate modes, which are preselected by the processing of 8x8 predictions computed from original samples, and supporting also 4x4 modes thanks to the incorporation of a separate reconstruction loop. The proposed intra-encoder (i.e., inter-prediction is not implemented) consumes the 79% of LUTs and the 83% of DSPs available on an Altera Arria II GX FPGA, providing real-time capabilities for resolutions up to 1080 pixels at 60 fps. The work presented in [175] optimizes the intra-prediction by a three-stage architecture that reduces dependencies on the reference generation, needed for the processing of the next Coding Unit (CU). The feedback to know the CU split scheme and the prediction mode selection is avoided by using a Hadamard-based decision method. In this way, the solution achieves a minimum operating frequency of 140 MHz, encoding Full HDTV in real-time (1920x1080 pixels at 30 fps). This design was implemented on a Xilinx Zynq XC7Z045 MPSoC, consuming the 38% of LUTs and the 4% of DSPs available in this device.

2.5.2.2 ASICs

Implementations on ASICs were a common practice in the space industry until the irruption of space-qualified reprogrammable FPGAs. These ad-hoc solutions provide low-power and high throughput performance, at the expense of high development time and Non-Recurring Engineering (NRE) costs, since ASICs cannot modify their behaviour once they have been manufactured. This prevents their adaptation to new mission requirements or the possibility of reconfiguring a part of the design in case of malfunction due to radiation effects. Most of the space missions that have included on-board data and/or image compression in the last decades employ ASICs as target devices, as it is reflected in [68]. Some examples are the Mars Odysssey, launched by NASA in 2001, which implements on ASIC not only a Rice encoder for data compression, but also a FL-based solution for lossless image compression that can be substituted for a DCT to compress in lossy mode; ALOS, a satellite launched by JAXA in 2006, which performs on an ASIC-JAXA IDCP both the JPEG baseline and the JPEG-LS described in Section 2.2, depending on the desired compression mode; or the PLEIADES-HR mission, funded by CNES and

launched in 2010, which implements a compression solution comprised by a DWT stage and a Bit-Plane encoder on Wavelet Image Compression Module (WICOM), an ASIC solution developed by Airbus Defence and Space.

ESA has also funded the development of some ASIC implementations specific for on-board data compression. The CWICOM chip [176], shown in Figure 2.4, is an image compression ASIC developed by Astrium that implements the CCSDS 122.0-B-1 Wavelet-based image compression standard. CWICOM, which is on-board the Gaia Astrometry Mission [177], provides a high compression performance (up to 60 Mpixels/s) by exploding spatial correlation, supporting both lossless and lossy compression. This solution takes advantage of a high-capacity and very efficient internal embedded memory organization to store the DWT coefficients without the necessity of external storage. In addition, the power consumption is reduced (<100 mW/MSamples/s) and the radiation hardness is 100 kRad TID, being also tolerant to SEU by using an EDAC mechanism. Other alternative is the Payload Rice Data Compressor (PRDC) [178], an ASIC data compressor that implements an extension of the Rice coding (including an optional pre-processing stage), which is outside the CCSDS 121.0-B-2 lossless data compression standard, though it is possible to disable certain features in order to comply with the standard. This ASIC is able to handle samples with a size from 4 to 24 bits, achieving a maximum throughput of 40 MSamples/s. The power dissipation of the PRDC is 630 mW at 20 MHz unloaded, while it provides a TID $>$ 30 kRad.

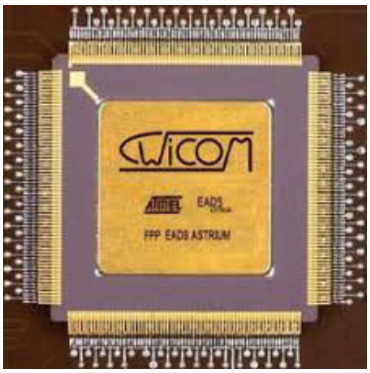


FIGURE 2.4: CWICOM ASIC for image compression (extracted from [176])

2.6 Conclusions

As presented throughout this Chapter, the specialized literature is plenty of different alternatives for data compression, which can be classified depending the nature of the

handled data, the level of losses introduced in the compression process or the performance features that are prioritized, such as low-complexity or high computational capabilities. Moreover, there are compression approaches specifically thought to work on-board satellites, taking into account the constraints of a harsh environment such as the outer space. However, it has been observed that there is not a common way to develop and implement universal compression solutions for space missions, which makes necessary ad-hoc decompression solutions on ground for each specific compression approach. For this reason, this Thesis focuses on compression algorithms that provide a good compromise between complexity and compression efficiency, therefore focusing on prediction-based decorrelators and entropy coders based con Golomb-Rice codes. More specifically, those compression algorithms proposed by the CCSDS standards for one-dimensional data and 3D images (i.e., multi- and hyperspectral) are considered. This is mainly motivated by the fact that, in addition to the trade-off they provide between compression performance and hardware complexity, decompression can be performed on ground by using a standard decompressor, allowing compatibility and reusability among different applications.

Besides, a gap has been detected in the state-of-the-art regarding versatility of on-board compression solutions. This means that there is not a collection of modules that can be reused for different purposes or for different space missions with different performance goals, needing for a complete development each time a compression solutions is required. This links with one of the main objectives of this Thesis, which is to provide modular compression solutions, based on the CCSDS standards, that allow to conform an optimized compression chain for a specific application taking into account different constraints, such as throughput, hardware occupancy or RD ratio, by selecting the appropriate modules among the available alternatives.

Regarding the physical platform to implement those compression solutions, space-grade FPGAs are targeted since they are considered an optimal candidate for current and future space missions thanks to their high performance, resources availability, reduced power consumption and acceptable development costs. More concretely, SRAM-based FPGAs are emerging as an interesting alternative, thanks to their higher performance, lower cost and the possibility to update the functionality during the space mission lifetime, in comparison with antifuse or flash-based technologies.

Chapter 3

Design and characterization of prediction-based preprocessing blocks

This Chapter presents different prediction-based approaches developed as functional blocks, which can be efficiently implemented on space-qualified FPGAs as spatial and/or spectral decorrelators, depending on the nature of the data acquired by the embarked sensor. The alternatives under study are mainly based on the CCSDS 123 compression standard for multi- and hyperspectral images, considering both Issue 1, focusing on lossless compression, and the recent Issue 2, that extends the functionality of its predecessor to support near-lossless compression. A simple predictor proposed in the CCSDS 121.0-B-3 lossless compression standard is also considered for generic data acquired on the satellite. Both algorithmic descriptions and architectural details of the proposed implementations are provided, in addition to preliminary synthesis results on a representative space-grade FPGA technology.

3.1 Outline

Prediction-based preprocessors are commonly the first stage in a compression chain. This is because their simplicity to be implemented on hardware and their efficiency to decorrelate redundant information present on the collected data. This correlation can be exploited in hyperspectral images in both the spatial and the spectral domains. When these preprocessors are implemented as part of a compression solution on-board satellites, they should reach not only an efficient data reduction, but they should also take into account some additional constraints, such as low-complexity and high throughput, as some applications, mainly in the field of Earth Observation, require support for real-time processing. Besides, robustness against radiation effects is desired, in order to prevent a malfunction in case of a bit flip.

After the study of the on-board data compression algorithms available in the specialized literature, it is considered that the prediction-based preprocessors proposed by the CCSDS standards provide a trade-off for efficient data decorrelation as part of on-board compression, offering acceptable Rate-Distortion ratios with a reduced computational complexity. Among these standards, the CCSDS 123.0-B-2 [31] describes a prediction-based near-lossless compressor of multi- and hyperspectral images, while its predecessor, the CCSDS 123.0-B-1 [30] was conceived for lossless compression only. In addition, other compression standards, such as the CCSDS 121.0-B-3 [27], constitutes a universal prediction-based compressor applicable to any kind of data.

This Chapter presents three different prediction-based stages and their implementation on space-grade FPGAs. The user can select the appropriate alternative depending on the data nature (generic information or hyperspectral images) and the desired compression ratio, distinguishing between lossless and near-lossless compression. These preprocessors have been developed as functional blocks, in order to be integrated as part of a whole compression chain. RTL and HLS design methodologies are used, keeping in mind the requirements of each implementation in terms of throughput, hardware occupancy and available development time to select the suitable workflow.

Two of these preprocessing approaches, the CCSDS 121.0-B-3 unit-delay predictor and the CCSDS 123.0-B-1 3D predictor, are currently part of SHyLoC, a pair of technology-independent IP cores described in VHDL, which implement the compression algorithms described in those respective lossless standards [59, 60]. These prediction-based preprocessors have been developed trying to maximize the throughput, ensuring at the same

time a low area footprint. For this reason, they have been developed following the RTL design methodology, focusing on generating a cycle-accurate description that allows to obtain as much throughput as possible, without compromising in excess the logic resources utilization.

The remaining approach implements the prediction-based preprocessor of the novel CCSDS 123.0-B-2 near-lossless compression standard, developed following the HLS methodology. In this way, a functional model of the algorithm behaviour is provided to perform an early design space exploration. This approach allows to detect optimal parameter values and to identify algorithm features that enable to maximize compression performance faster than following an RTL-based methodology, since it is done at a higher level of abstraction. Moreover, the prototyping is accelerated, demonstrating the viability of the proposed compression solution to be implemented in a space-grade device. Therefore, the main objective of this implementation is to demonstrate the feasibility of the CCSDS 123.0-B-2 near-lossless compression algorithm on FPGA, which is prioritized against other performance metrics, such as the throughput.

Preliminary synthesis results are provided for each one of the proposed prediction-based pre-processing stages in a representative space-grade FPGA with different sets of generic parameters. Concretely, results are presented for Xilinx Kintex UltraScale XCKU040, since it is supported by both RTL and HLS workflows.

3.2 CCSDS 121.0-B-3 unit-delay predictor

3.2.1 Algorithm overview

The prediction-based preprocessor defined in the CCSDS 121.0-B-3 standard [27] removes correlation between consecutive input samples and maps them into unsigned values, as shown in Figure 3.1. These mapped samples are the input of the block-adaptive entropy coder, the responsible of performing the encoding stage. This preprocessing stage is based on a simple and reversible unit-delay predictor, which employs the previous sample $s(t-1)$ as an estimator of $s(t)$. The introduction of this prediction-based preprocessor requires the periodic insertion of reference samples (i.e., an unmodified input sample) in the output compressed stream to be able to obtain the original image on the decompression side. The reference interval r is a user-defined parameter limited to 4096, which specifies with which

frequency the reference samples are inserted in the output bitstream. The lower the r value, the higher is the robustness of the preprocessor against radiation effects, since more reference samples are inserted. This implies that, in case of a bit flip, only the corrupted data segment between two reference samples is lost.

The predicted sample $\hat{s}(t)$ is equal to the previous sample $s(t-1)$, except for the first sample in a reference interval, which is $\hat{s}(t) = s(t)$ in this specific case (i.e., $t = 0$).

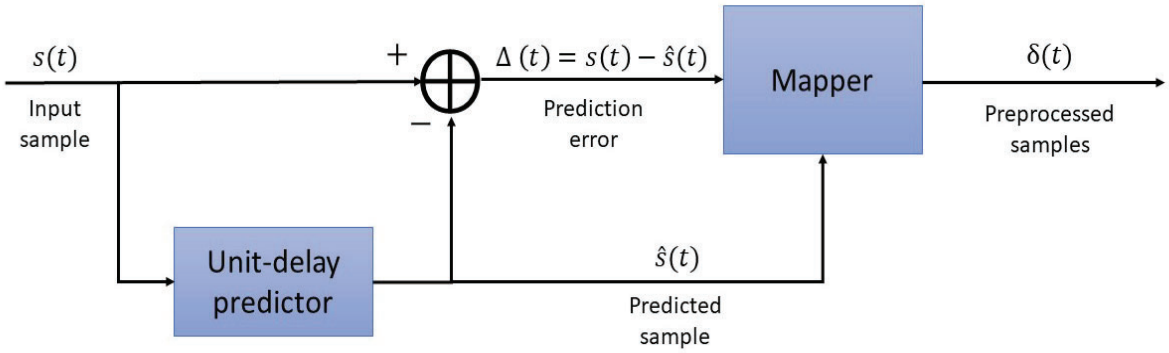


FIGURE 3.1: Unit-delay predictor overview

The mapper receives the prediction residual $\Delta(t)$, which is the difference between the predicted and the input samples, generating a non-negative integer $\delta(t)$, known as mapped residual. For every $\hat{s}(t)$, there are 2^n possible $\Delta(t)$ values, being n the input sample bit-depth. The mapper follows Equation 3.1:

$$\delta(t) = \begin{cases} 2 \cdot \Delta(t), 0 \leq \Delta(t) \leq \theta(t) \\ 2|\Delta(t)| - 1, -\theta(t) \leq \Delta(t) < 0, \\ \theta(t) + |\Delta(t)|, otherwise \end{cases} \tag{3.1}$$

where $\theta(t)$ is

$$\theta(t) = min(\hat{s}(t) - s_{min}, s_{max} - \hat{s}(t)), \tag{3.2}$$

being $s_{min} = -2^{n-1}$ and $s_{min} = 0$ for signed or unsigned input samples, respectively. In the same way, $s_{max} = 2^{n-1} - 1$ and $s_{max} = 2^n - 1$ when processing signed or unsigned samples, respectively.

3.2.2 Block design

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 $\Delta(t)$ and the mapper. At this point, mapped prediction residuals $\delta(t)$ are obtained, which are then sent to an entropy coder. The top module of the unit-delay predictor core includes, in addition to the unit-delay predictor and the mapper, the necessary logic to bind the components that perform the reception of input samples and the flow control of the output interface to send the mapped prediction residuals $\Delta(t)$, which is mainly managed by a couple of FIFOs and a Finite-State Machine (FSM). The block diagram of this preprocessing stage is shown in Figure 3.2.

The unit-delay predictor has been conceived in a way that it can be easily connected to other functional blocks to conform more complex systems for a fully compression process. Input and output data interfaces are based on a handshaking protocol, which associates a *Valid* flag to each input/output data. The *Ready_Ext* input signal informs that the data receiver is expecting new output samples. Additionally, there are other control signals, such as *ForceStop*, which allows to suspend at any point the prediction under the user demand.

Runtime configuration is directly received through an AHB slave interface and validated with a *Valid* flag. According to the received configuration, a FSM controls the operation of the rest of the units in the design. The *fsm* module may bypass the pre-processing stage (both the unit-delay predictor and the mapper) by activating the *Bypass* signal, in order to periodically insert the reference samples. The *components* module performs the pre-processing itself, containing both the unit-delay predictor which computes the predicted samples based on the input data, and the mapper module. Moreover, the 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 predicted data in groups of J samples, until they are required by the block-adaptive encoder.

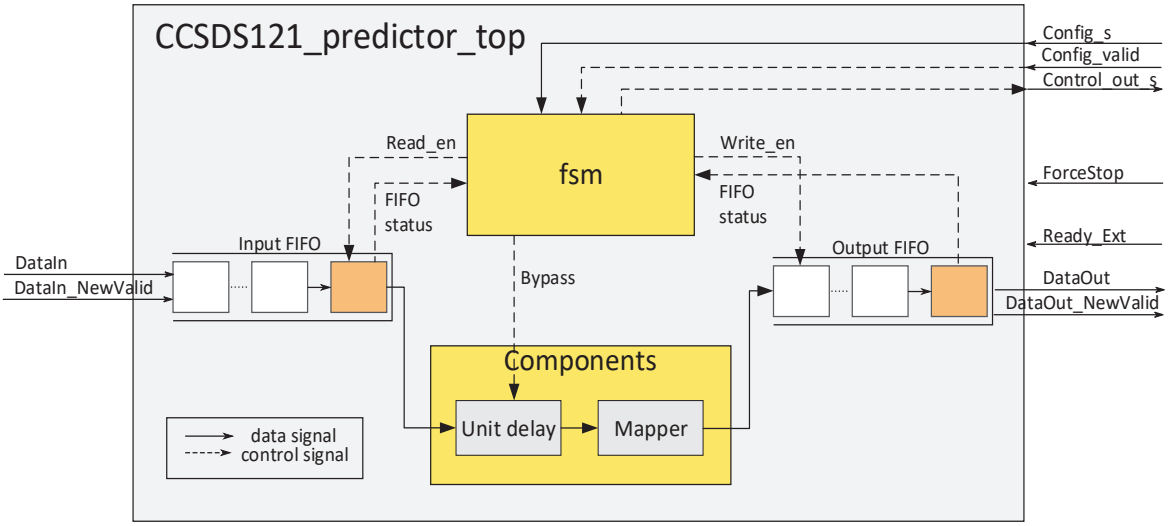


FIGURE 3.2: Block diagram of the CCSDS 121.0-B-3 predictor top module

3.2.3 Block characterization

Preliminary synthesis results for the CCSDS 121.0-B-3 unit-delay predictor have been obtained using Synopsys Synplify Premier P-2019.09-SP1.The parameter sets used for synthesis purposes are summarised in Table 3.1. The main parameters that change among configurations are the input sample bit-depth (controlled by the *D_GEN* generic), using 8, 16 and 32 to analyse the impact of the bit depth extension in the internal operations; or the block size *J*, which defines the number of samples that comprise an input block. As it is also reflected in Table 3.1, the maximum image size is fixed to 1024 for each one of the three coordinates just for synthesis purposes. In addition, other image parameters are considered, such as the sign and the endianness of the input samples.

TABLE 3.1: Sets of synthesis configurations for the CCSDS 121.0-B-3 unit-delay predictor

Generic	Set1	Set2	Set3	Set4
EN_RUNCFG	1	0	1	0
RESET_TYPE	1	1	1	1
Nx_GEN	1024	1024	1024	1024
Ny_GEN	1024	1024	1024	1024
Nz_GEN	1024	1024	1024	1024
D_GEN	32	16	16	8
ENDIANESS_GEN	0	1	0	1
IS_SIGNED_GEN	0	0	1	0
J_GEN	64	8	32	8
REF_SAMPLE_GEN	4096	512	256	256

The timing results for the CCSDS 121.0-B-3 unit-delay predictor in terms of maximum clock frequency are reflected in Table 3.2 for Xilinx Kintex UltraScale XCKU040. In addition, that table contains resources utilization in terms of memory blocks, arithmetical units and logic resources. The maximum throughput for each configuration, measured in terms of MSamples/s, is approximately equivalent to the maximum clock frequency achieved, since this predictor is able to process one sample per clock cycle.

Results show that the best outcome in terms of maximum clock frequency is 205.8 MHz when using the most reduced configuration (i.e., the Set4), being $D = J = 8$. The extension of the dynamic range implies an impact in timing performance, obtaining worst results when $D = 32$ and reducing an average of 17% when changing from $D = 16$ to $D = 32$ (i.e., from Set3 to Set1).

In terms of resources utilization, the maximum logic usage for Kintex UltraScale is approximately the 0.3% of the available LUTs without using embedded memory blocks. These results are obtained when the predictor is configured to manage input samples with $D = 32$ (i.e., Set1), the most critical configuration in terms of hardware occupancy. It is also appreciated that BRAMs are not used, since the synthesis tool maps the different FIFOs present in the design as distributed memory by using logic resources. This is because the small size of those FIFOs, which are under the threshold that the synthesis tool defines to map memory elements as dedicated or distributed memory. In any case, this threshold can me modified by the user in order to use BRAMs and thus increasing the maximum clock frequency of the predictor.

As it can be observed analysing the differences between results for Set2 and Set3, the block size generic *J_GEN* also has an impact in the resources utilization, though it is reduced compared to the penalty introduced by increasing *D_GEN*.

Table 3.2: CCSDS 121.0-B-3 unit-delay predictor - Synthesis on Xilinx Kintex UltraScale XCKU040

Parameters	**Total**	**Set1**	**Set2**	**Set3**	**Set4**
Block RAMs	600	0	0	0	0
DSP48	1920	3	3	3	3
Registers	484800	199	131	140	113
LUTs	242400	618	281	341	193
Maximum Frequency (Clk_S) (MHz)		129.7	181.0	167.8	205.8

From the obtained results it can be concluded that the developed CCSDS 121.0-B-3 unit-delay predictor constitutes a low-complexity solution for generic data reduction, providing a high throughput (even achieving real-time processing on-board satellites) together with a reduced hardware occupancy on space-grade FPGAs.

3.3 CCSDS 123.0-B-1 spectral and spatial decorrelator for HSI lossless compression

3.3.1 Algorithm overview

The top-level hierarchy of the CCSDS 123.0-B-1 prediction stage is reflected in Figure 3.3, including all the necessary functional units to generate a predicted mapped residual $\delta(t)$ from an input sample $s_{z,y,x}$, which will be explained throughout this Section.

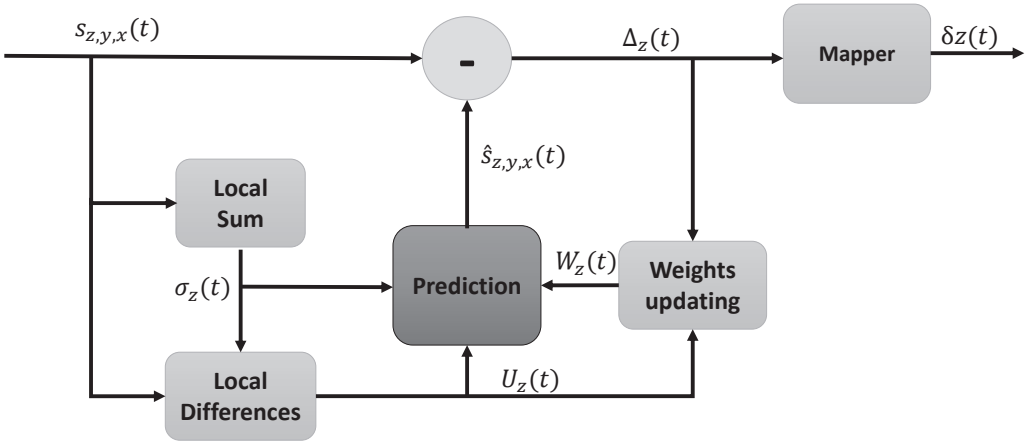


FIGURE 3.3: CCSDS 123.0-B-1 predictor overview

The preprocessor of the CCSDS 123.0-B-1 standard [30] computes the value of the current input sample using a set of pixels in its vicinity, counting on samples in the same band z as well as in previously processed bands, as it is shown in Figure 3.4. The predictor processes the input image in a single pass, independently of the order in which the input samples are arranged.

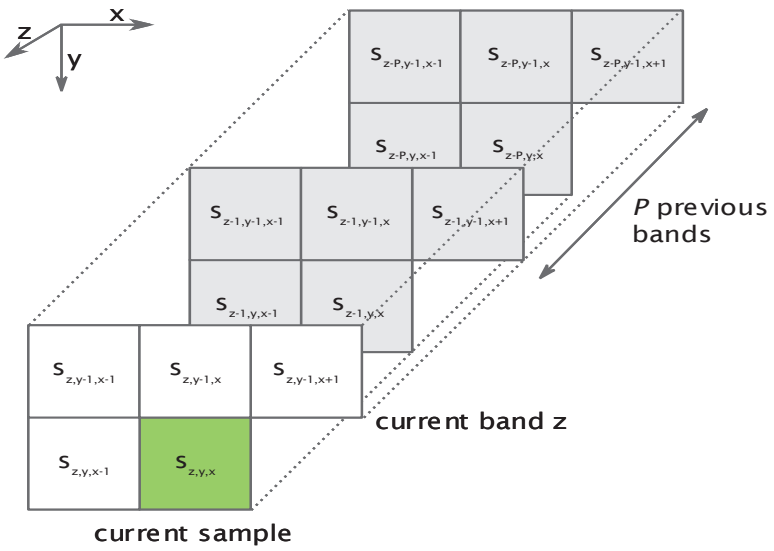


FIGURE 3.4: Set of samples used for prediction

For each input sample $s_{z,y,x}$, first a *local sum* $\sigma_{z,y,x}$ is computed, which is a weighted sum of neighbouring samples in the current band z (i.e., in the spatial domain). 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 3.3 describes the way to compute the local sums under the wide neighbour-oriented mode, while Equation 3.11 indicates the samples that are used when the column-oriented local sums are selected and their associated weighted value. Input sample values $s_{z,y,x}$ are directly employed to compute these local sums. Local sums are also computed for each one of the P previous bands used for prediction. The P value can be configured between 0 and 15, though it has been observed that no significant improvements are achieved when setting values of P higher than 3 [179]. The value of the local sum at the beginning of each band $\sigma_{z,0,0}$ is not considered, since it is not used during the computations.

$$\sigma_{z,y,x} = \begin{cases} s_{z,y,x-1} + s_{z,y-1,x-1} + s_{z,y-1,x} + s_{z,y-1,x+1}, & y > 0, 0 < x < N_x - 1 \\ 4s_{z,y,x-1}, & y = 0, x > 0 \\ 2(s_{z,y-1,x} + s_{z,y-1,x+1}), & y > 0, x = 0 \\ s_{z,y,x-1} + s_{z,y-1,x-1} + 2s_{z,y-1,x}, & y > 0, x = N_x - 1 \end{cases} \tag{3.3}$$

$$\sigma_{z,y,x} = \begin{cases} 4s_{z,y-1,x}, & y > 0 \\ 4s_{z,y,x-1}, & y = 0, x > 0 \end{cases} \tag{3.4}$$

Then, the *local differences* are computed, by subtracting the neighbour sample values from the previously computed 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 Equations 3.5, 3.6 and 3.7, using for each case the value of the adjacent sample on top, left and top-left of the current one, respectively. These differences are then grouped into 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. T On the other hand, the directional differences of the current band are also included with the full mode.

$$d^N_{z,y,x} = \begin{cases} 4s_{z,y-1,x} - \sigma_{z,y,x}, & y > 0 \\ 0, & x > 0, y = 0 \end{cases} \tag{3.5}$$

$$d^W_{z,y,x} = \begin{cases} 4s_{z,y,x-1} - \sigma_{z,y,x}, & x > 0, y > 0 \\ 4s_{z,y-1,x} - \sigma_{z,y,x}, & x = 0, y > 0 \\ 0, & x > 0, y = 0 \end{cases} \tag{3.6}$$

$$d^{NW}_{z,y,x} = \begin{cases} 4s_{z,y-1,x-1} - \sigma_{z,y,x}, & x > 0, y > 0 \\ 4s_{z,y-1,x} - \sigma_{z,y,x}, & x = 0, y > 0 \\ 0, & x > 0, y = 0 \end{cases} \tag{3.7}$$

The optimal combination of the local sum and prediction modes is highly dependent of the target HSI sensor and the image nature, and it should reach a compromise between compression performance and hardware occupancy.

The next step is to compute a weighted sum of the elements in the local differences vector, making use of an internal *weight vector*, $W_{z,y,x}$. A weight vector is separately maintained for each band, and the resolution of each weight value is defined by Ω, a user-defined parameter in the range $4 \leq \Omega \leq 19$. The higher is the Ω value, the higher is the accuracy during the prediction calculation. 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 bands. With the custom weight initialization, the initial weights are provided by the user, and each band may have different values. This latter mode is specially relevant when it is known beforehand that certain initial weight values improve the compression performance for an specific instrument.

The inner product of the $U_{z,y,x}$ and $W_{z,y,x}$ components, denoted as *predicted central local difference* $\hat{d}_{z,y,x}$, is used to calculate the *predicted sample* as

$$\hat{s}_{z,y,x} = \left\lfloor \frac{\tilde{s}_z(t)}{2} \right\rfloor, \tag{3.8}$$

which is the estimated value for an input sample, taking into account image statistics (i.e., the value of preprocessed samples in both the spatial and the spectral vicinity). The scaled predicted sample $\tilde{s}_z(t)$ is estimated as

$$\hat{s}_{z,y,x} = \begin{cases} clip(mod_R^* \left\lfloor \frac{\hat{d}_z(t)+2^{\Omega}\cdot(\sigma_z(t)-4s_{mid})}{2^{\Omega+1}} \right\rfloor + 2s_{mid}+1, \{2s_{min}, 2s_{max}+1\}), & t>0 \\ 2\cdot s_{z-1}(t), & t=0, P>0, z>0 \\ 2\cdot s_{mid}, & t=0 \text{ and } (P=0 \text{ or } z=0), \end{cases} \tag{3.9}$$

being $t = y \cdot Nx + x$.

Finally, weight values are updated with each new sample based on the prediction residual $\Delta_{z,y,x}$ (i.e., the difference between the input and the predicted sample), the local differences and some user-defined parameters. The prediction residual is mapped into an unsigned integer $\delta_{z,y,x}$, which is passed to the entropy coder stage.

3.3.2 Block design

The predictor can be configured with a wide set of configuration parameters which can be selected at compile-time or at runtime. In case of runtime configuration, it is received through an AMBA AHB slave interface. Three different architectures are devised for the predictor of the CCSDS-123, one for each of the possible input sample arrangements (BIP, BSQ and BIL), in order to find the best compromise between complexity and throughput [59, 60]. Each architecture includes the prediction core itself and a control module, which

includes a FSM to schedule the different operations to be performed. Independently of the selected predictor architecture, input and output data interfaces are shared to ease connectivity with other functional blocks. Similarly to the CCSDS 121.0-B-3 unit-delay predictor, a handshaking protocol with associated *Valid* and *Ready* signals is defined, though the latter is just present in the output interface. Besides, some additional control signals are defined to inform about the state of the prediction (i.e., finishing, processing the last sample) or to suspend it.

Specific schedules are proposed for each input order, avoiding data dependencies at the same time that memory occupancy is kept in mind [59]. The BSQ and BIL schedule is shown in Figure 3.5, where it is observed that the weight vector must be updated before the prediction of a new sample, which shall be computed by the Multiply and Accumulate (MAC) stage. This implies a limitation in the number of operations that can be scheduled in parallel. The schedule of the BIP order is illustrated in Figure 3.6, where the current sample $s(t)$ is processed in all the bands before the contiguous sample in the spatial domain $s(t+1)$ is processed. Therefore, the existing data dependencies do not prevent to start the compression of a new sample before the prediction of the current one has finished, because the corresponding weight vector will be already updated and available. This allows to start the compression of a new sample in every clock cycle, by applying a pipeline strategy.

A block diagram overview of the predictor internal structure is shown in Figure 3.7, where the common functional units to all the predictor architectures are included. The input samples to be processed are first arranged in a set of FIFOs, as determined by the compression order. As samples are compressed, they are rearranged in a way that the

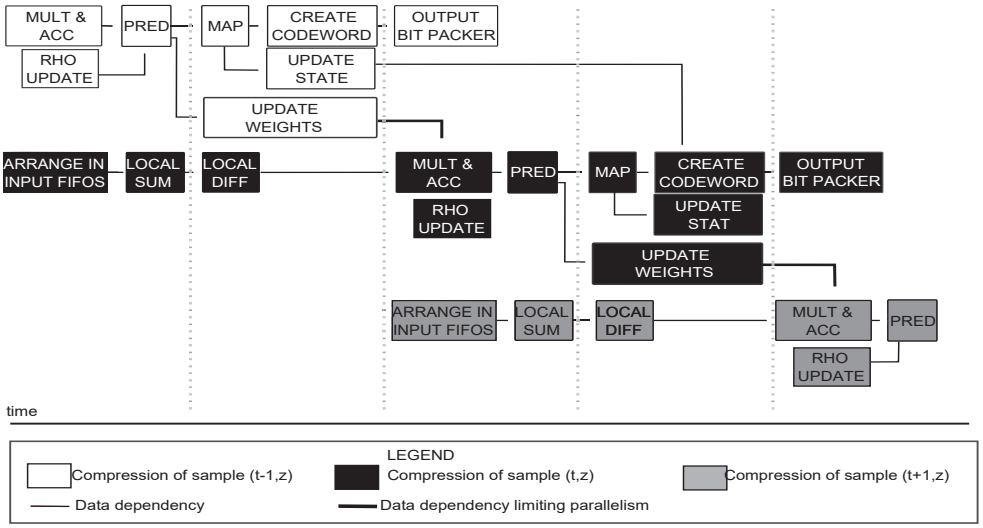


FIGURE 3.5: CCSDS 123.0-B-1 algorithm - BSQ and BIL schedule

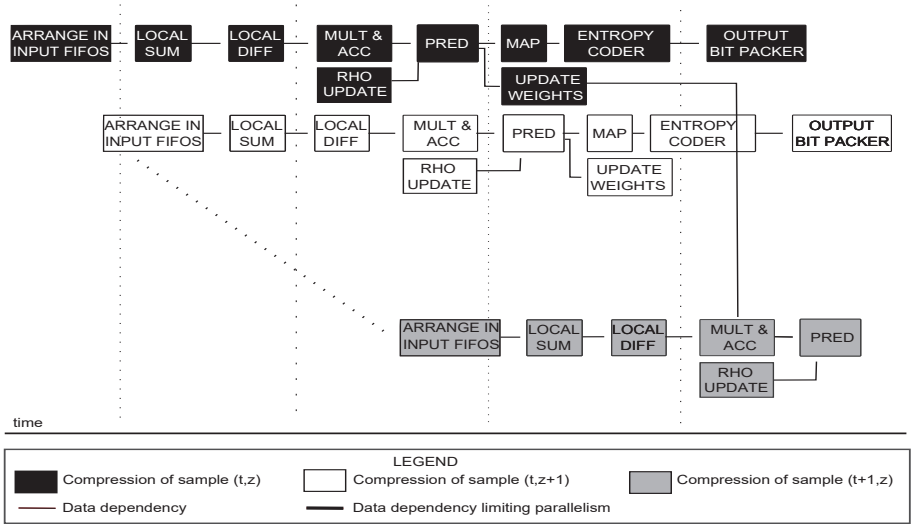


FIGURE 3.6: CCSDS 123.0-B-1 algorithm - BIP schedule

already processed samples become the neighboring samples for subsequent samples. Large memory elements that can be stored externally depending on both the selected processing order and the logic resources availability of the target device are depicted in blue in Figure 3.7.

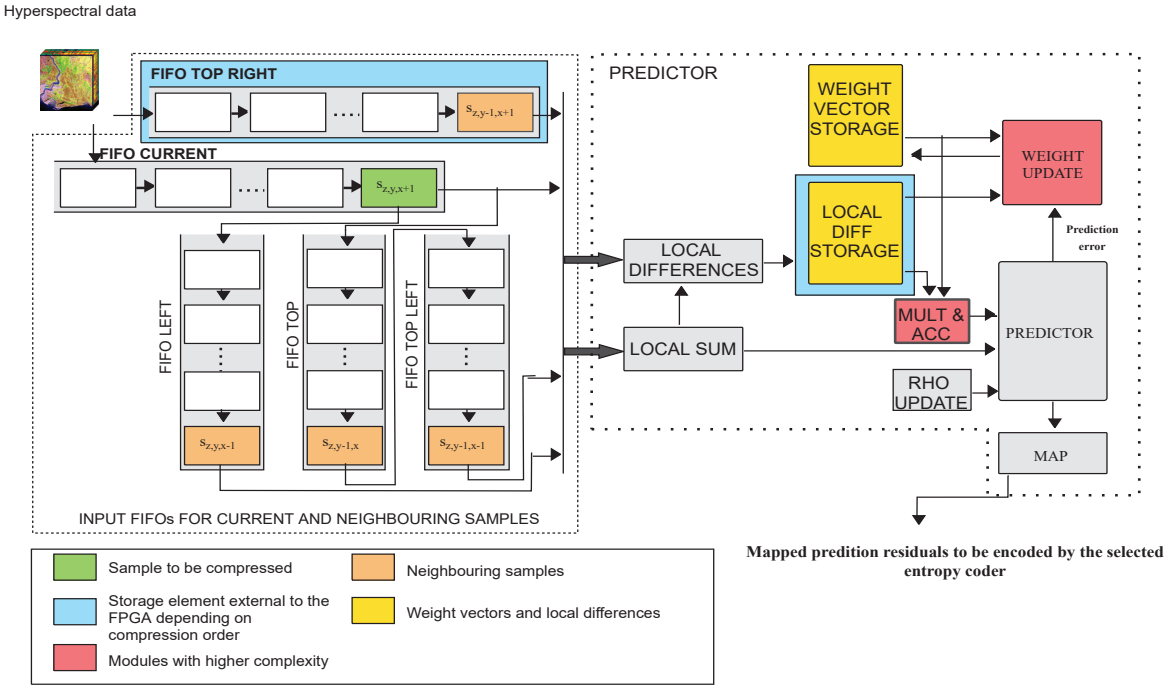


FIGURE 3.7: CCSDS 123.0-B-1 predictor internal structure

The modules with the highest mathematical complexity are shown in red, which are the MAC unit and the weight update stage, since they are the stages that require the execution of multiplications.

Regarding memory utilization, each input sample arrangement requires different storage for certain internal elements, such as the FIFOs used to store the neighbouring samples, the local differences and the weight vectors. The number of elements to be stored for each possible input order is reflected in Table 3.3, being $C_z = P\_MAX + 3$ under full prediction mode and $C_z = P\_MAX$ when reduced prediction mode is selected. Because of the amount of internal storage required by the BIP and BIL architectures, a version of them called BIP-MEM and BIL-MEM, respectively, are developed, which use an external memory to store intermediate results. An AMBA AHB master interface is included to access this memory. The storage elements which require the highest amount of memory are the *FIFO_TOP_RIGHT*, which stores the neighbouring samples at that direction, and the *Local differences storage*, in BI (i.e., both BIP and BIL) and BSQ orders, respectively.

Table 3.3: CCSDS 123.0-B-1 algorithm - Number of elements to be stored for each possible input arrangement

Order	Neighbouring Samples	Local Differences	Weights
BSQ	$N_x + 1$	$N_x \cdot N_y \cdot P$	C_z
BIP	$(N_x + 1) \cdot N_z$	P	$N_z \cdot C_z$
BIL	$(N_x + 1) \cdot N_z$	$N_x \cdot P$	$N_z \cdot C_z$

Specific optimizations for each predictor architecture are described in the next lines, depending on the selected processing order:

3.3.2.1 BIP architecture

The BIP architecture has been developed in a way that is capable of accepting and processing an input sample per clock cycle. For this reason, this architecture is the one that reaches the highest possible throughput, because it is able to avoid data dependencies in the processing of consecutive input samples [59].

The local differences vector $U_{z,y,x}$ is stored into the internal FPGA memory. The multiply-accumulate operations needed for the computation of the dot product of the local differences and weight vectors needed for the prediction, $\hat{d}_{z,y,x}$, are performed using the structure depicted in Figure 3.8. This structure makes it possible to 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 3.9. The number of instances of multipliers and accumulators used for the dot product and the amount of weight vector units are calculated based on the user-defined parameter *P_MAX*, following Equation 3.10:

$$Height\_tree = log_2\lceil C_z \rceil, \tag{3.10}$$

As mentioned before, The BIP-MEM architecture differs from the BIP one only in the storage of *FIFO_TOP_RIGHT* in an external memory, which can be necessary depending on the target device since the size of this FIFO is $(N_x + 1) \cdot N_z$ elements. This communication is done by using an AHB master interface, needing one read and one write operation to compress a sample and, consequently, reducing throughput compared to the BIP architecture.

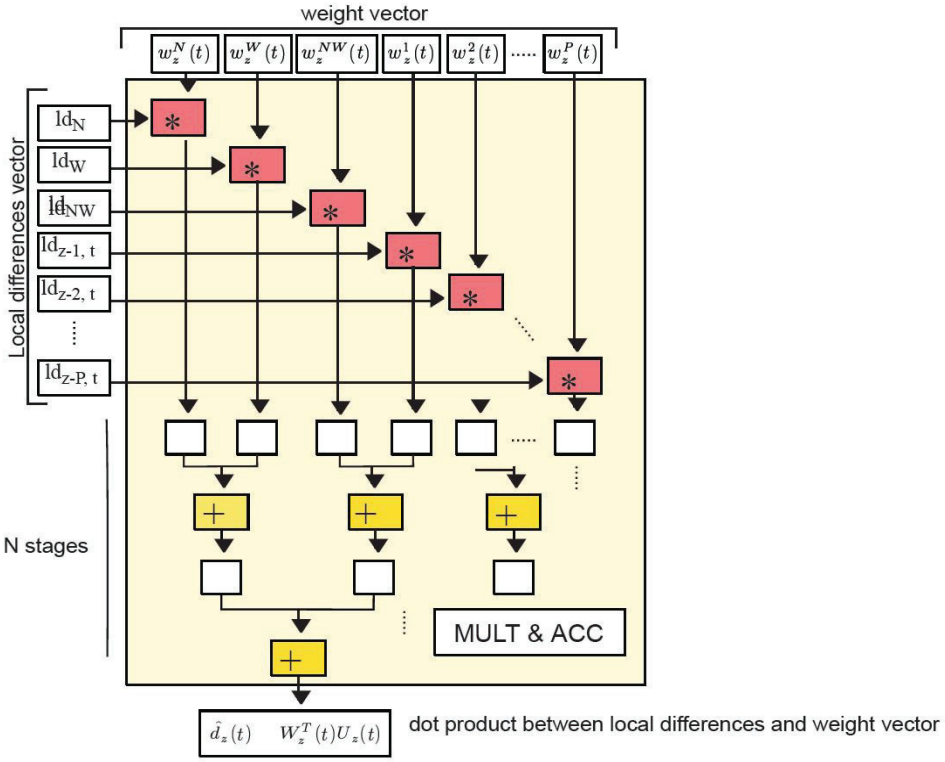


FIGURE 3.8: CCSDS 123.0-B-1 predictor - Multiply-accumulate unit to perform the dot product in BIP

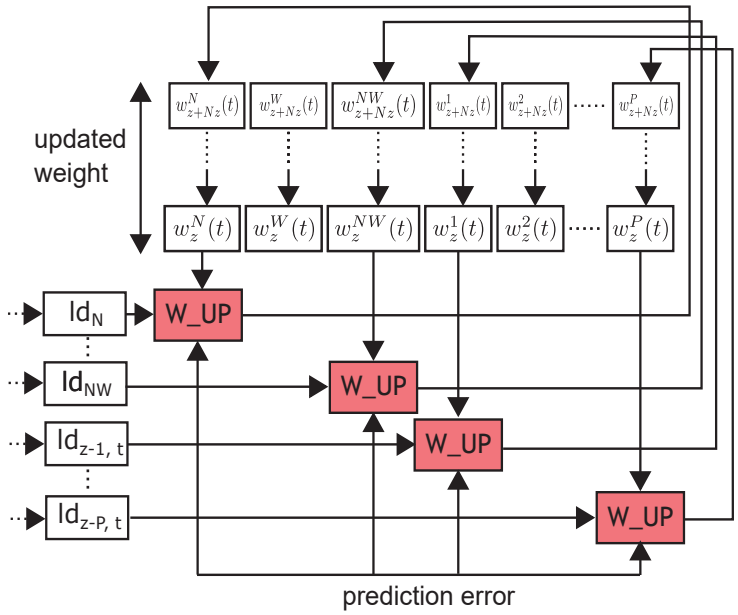


FIGURE 3.9: CCSDS 123.0-B-1 predictor - Weight update in BIP

3.3.2.2 BSQ architecture

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

The BSQ architecture requires to store a complete vector of local differences per sample during the compression of a band z. These local differences vectors (a total of $N_x \cdot N_y$) are stored in an external memory and sent through the AHB master interface. The memory addresses in which the vectors are stored are calculated by the preprocessor as shown in Figure 3.10, in such a way that the memory locations are appropriately reused when available. One local difference value needs to be stored per sample, and P values need to be read, which correspond to the central local differences. Within a vector, decreasing write addresses and increasing read addresses are used.

In BSQ, data dependencies place an important throughput limitation. Since local differences need to be retrieved from an external memory, the parallelization of the MAC and the weight update operations do not provide an improvement, as it happens in BIP and BIL (shown in Figures 3.8 and 3.9). This is because readings from the external memory constitute the main bottleneck of the BSQ architecture. For this reason, these operations are serialized in order to reduce hardware occupancy.

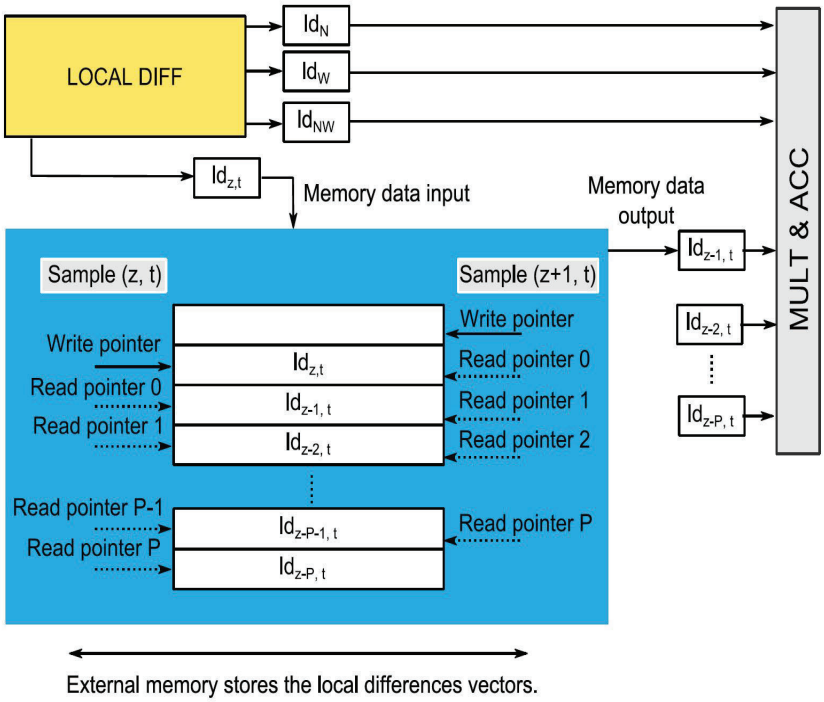


FIGURE 3.10: CCSDS 123.0-B-1 predictor - Local differences storage in an external memory in BSQ

3.3.2.3 BIL architecture

BIL inherits most of the components from the BIP architecture, including the way in which the MAC and the weight update operations are calculated. The main difference is the local differences storage. In BIL, it is necessary to store one vector per sample in a line of pixels. This storage is placed into the FPGA memory resources in several chained FIFOs that store both the central and the directional differences, in a way that they are available when requesting to process a concrete sample, as shown in Figure 3.11.

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

In the same way that for the BIP-MEM architecture, the BIL-MEM one stores the *FIFO_TOP_RIGHT*, which contains $(N_x + 1) \cdot N_z$ elements, in an external memory, whereas the BIL architecture only uses internal memory available inside the FPGA.

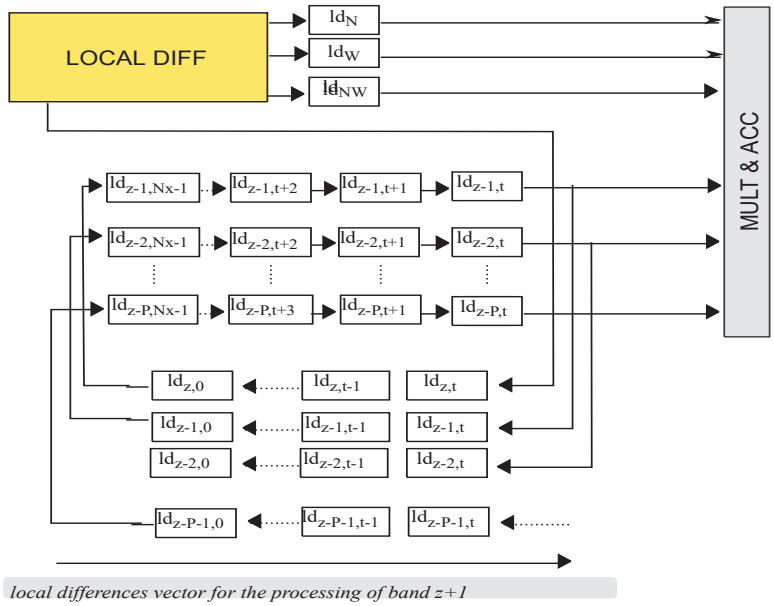


FIGURE 3.11: CCSDS 123.0-B-1 predictor - Local differences storage in BIL

3.3.3 Block characterization

As for the CCSDS 121.0-B-3 unit-delay predictor presented in Section 3.2, preliminary synthesis results for the CCSDS 123.0-B-1 prediction-based preprocessor have been obtained using Synopsys Synplify Premier P-2019.09-SP1. Table 3.4 shows the CCSDS 123.0-B-1 predictor baseline configuration values, which are common to all the preliminary mapping cases. This configuration, selected to maximize the CR taking into account the parameter tuning presented in [179], is received at compile-time and it is applied to two different acquisition scenarios: multispectral, by using a Landsat scene (1024 lines, 1024 columns and 8 bands, with $D = 8$); and hyperspectral, represented by an AVIRIS image (512 lines, 680 columns and 224 bands, with $D = 16$). For each acquisition scenario, synthesis results are provided for the five possible predictor architectures (BIP, BIP-MEM, BSQ, BIL and BIL-MEM).

Results in terms of timing are summarised in Table 3.5 for Xilinx Kintex UltraScale XCKU040. The system clock frequency achieved is above 144 MHz for all the configurations implemented. However, not all the architectures provide the same throughput because of data dependencies, being BIP the one that can reach the maximum throughput of one sample per clock cycle. This preprocessor has been designed in such a way that the AHB clock is faster than the system clock to avoid delays in the processing due to the communications with the external memory, a condition that is fulfilled in all the implemented architectures.

Table 3.4: Baseline synthesis configuration for the CCSDS 123.0-B-1 predictor

Generic	Value	Description
P_MAX	3	Number of previous bands used for prediction
PREDICTION_GEN	0	Full prediction mode
LOCAL_SUM_GEN	0	Wide-neighbour local sum
OMEGA_GEN	13	Weight resolution
R_GEN	32	Register size
VMAX_GEN	3	Weight Update Scaling Exponent Final Parameter
VMIN_GEN	-1	Weight Update Scaling Exponent Initial Parameter
T_INC_GEN	6	log_2 Weight Update Scaling Exponent Change Interval
WEIGHT_INIT_GEN	0	Default weight initialization

Table 3.5: CCSDS 123.0-B-1 predictor - Maximum frequency on Xilinx Kintex UltraScale XCKU040

Config	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	152.6	153.7	144.5	163.9	155.7
AVIRIS	150.7	152.3	148.2	154.1	154.7

Figure 3.12 show resources utilization in Kintex UltraScale XCKU040 for Landsat and AVIRIS scenes. In general, the CCSDS 123.0-B-1 preprocessor makes use of low logic resources which proves its reduced complexity. The usage of DSPs, LUTs and registers is almost constant, with slight differences depending on the selected predictor architecture and the acquisition scenario. These differences are noticeable in the case of memory usage, which is mainly determined by the predictor architecture, being BIP and BIL the ones that show a higher impact on memory resources utilization.

The use of BRAMs is proportional to the size of an spectral line ($N_x N_z$), 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. The P value 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 registers), 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 bit-width of each element of the weights vector.

For Kintex UltraScale, up to 12% of memory resources are used for the AVIRIS scenario. With respect to the DSPs and LUTs utilization, all the architectures and configurations

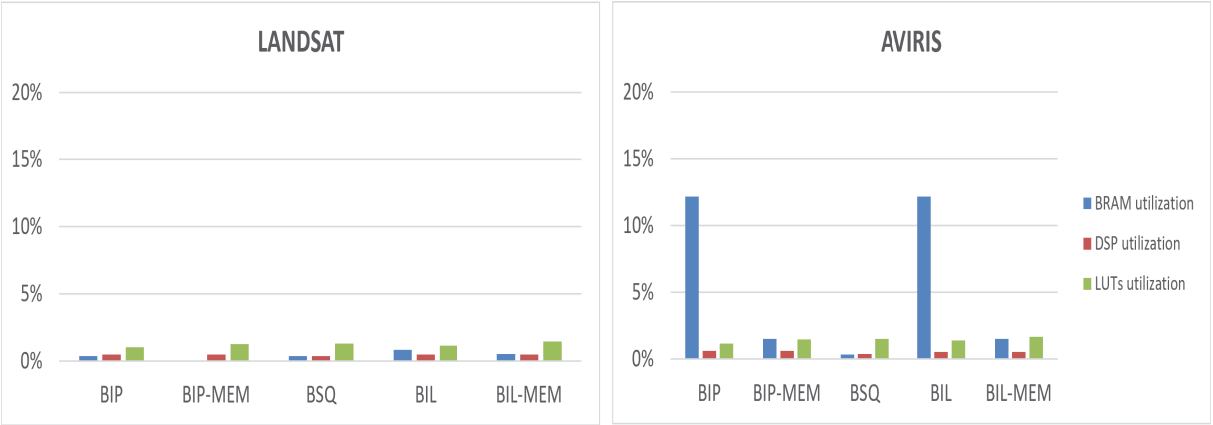


FIGURE 3.12: CCSDS 123.0-B-1 predictor - Resources utilization on Xilinx Kintex UltraScale XCKU040

provide similar results, with values around the 1.2% for DSPs and between 1.5% and 3% for LUTs. It is observed that LUT utilization is proportional to the image dimensions. It is also remarkable that DSPs utilization tends to be lower for the BSQ architecture, due to its serial implementation. In addition, BIP and BIL architectures use less LUTs compared with the other predictor alternatives, due to the absence of the AHB interface to communicate with the external memory.

The resources utilization of each internal module of the CCSDS 123.0-B-1 predictor is also analysed in Table 3.6 when implementing the BIP architecture. As it can be observed, the critical stage in terms of memory usage is the one responsible of organizing the neighbouring samples in order to be available when the current sample is going to be processed. Concretely, this module uses around the 11% of the BRAMs available in the device. On the other hand, the weights update stage is the one with the highest logic resources demand, using the 1.1% of the available LUTs. Although these results are obtained when the BIP architecture is implemented, minimum differences are observed with respect to BSQ and BIL orders.

TABLE 3.6: CCSDS 123.0-B-1 predictor - Resources utilization on Xilinx Kintex UltraScale XCKU040 per module in BIP order

Module	Total	Vicinity Disposition	Local Sums	Local Diffs	Prediction	Weights Update	Mapper
Block RAMs	600	67	0	0	0	6	0
DSP48	1920	0	0	0	7	1	0
Registers	484800	119	189	219	398	817	104
LUTs	242400	149	180	186	370	2586	171

As a summary, and taking into account the obtained results in terms of both throughput and resources utilization, the proposed implementation of the CCSDS 123.0-B-1 prediction-based preprocessor is considered an efficient approach for spatial and spectral decorrelation of hyperspectral images acquired on EO satellites. The results in terms of timing when the BIP architecture is selected meet real-time requirements for the HSI sensors embarked on current space missions, since a throughput of 1 sample per clock cycle is achieved. This performance is reached together with a low area footprint and a high flexibility, which allows to adapt the predictor behaviour to the target application to maximize the compression ratio.

3.4 CCSDS 123.0-B-2 spectral and spatial decorrelator for HSI near-lossless compression

3.4.1 Algorithm overview

The CCSDS 123.0-B-2 standard includes several modifications regarding its predecessor, mainly aimed at supporting near-lossless compression. A new feature is the calculation of a high-resolution predicted sample $\breve{s}_z(t)$, used to improve the precision of the prediction computation. Then, a quantizer is introduced in the prediction chain, in order to introduce losses. Data loss is controlled by the maximum error limit $m_z(t)$, which can be absolute or relative to the sample magnitude. A relevant novelty regarding Issue 1 is the calculation of the sample representatives $s''_{z,y,x}$, which are a reconstruction of the input samples after the quantization to perform the prediction of the next sample in the same way that it is done during the decompression, where input samples are not available.

The block hierarchy of the prediction stage is shown in Figure 3.13, remarking the new modules regarding Issue 1 for supporting near-lossless compression.

Issue 2 of the standard also introduces *narrow* local sums, which avoid the use of the sample representative at the left of the current one in the same band (i.e., $s''_{z,y,x-1}$), which is replaced by the sample representative in that position but in the previous band (i.e., $s''_{z-1,y,x-1}$), favouring in this way optimization strategies on hardware for improving throughput. Equations 3.11 and 3.12 describe how the local sums are calculated under the narrow neighbour-oriented and column-oriented modes, respectively, where s_{mid} represents the mid-range sample value. Sample representative values $s''_{z,y,x}$ are used to compute the

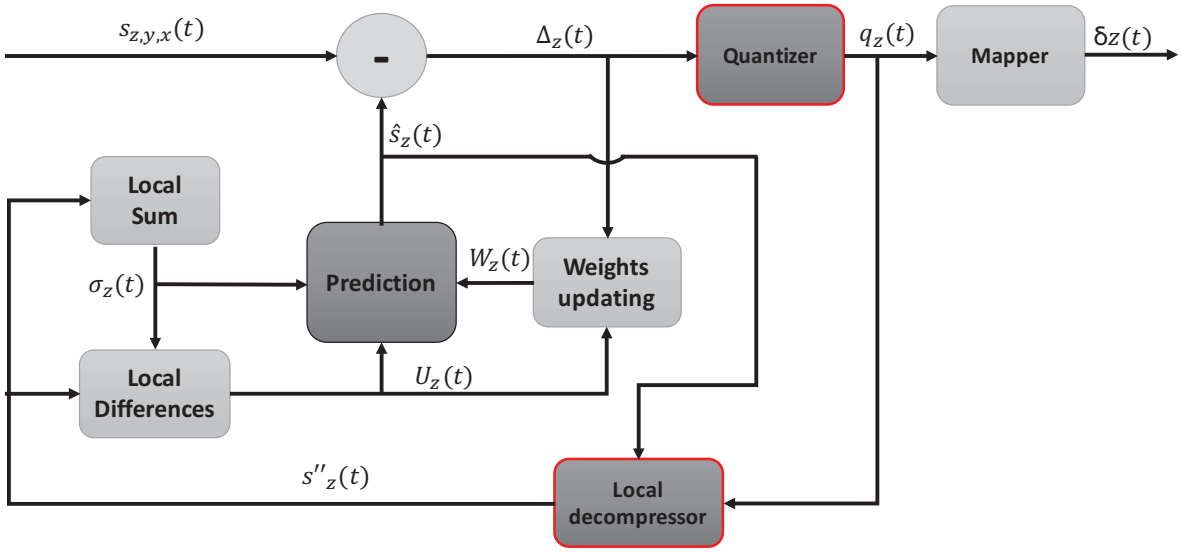


FIGURE 3.13: CCSDS 123.0-B-2 predictor overview

local sums, if near-lossless compression is selected; otherwise, input samples are directly employed, as it is done in Issue 1 of the standard and described in Section 3.3.1.

$$\sigma_{z,y,x} = \begin{cases} s''_{z,y-1,x-1} + 2s''_{z,y-1,x} + s''_{z,y-1,x+1}, & y > 0, 0 < x < N_x - 1 \\ 4s''_{z-1,y,x-1}, & y = 0, x > 0, z > 0 \\ 2(s''_{z,y-1,x} + s''_{z,y-1,x+1}), & y > 0, x = 0 \\ 2(s''_{z,y-1,x-1} + s''_{z,y-1,x}), & y > 0, x = N_x - 1 \\ 4s_{mid} & y = 0, x > 0, z = 0 \end{cases} \tag{3.11}$$

$$\sigma_{z,y,x} = \begin{cases} 4s''_{z,y-1,x}, & y > 0 \\ 4s''_{z-1,y,x-1}, & y = 0, x > 0, z > 0 \\ 4s_{mid}, & y = 0, x > 0, z = 0 \end{cases} \tag{3.12}$$

Once the value of the *predicted central local difference* $\tilde{s}_z(t)$ is computed, in the same way as in Issue 1 of the standard, it is used to calculate the *high-resolution predicted sample* $\breve{s}_z(t)$ which is also employed next to compute the *double-resolution predicted sample* $\tilde{s}_z(t)$. These values are calculated according to formulas 37 and 38 in [31]. Then, $\tilde{s}_z(t)$ is used to obtain the *predicted sample* as

$$\hat{s}_{z,y,x} = \left\lfloor \frac{\tilde{s}_z(t)}{2} \right\rfloor, \tag{3.13}$$

Alternatively, the predicted sample can be simplified as [180]

$$\hat{s}_{z,y,x} \approx \left\lfloor \frac{\hat{d}_{z,y,x} + 2^{\Omega}\sigma_{z,y,x}}{2^{\Omega+2}} \right\rfloor, \tag{3.14}$$

assuming that the value of the register size R is high enough to avoid overflow in the computation of $\breve{s}_z(t)$. In this way, it is possible to obtain the predicted sample value without previously calculating both the high- and the double-resolution terms. Under near-lossless compression, the *prediction residual* $\Delta_z(t)$ feeds the quantizer, while it is directly mapped into an unsigned integer $\delta_{z,y,x}$ and passed to the entropy coder under lossless compression, as in Issue 1 of the standard. The quantizer employs a uniform bin size with a value of $2m_z(t)+1$, being $m_z(t)$ being the maximum error limit defined by the user. $m_z(t) = 0$ implies lossless compression, where the input image can be fully reconstructed. By increasing $m_z(t)$ the compression ratio improves at the cost of introducing quantization noise, which affects the quality of the reconstructed image. The quantizer index $q_z(t)$ is computed as

$$q_z(t) = sgn(\Delta_z(t)) \left\lfloor \frac{\mid \Delta_z(t) \mid + m_z(t)}{2m_z(t)+1} \right\rfloor. \tag{3.15}$$

The maximum error limit $m_z(t)$ is controlled by defining a maximum *absolute error* a_z and/or a *relative error* limit r_z. These errors can be identical for the whole image (i.e. band-independent) or different for each band z (i.e. band-dependent). The latter is useful when the target application requires specific spectral channels to be preserved with higher fidelity. The value of each error option is limited by its dynamic range, D_a or D_r considering absolute and/or relative errors, respectively. D_a and D_r should be in the range $1 \leq D_a, D_r \leq min(D-1, 16)$, being D the dynamic range of the input samples. The maximum error is defined as $m_z(t) = a_z(t)$ or

$$m_z(t) = \left\lfloor \frac{r_z(t) \mid \hat{s}_z(t) \mid}{2^D} \right\rfloor, \tag{3.16}$$

according to the type of error limits selected, absolute and/or relative. In case that both error limits are used, $m_z(t)$ takes the most restrictive value.

Using absolute errors guarantees that the maximum absolute difference between $s_z(t)$ and $q_z(t)$ is limited to a certain magnitude, while relative errors allows samples to be reconstructed with different precision and, consequently, reconstructing with lower error those samples with lower magnitude. Lossless compression is supported by defining $m_z(t) = 0$, implying that $q_z(t) = \Delta_z(t)$.

Sample representatives, introduced at the beginning of this subsection, are also needed to reduce the impact of the quantization, approximately reconstructing the original samples $s_z(t)$, in the same way that the decompressor does. For this reason, $s''_z(t)$ are used to compute the predicted sample, instead of $s_z(t)$. Three user-defined parameters are used to control the deviation of the sample representatives values from the quantizer bin center $s'_z(t)$ (i.e., the discretized value of the predicted sample, taking into account the selected quantization step): the sample representative resolution Θ; the damping ϕ_z, which limits the effect of noisy samples during the calculation of $s''_z(t)$; and the offset ψ_z, which tunes the sample representative value towards $s'_z(t)$ or $\hat{s}_z(t)$. The range of allowed values for both ϕ_z and ψ_z is limited between 0 and $2^\Theta - 1$, being also possible to define a different value for each band z. Setting ϕ_z and ψ_z to 0 ensures that $s''_z(t)$ is equal to $s'_z(t)$, while higher values of one and/or both of them result in values closer to $\hat{s}_z(t)$. Non-zero values for ϕ_z and ψ_z tends to provide higher compression performance if hyperspectral images have a high spectral correlation between adjacent bands, as it is claimed in [181]. Then, the clipped version of the quantizer bin center $s'_z(t)$ is obtained following

$$s'_z(t) = clip(\hat{s}_z(t) + q_z(t)(2m_z(t) + 1)), \{s_{min}, s_{max}\}), \tag{3.17}$$

being s_{min} and s_{max} the lower and upper limits in the range of possible sample values, which is directly dependent on the dynamic range D. If $m_z(t) = 0$, then $s'_z(t) = s_z(t)$. The sample representative, $s''_z(t)$, is obtained as

$$s''_z(t) = \begin{cases} s_z(0), t = 0 \\ \left\lfloor \frac{\tilde{s}''_z(t)+1}{2} \right\rfloor, t > 0, \end{cases} \tag{3.18}$$

being $\tilde{s}''_z(t)$ the *double-resolution sample representative*, which is calculated as

$$\tilde{s}''_z(t) = \left\lfloor \frac{4(2^{\Theta} - \phi_z) \cdot (s'_z(t) \cdot 2^{\Omega} - sgn(q_z(t)) \cdot m_z(t) \cdot \psi_z \cdot 2^{\Omega-\Theta}) + \phi_z \cdot \breve{s}_z(t) - \phi_z \cdot 2^{\Omega+1}}{2^{\Omega+\Theta+1}} \right\rfloor, \tag{3.19}$$

taking into account the value of the quantization bin center and the high-resolution predicted sample, in addition to some user-defined parameters.

Backwards compatibility is guaranteed with Issue 1 of the standard, in case that the following assumptions are accomplished:

- The maximum dynamic range D has to be less or equal to 16.
- Narrow local sums are not chosen.
- The quantizer fidelity control method has to be set to lossless (i.e., $m_z(t) = 0$).
- Set the sample representative parameters (the resolution Θ, the damping ϕ_z and the offset ψ_z) to 0.
- Supplementary tables are not part of the header.

3.4.2 Block design

Unlike the CCSDS 123.0-B-1 standard, where BIP order is the one that achieves the maximum throughput, data dependencies introduced in the CCSDS 123.0-B-2 algorithm imposes a penalty when processing in this arrangement. The introduction of the sample representative calculation is identified as the bottleneck, since they are necessary to compute local sums and local differences instead of using the original input samples. In the case of the BIP ordering, the representative of the previous sample is not used to compute local sums, but it is required to compute the central local difference of the previous band. This introduces a strong data dependency, in which a sample must be almost fully processed before the processing of the next one can start, as shown in Figure 3.14.

BIL and BSQ orderings in the CCSDS 123.0-B-1 standard present a strong data dependency in the weights updating, where the inner product to obtain the predicted central local difference $\hat{d}_{z,y,x}$ for the current sample $s(t)$ cannot be computed until the weights are

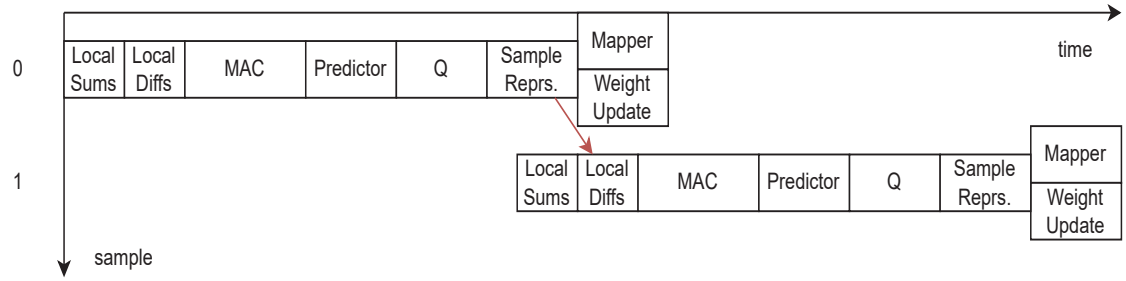


FIGURE 3.14: CCSDS 123.0-B-2 algorithm - BIP schedule

updated for the previous sample $s(t-1)$. This dependency still applies in the CCSDS 123.0-B-2 standard. In addition, the dependencies with sample representatives in the neighbouring of the one under process can appear, depending on the selected local sum and prediction mode. The critical datapath can be softened by combining the reduced prediction mode with narrow local sums. In this situation, the dependency on sample representatives is removed (provided that the image is large enough in the spatial dimension) and the only limiting factor is the dependency on weights, as reflected in Figure 3.15. There is an additional optimization which can be performed in relation with weights updating, which is based on [182]. It consists in the computation of both possible results of the prediction error $e_z(t)$ in parallel with the sample representative calculation. Then, when the sample representative and the quantizer bin center are computed, the appropriate term is selected, depending on the sign of $e_z(t)$. This allows to slightly improve the throughput at the cost of duplicating the logic used for weights updating.

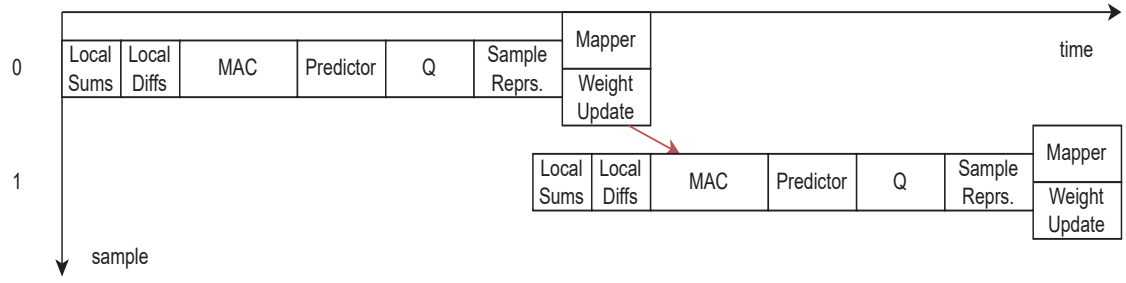


FIGURE 3.15: CCSDS 123.0-B-2 algorithm - BSQ and BIL schedule with reduced prediction and narrow local sums

Taking into account this analysis, the predictor will be processed the samples in BIL order. Although it is expected that BSQ provides similar performance than BIL, EO space missions commonly integrate pushbroom sensors, which makes possible to compress samples on-the-fly if they are processed in BIL order. Otherwise, a samples reordering

module is needed prior to the compression stage, compromising the throughput of the processing chain.

Therefore, the CCSDS 123.0-B-2 predictor architecture has been developed in BIL order and following an HLS design methodology. This solution includes all the necessary functional units to generate a mapped prediction residual $\delta_z(t)$ from a given input sample $s_{z,y,x}$. The top-level module receives the user-defined predictor parameters, together with the image size and the dynamic range D through an AXI4-Lite interface in compile-time, providing an interface that is suitable for configuration purposes that work at low data rates. This configuration can be changed also in runtime, but the preprocessor operation should stop until the new configuration is overwritten in the defined AXI4-Lite configuration registers.

Input samples are received by using a lightweight AXI4-Stream interface, which is thought for burst transfers at high data rates, including also a simple handshaking protocol for synchronisation purposes, comprised by a *Valid* and *Ready* signals. Migration from AMBA AHB, used in previous prediction-based preprocessor developments, to AXI interfaces is imposed by the HLS tool used, which automatically infers this kind of interfaces for the most recent Xilinx devices. Anyway, input and output data interfaces can be synthesized with a native protocol (i.e. as a custom or ad-hoc interface), if it is required to be connected to other modules, such as an entropy coder. At the output, the mapped prediction residuals $\delta_z(t)$ are stored in an intermediate FIFO, which then feeds the selected encoding stage.

This functional block is comprised by a total of 9 functional units, as shown in Figure 3.16. Additional effort is required to implement the CCSDS 123.0-B-2 predictor compared to the one defined by its predecessor. More complex arithmetic operations (mainly multiplications and non power-of-two divisions) are introduced in the quantization loop to support near-lossless compression. Two memories are created at top-level, which are then used by the internal functional units: *topSamples*, which stores the previous spectral line ($N_x \cdot N_z$ samples), needed to perform both the local sum and the directional local differences (if full prediction mode is selected); and *currentSamples*, responsible of storing $N_x \cdot P$ samples to compute the central local differences. For as long as the image is being processed, the samples which are not needed anymore to calculate the central local differences because they fall more than P bands behind the sample being currently processed, are moved from *currentSamples* to *topSamples*. In the same way, samples in *topSamples* are replaced each time a spectral line is fully predicted, prior to start the processing of the next line. The sample at the same position that the current one but in the previous band $s_{z-1,y,x}$ is also

stored in a register, since it is used to calculate the double-resolution predicted sample $\tilde{s}_z(t)$.

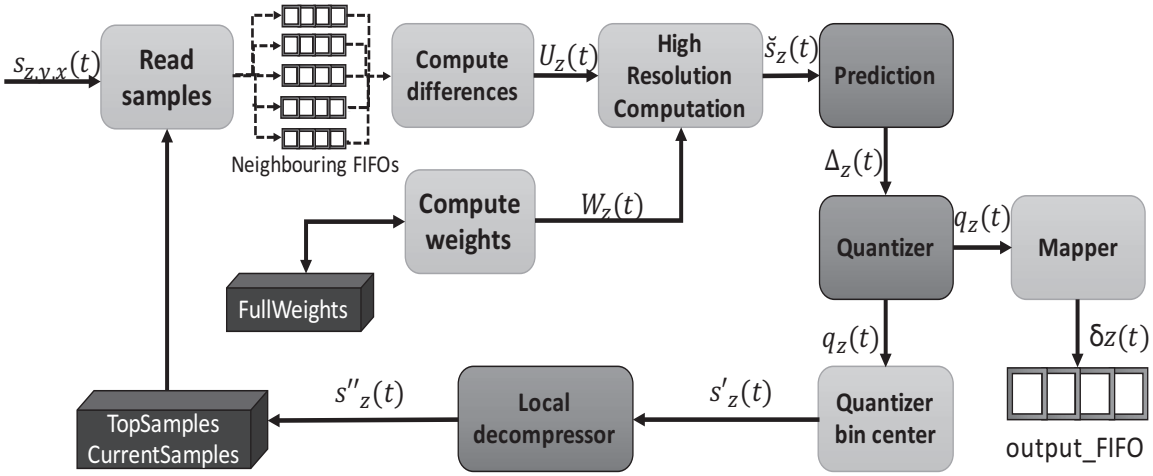


FIGURE 3.16: CCSDS 123.0-B-2 predictor block diagram

First of all, the previously preprocessed samples, which are stored in *currentSamples* and *topSamples*, and the current one are ordered in five FIFOs in the *Read_samples* module, corresponding to the current position and the vicinity needed to perform both the local sum and the directional local differences (i.e., samples at the left, top-left, top and top-right positions). These FIFOs store P elements.

Both the local sum and the local differences are computed in the *Compute_differences* module. The selection of the local sum and prediction mode is defined at compile-time by user-defined parameters. The local differences are calculated simultaneously in a single clock cycle independently of the selected prediction mode, since there are not data dependencies.

Once the local differences vector $U_z(t)$ is available, $\hat{d}_z(t)$ is computed. After that, all the variables needed to calculate the high-resolution predicted sample are available, obtaining $\check{s}_z(t)$ in the *High_resolution_prediction* unit. The result of this operation is truncated taking into account the value of the register size R, though this calculation can be omitted if it is guaranteed that the size of the operation result never exceeds R. Then, the double-resolution predicted sample $\tilde{s}_z(t)$ and the predicted sample $\hat{s}_{z,y,x}$ are calculated in the *Prediction* module, using right shifts operations to implement the divisions by power of two. Finally, the output of this unit is the prediction residual, $\Delta_z(t)$. As this submodule only employs simple arithmetic and logic operations, its process takes a single clock cycle.

If $m_z(t) \neq 0$ (i.e., near-lossless compression), the next step is the *Quantizer* unit; otherwise, it is bypassed. The quantizer is a critical point in the datapath, since it makes use of a division that it is not efficiently implemented by HLS tools, considerably delaying the whole process. In the proposed solution, a new approach is presented, substituting the division by a multiplication by the inverse of the division, operation that is highly optimised by using the internal DSPs of the Kintex UltraScale FPGA technology. For this purpose, a LUT is previously defined with the result of the division $\frac{1}{X}$ in fixed-point (i.e., integer precision), being X the different divisor values in the range of error limits defined by the target application. The required LUT size would be of $2^{D_a} - 1$ words if only absolute errors are used, $2^{D_r} - 1$ words if only relative errors are used, and the minimum of both values if both error types are defined. To avoid rounding issues for certain divisors, the computation of the inverse values of denominator is done in excess, depending on the selected error control method. Following this strategy also in the divisions performed in the *Mapper* unit, the responsible of generating the unsigned mapped residuals $\delta_z(t)$, a reduction of around the 30% of the predictor latency is observed, compared with the version that implements directly the division.

The *Quantizer_bin_center* and the *Local_decompressor* units are the responsible of performing the samples reconstruction, in order to properly estimate the value of the sample representative $s''_{z,y,x}$, used during the processing of the next image samples. While the first calculates the bin center $s'_z(t)$ in one clock cycle, as indicated in Equation 3.17, the latter estimates the value of $s''_{z,y,x}$ only if the user-defined parameter $\Theta \neq 0$; otherwise, $s''_{z,y,x} = s_{z,y,x}$. The calculation of the sample representative value is optimized for a hardware implementation, substituting power-of-two operations and the division reflected in Equation 3.18 by logical shifts. Since the rest of parameters used for the calculation are previously known, including the sign of the quantized index $q_z(t)$, this step is also executed in only one clock cycle.

Finally, the prediction chain includes the *Compute_weights* module, which performs both the weights initialization or their update, depending on the current image coordinates. The latency of the weights updating directly depends on the selected prediction mode, which defines the number of components in the weights vector. However, this process is done simultaneously for the different weight components, since there are not data dependencies among them. In addition, the proposed solution takes into account data dependencies with adjacent samples during the weights update when the compression is done in BIL order, trying to reduce the impact of those dependencies in the global throughput. An extra memory is defined in the design, denoted as *FullWeights*, whose size is $N_z \cdot C_z$ and

used to store the state of the weights vector at each band z, recovered them to process the next image pixel in the same band.

3.4.3 Block characterization

Since the CCSDS 123.0-B-2 predictor has been modelled following an HLS design flow and using Xilinx Vitis HLS tool (v2020.2), it is mapped on Kintex UltraScale XCKU040. The reason is that it is equivalent to the space-grade XQRKU060 FPGA. Taking into account the strengths of this workflow, it has been used to perform a design space exploration of the CCSDS 123.0-B-2 predictor at early stages of the development process to analyse the impact of implementing specific set of predictor features, in terms of resources utilization. Results derived from this study are relevant to approximately know the predictor footprint once it is going to be integrated in a full compression chain with other additional stages, such as an entropy coder. For all the configurations under analysis, the predictor is synthesized reaching a maximum clock frequency of 125 MHz.

The baseline configuration used for synthesis purposes is summarised in Table 3.7, restricting supported image dimensions to the ones of the AVIRIS scenes, since they are the ones used for verification and validation purposes. The value of D is also fixed to 16 to target AVIRIS or a similar sensor. The rest of parameter values have been selected trying to maximize the CR in lossless mode. In any case, all these parameter values can be changed in runtime, if needed. The maximum error limit, using absolute and/or relative values, can be changed depending on the target application and the required compression performance.

First of all, the different combinations of local sum and prediction modes are analysed. In Table 3.8, results are provided for neighbour-oriented local sums, distinguishing between wide neighbour-oriented (WN) and narrow neighbour-oriented (NN). In the same way, Table 3.9 presents the results for column-oriented local sums, including wide column-oriented (WC) and narrow column-oriented (NC). For the different situations under study, the consumption of BRAMs and DSPs is identical, since not specific memory requirements or extra arithmetic calculations are added based on the selected local sum and prediction mode. Registers usage is also equal for the defined combinations. Regarding LUTs consumption, slight differences are appreciated, increasing this usage when full prediction mode and wide local sums are selected. In any case, these differences are minimum, implying a difference around the 0.4% when full prediction mode is selected instead of the

TABLE 3.7: Baseline synthesis configuration for the CCSDS 123.0-B-2 predictor

Parameter	**Value**
Image parameters	
Columns, N_x	677
Lines, N_y	512
Bands, N_z	224
Dynamic Range, D	16
Encoding Order	BIL
Predictor parameters	
Bands for Prediction, P	3
Weight Resolution, Ω	16
Sample Adaptive Resolution, Θ	2
Sample Adaptive Offset, ψ_z	1
Sample Adaptive Damping, ϕ_z	1
Absolute Error Bitdepth, D_a	8
Relative Error Bitdepth, D_r	8

TABLE 3.8: CCSDS 123.0-B-2 predictor - Resources utilization on Xilinx Kintex UltraScale XCKU040 depending on the combination of local sum and full prediction

Config	**Total**	**WN+full**	**WN+reduced**	**NN+full**	**NN+reduced**
36Kb BRAM	600	7	7	7	7
DSP48E	1920	9	9	9	9
Registers	484800	1894	1894	1894	1894
LUTs	242400	3994	3980	3990	3976

TABLE 3.9: CCSDS 123.0-B-2 predictor - Resources utilization on Xilinx Kintex UltraScale XCKU040 depending on the combination of local sum and reduced prediction

Config	**Total**	**WC+full**	**WC+reduced**	**NC+full**	**NC+reduced**
36Kb BRAM	600	7	7	7	7
DSP48E	1920	9	9	9	9
Registers	484800	1894	1894	1894	1894
LUTs	242400	3990	3980	3984	3975

reduced one (directional local differences are taking into account), and a 0.1% when wide local sums are used instead of the new narrow option.

Higher differences are observed depending on the selected compression mode, as reflected in Table 3.10. Lossless compression mode with narrow neighbour-oriented local sums and full prediction mode is used as baseline for this comparison. Just introducing the sample reconstruction stage implies a huge increment in the memory usage regarding the purely lossless mode, because of the introduction of the *topSamples* and *currentSamples* memories, used to store sample representatives of previously processed pixels used to predict the current one. Besides, DSPs usage is increased a 200% by introducing sample representatives

TABLE 3.10: CCSDS 123.0-B-2 predictor - Resources utilization on Xilinx Kintex UltraScale XCKU040 depending on the applied absolute error limit

Config	Total	Lossless	Lossless (S. Representative)	Near-lossless (Absolute error)	Near-lossless (Relative error)
36Kb BRAM	600	7	76	76	76
DSP48E	1920	9	27	31	32
Registers	484800	1894	5257	6569	6673
LUTs	242400	3990	10940	15767	15812

computation. An extra DSP consumption is appreciated under near-lossless compression to compute optimized multiplications in the quantizer stage, which is dependent on the maximum error limit used.

Regarding logic resources utilization, an increment of the 177% and 174% is appreciated in the registers and LUTs usage, respectively, when samples reconstruction is introduced. In addition, working in near-lossless mode implies an average increment around a 26% and 44% in the registers and LUTs utilization, respectively, compared to the lossless architecture with samples reconstruction and depending on the maximum error limit applied. In any case, taking the most restrictive alternative in terms of resources utilization (i.e. near-lossless compression applying relative error limits), just the 12.6% and 1.7% of the BRAMs and DSPs available in the device are used, respectively. Regarding logic resources consumption, the 1.4% and 6.5% of the available registers and LUTs are consumed, respectively.

Finally, as it was also done for the CCSDS 123.0-B-1 predictor in Section 3.3.3, the resources utilization of each internal module is obtained, mainly to analyse the impact of introducing the new modules to support near-lossless compression. These results are shown in Table 3.11. As for the CCSDS 123.0-B-1 predictor, the most memory consuming stage is the one that organize the neighbouring for the local sum and local differences calculation, which is in this approach also the responsible of properly managing the *topSamples* and *currentSamples* memories. This module uses the 24.5% of the BRAMs available in the Kintex UltraScale XCKU040 FPGA when targeting AVIRIS scenes.

In terms of logic resources utilization, the optimization that replaces the division performed in the quantizer by a multiplication implies the reduction of its complexity, consuming just some LUTs and 1 DSP to perform that multiplication. The modules that use more LUTs are again the weights update, as in the CCSDS 123.0-B-1 predictor; the prediction itself, which is more complex in this new Issue of the standard for the introduction of the high-resolution predicted sample; and the local decompressor stage (i.e. the sample

representatives computation), which also performs some internal multiplications. The highest LUT utilization is around the 1.5% for the weights update module, which is considered minimum. The sample representatives stage is the one that uses more DSPs (around the 0.2% of the total available).

TABLE 3.11: CCSDS 123.0-B-2 predictor - Resources utilization on Xilinx Kintex UltraScale XCKU040 per submodule

Module	Total	Read Samples	Local Sums	Local Diffs	Prediction	Weights Update
36Kb BRAM	600	147	0	0	0	4
DSP48E	1920	1	0	0	0	1
Registers	484800	174	0	169	390	1048
LUTs	242400	900	398	1096	2021	3524

Module	Total	Quantizer	Local Decompressor	Mapper
36Kb BRAM	600	1	0	1
DSP48E	1920	1	4	2
Registers	484800	111	664	210
LUTs	242400	240	1849	559

These numbers demonstrate the viability to implement the presented HLS implementation of the CCSDS 123.0-B-2 predictor in a commercial FPGA equivalent to a space-grade one, in terms of logic resources availability. In addition, obtained results will allow the application of mitigation techniques against radiation effects, since enough logic resources are available to apply hardware redundancy. As a final conclusion, it can be observed that resources utilization, specially the memory usage, is closely related to the target sensor and the acquired scene size.

3.5 Conclusions

In this Chapter, three different prediction-based preprocessors have been developed, all of them compliant with the CCSDS standards. The unit-delay predictor, based on the CCSDS 121.0-B-3 standard, is focused on one-dimensional data decorrelation. The 3D predictors compliant with the CCSDS 123.0-B-1 and 123.0-B-2 standards provide both spatial and spectral decorrelation when processing multi- and hyperspectral images under lossless and near-lossless compression, respectively.

Results have been provided for each one of these approaches targeting the Xilinx Kintex UltraScale XCKU040, demonstrating the goodness of the proposed functional blocks to fit well on a space-grade FPGA. Besides, characterization results have been obtained for other radiation-tolerant FPGA technologies (e.g. NanoXplore Brave NG FPGAs or Microsemi RTG4) that were not included in this document for the sake of conciseness.

Chapter 4

Design and characterization of entropy coding blocks

In this Chapter, hardware implementations of different entropy coders are presented as functional blocks, which can be easily integrated as part of a whole compression chain. Entropy coders aim at reducing as much as possible the compressed bitstream, by representing the prediction residuals with the minimum possible number of bits to ensure a proper decompression. The developed implementations function as described in the CCSDS 121.0-B-3, 123.0-B-1 and 123.0-B-2 compression standards, which define the entropy coders named block-adaptive, sample-adaptive and hybrid, respectively. A theoretical introduction to these algorithms and the architectural design implementing each alternative are provided. Besides, some preliminary synthesis results in a space-representative FPGA are presented to demonstrate the viability of implementing these encoding solutions on-board satellites.

4.1 Outline

In addition to a preprocessor, the entropy coding constitutes a key stage in a compression chain. It is the responsible of coding the received prediction residuals with the minimum possible number of bits. As in the case of prediction-based preprocessors presented in Chapter 3, the CCSDS also propose different entropy coding alternatives in the published data compression standards, which provide low-complexity solutions to be used by the space industry when it comes to on-board data processing, ensuring at the same time an acceptable RD ratio.

This Chapter presents three different entropy coding solutions developed as functional blocks, which can be connected to the developed prediction-based preprocessors in a plug and play manner to conform full compression chains, depending on the mission requirements in terms of compression ratio, throughput or hardware occupancy. Both RTL and HLS design methodologies are considered, selecting the appropriate workflow depending on the constraints imposed by each solution development, in terms of both performance and available design time.

Two of these solutions implement the block-adaptive and the sample-adaptive encoders, described in the CCSDS 121.0-B-3 [27] and 123.0-B-1 [30] lossless compression standards, respectively, as VHDL technology-agnostic solutions. These two entropy coding implementations have been then integrated as part of SHyLoC [59, 60]. The RTL design methodology has been followed in this case to maximize the throughput of the proposed solutions, with the purpose of reaching real-time capabilities. At the same time, these optimized approaches are developed keeping in mind a low area footprint, which is a main constraint for on-board processing solutions.

In addition, an HLS implementation of the hybrid encoder proposed in the CCSDS 123.0-B-2 near-lossless compression standard is presented, in order to obtain a compression solution fully compliant with that standard, by joining this encoder to the 3D predictor presented in Section 3.4. In this way, the prototyping of this complex algorithm is possible in a short time, analysing at early stages of the space mission program the viability of implementing it in space-grade Xilinx Kintex UltraScale FPGA technology. In any case, this encoder can be also connected to other preprocessing stages by defining the appropriate data interfaces during its synthesis.

Finally, preliminary synthesis results on a space-grade FPGA, specifically the Xilinx Kintex UltraScale XCKU040, are presented for these entropy coding modules, demonstrating in this way their feasibility to work on-board satellites. Representative configuration sets are defined for each approach, in order to analyse both the performance and the hardware occupancy under different possible scenarios.

4.2 CCSDS 121.0-B-3 block-adaptive encoder

4.2.1 Algorithm overview

The entropy coder described in the CCSDS 121.0-B-3 standard is based in adaptive Rice coding [27], a subset of Golomb codes that uses a power of two as tunable parameter. This parameter eases the hardware implementation, since multiplications and divisions by powers of two are efficiently implemented applying logic shifts. This entropy encoder achieves suitable performance over different overlapping ranges of entropy, since it takes into account the variability of the geometrical distribution of the alphabet along the time, not calculating the optimal code, such as Golomb coding, but the one that it is more adjusted to the current input samples. The incoming samples (preprocessed or not, depending on the presence or absence of a prediction stage, respectively) 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 of J samples is coded with the compression 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. It is the most basic option.
- **Sample splitting**. First, each input sample is split by removing the k least significant bits. The MSBs of δ_i $(n-k)$ are coded with the FS, while the LSBs are left uncompressed.
- **Second-extension**. In this low-entropy option 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**. The input samples are flushed unaltered (i.e. the entropy coder is bypassed).

A unique identifier (ID) is attached to each compressed block for all the aforementioned compression options. The standard also provides the possibility for the user to choose either a basic or a restricted set of coding options. Nonetheless, the restricted codeset is just selectable when $D \leq 4$, reducing in this case the number of available coding options and thus allowing the use of shorter ID bit sequences.

In addition, parameters used during the entropy coding stage are included in a header, which is attached at the beginning of the encoded bitstream and whose structure depends on the selected header generation mode: packet or file format. This latter option is the main novelty of Issue 3 of the standard, codifying the value of the parameters taken account during encoding. This information is required on the decompression side to be able to efficiently reconstruct the original data, knowing also which encoding option has been used for each group of J samples, thanks to the compressed block ID.

In case two or more compression options produce a codeword with the same bit length for an specific group of J samples and it is identified as the shortest one, some precedence rules are established:

1. The *No compression* option has the highest priority, being selected in case of tie with other coding options.
2. The next one in order of priority is the *Second-extension*, which should be chosen when it minimizes the encoded length.
3. Otherwise, the coding option having the smallest code parameter value k should be chosen.

4.2.2 Block design

The block-adaptive encoder includes, in addition to the encoding functionality, the necessary logic to receive the runtime configuration through the AHB slave and the mapped residuals δ_i, using in this case an ad-hoc dedicated interface. The block-adaptive encoder diagram is depicted in Figure 4.1.

First, the *int* module reads the runtime configuration, validating the values received and raising an error if the configuration values are out of range. If runtime configuration is disabled, this module is bypassed throughout the datapath, directly working with the compile-time configuration.

Then, the configuration values are sent to the *header_gen* module, which generates the different header fields depending on the selected header format, allowing to select between packet and the new file format, introduced in Issue 3 of the standard. This selection is done through the *HEADER_FORMAT* generic, which is set to 0 to generate the header according to the packet format, and it is equal to 1 if file format is selected. The header can contain, in addition to fields related to the encoder parameters and the input data

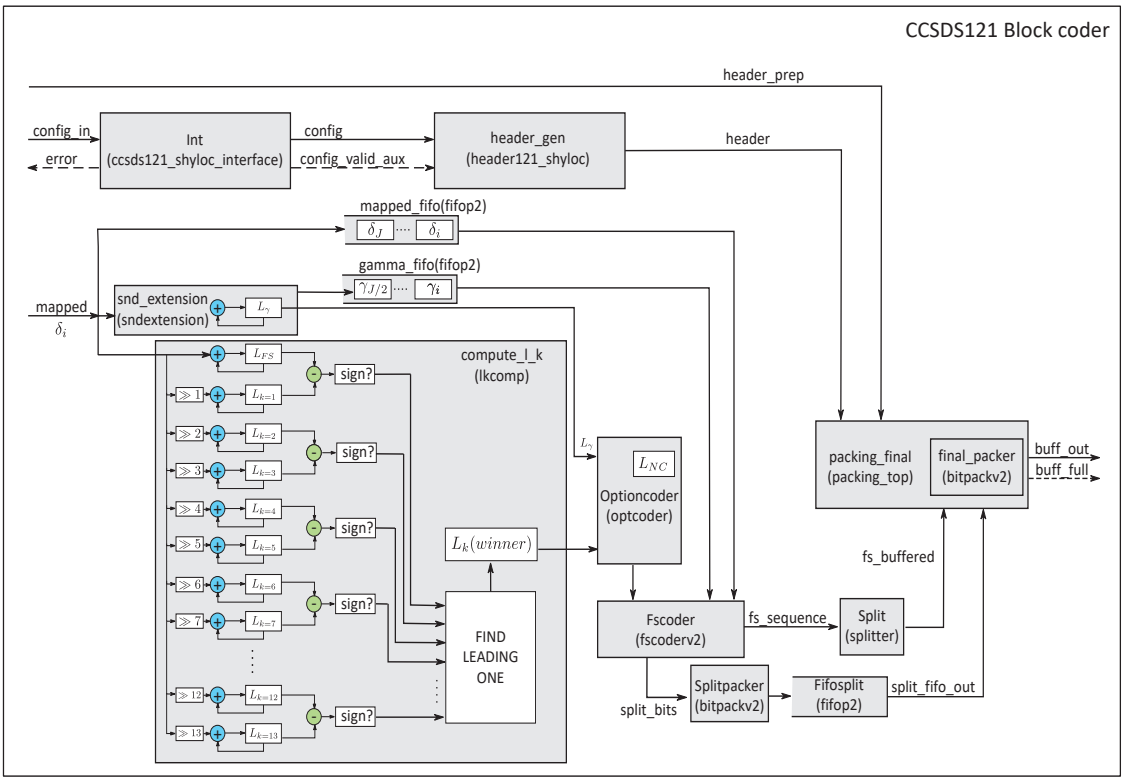


FIGURE 4.1: Block diagram of the CCSDS 121.0-B-3 block-adaptive entropy coder

features. If this encoder is part of a full compression chain, a dedicated input is employed to receive header fields from an external preprocessor connected to the encoder. Once the header has been properly generated, it is transmitted 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, which should have a value between 8 and 64, in steps of 8 (i.e., a byte boundary). Every time the output buffer is full with *W_BUFFER* bits, a *Valid* flag indicates that the output value can be captured by the external receiver.

The rest of the modules constitute the encoding engine itself, implementing the encoding options described in Section 4.2.1. The input mapped residuals δ_i are stored in the *mapped* FIFO until a block of J samples is completed. The size of the *mapped* and the rest of the FIFOs in the design is optimized according to the user-defined values specified for J and D. These FIFOs are mapped into BRAMs when $J > 32$, enabling in this case the EDAC mechanism by setting its associated generic to 1, which is able to correct single errors and detect and inform about Multiple Bit Upsets (MBUs).

The *snd_extension* module computes the length of a block encoded with the second-extension option, storing its output in the *gamma* FIFO, while the *compute_l_k* module calculates the length of a block encoded with the FS, as well as all the sample splitting options, selecting the value of k that yields the shortest encoded block by subtracting the consecutive options $L_k - L_{k+1}$, until the first negative result is found. The number of options to be evaluated depends on the dynamic range of the input samples D and other user-selected configuration parameters. In addition, the *compute_l_k* module identifies if a block of samples 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 denoted as L_k*(winner)*, and it is compared with the lengths obtained by the *snd_extension* and *no_compression* modules in the *optioncoder* module. To compute the length of a block of samples for all encoding options, it is necessary to sum and accumulate the number of bits taken by every mapped prediction residual δ_i in a block, for each one of the available encoding options. The selector will then choose the encoding option which produces the output with the minimum number of bits, taking also into account the bits used by the option identifier, which is generated according [27]. It has to be remarked that the maximum length of a compressed block of samples will be that of applying the *no compression* option ($J \cdot D$), so the size of the register that saves the number of bits for each option, can be established as $log_2(J \cdot D)$.

Once the suitable encoding option is selected, the *fscoder* module encodes the stream according to the FS sequence, joined to the option identifier. When encoding the sample-split option, the split bits are packed separately in buffers and then attached to the FS codeword, using a FIFO to temporarily store the split bits. The sequence encoded by the *fscoder*, as well as the uncompressed sample splits, are sent to the final packer to be outputted.

The entropy coder must also manage periodic reference samples introduced by the unit-delay predictor, in case it is connected to the encoder input. It should be able to identify which blocks of samples include a reference sample. This process is done by introducing a dedicated state in the block-coder FSM. The reference sample is coded after the unique identifier for each coding option in all the cases, including the *no compression* option. The *snd_extension* and the *compute_l_k* modules should 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 D, 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.

4.2.3 Block characterization

Different sets of configuration parameters have been defined, with the purpose of analysing the impact of the block-encoder parameters in terms of logic resources utilization and maximum clock frequency. These synthesis sets are summarised in Table 4.1. The main parameters that change among configurations are the block size *J_GEN*, which defines

TABLE 4.1: Sets of synthesis configurations for the CCSDS 121.0-B-3 block-adaptive encoder

Generic	Set1	Set2	Set3	Set4
EN_RUNCFG	0	0	0	1
RESET_TYPE	1	1	1	1
Nx_GEN	1024	1024	1024	1024
Ny_GEN	1024	1024	1024	1024
Nz_GEN	1024	1024	1024	1024
D_GEN	16	16	16	16
J_GEN	16	32	64	16
W_BUFFER_GEN	32	32	64	64

the number of samples that comprise an input block; and the width of the output buffer, *W_BUFFER_GEN*. As it is also reflected in Table 4.1, the maximum image size is fixed to 1024 for each one of the three coordinates and the dynamic range D to 16, since it is the maximum allowed by the block-adaptive encoder. Since $D = 16$, the basic codeset is always selected.

The maximum clock frequency for each configuration set is reflected in Table 4.2 for Xilinx Kintex UltraScale XCKU040. The hardware occupancy in terms of memory blocks, arithmetical units and logic resources is also provided in that table for each one of the proposed configuration sets. Since the block-adaptive encoder processes one sample per clock cycle, the maximum throughput (measured in MSamples/s) will be equal to the maximum clock frequency achieved by each synthesis set.

The best result in terms of maximum clock frequency achieves a maximum value of 188.3 MHz when using the simplest configuration (i.e., Set1), with $J = 16$ and $W\_BUFFER = 32$. In general, it is observed that high values of J and W_BUFFER have a slight negative impact in maximum clock frequency. This negative impact is noted when comparing synthesis results between Set1 and Set3, being the latter the most restrictive one, attending to J and W_BUFFER values (both equal to 64, the maximum allowed). Using Set3 implies a maximum clock frequency reduction around the 14% regarding to Set1.

In terms of logic resources utilization, 1.8% of the available LUTs are utilized on Kintex UltraScale when using the most restrictive configuration (i.e., Set3). The BRAM usage is around the 0.6% of the total available employing also Set3. For the rest of hardware resources, it is observed that the trend is to increase for high values of J and W_BUFFER parameters.

Taking into account these results, it is concluded that the proposed implementation based on the CCSDS 121.0-B-3 block-adaptive encoder allows on-board data processing at high rates, with low area footprint.

TABLE 4.2: CCSDS 121.0-B-3 block-adaptive encoder - Synthesis on Xilinx Kintex UltraScale XCKU040

Parameters	**Total**	**Set1**	**Set2**	**Set3**	**Set4**
Block RAMs	600	0	3	4	0
DSP48	1920	2	2	4	2
Registers	484800	1130	1365	1576	1560
LUTs	242400	3428	3428	4317	3708
Maximum Frequency (Clk_S) (MHz)		188.3	173.7	160.2	161.7

4.3 CCSDS 123.0-B-1 sample-adaptive encoder

4.3.1 Algorithm overview

The sample-adaptive encoder is a more sophisticated but less computationally complex version of an adaptable Rice coder than the block-adaptive one. However, the sample-adaptive encoder presents a theoretical limitation in terms of CR that prevents from achieving less than 1 bpp. This constraint is not present in neither the block-adaptive nor the hybrid encoders, which efficiently exploits low-entropy data to enhance the CR.

With the sample-adaptive encoder, the input residuals $\delta(t)$ are compressed individually instead of in blocks, by using a family of Golomb-power-of-2 (GPO2) codes [30]. Each code is identified by an index k, which is selected for each particular sample depends on the image statistics (i.e., information of previously processed samples). The coding process is performed as described next when $t > 0$:

1. If $\lfloor \delta(t)/2^k \rfloor < U_{max}$, the codeword is formed by $\delta(t)/2^k$ zeros, followed by a one and the k least significant bits of $\delta(t)$. The unary length limit U_{max} is a user-defined parameter in the range $8 \leq U_{max} \leq 32$.

2. Otherwise, the codeword consists of U_{max} zeros, followed by the representation of $\delta(t)$ with D bits.

The first mapped prediction residual in each band $\delta_z(0)$ is uncompressed and attached to the output bitstream represented with D bits. In addition, fill bits are appended at the end of the compressed image just after the last codeword, until the next output word boundary is reached.

Image statistics are maintained by two variables, whose values are used to choose the corresponding $k_z(t)$ for each input sample: a counter $\Gamma(t)$ and an accumulator $\Sigma_z(t)$, which are computed independently for each image band. In this way, the index $k_z(t)$ is the largest integer, lower or equal than $D-2$, that satisfies Equation 4.1. The ratio between both statistics provide an estimation of the mean $\delta(t)$ for the band z under process.

$$\Gamma(t)2^{k_z(t)} \leq \Sigma_z(t) + \lfloor \frac{49}{2^7}\Gamma(t) \rfloor \tag{4.1}$$

These statistics are updated with every new sample, depending on the chosen compressor configuration. Besides, both the counter and the accumulator are periodically rescaled applying a division by two, avoiding infinite values and, consequently, overflow in the used data-types. The interval at which statistics are rescaled is controlled by the rescaling counter size $\gamma*$, which shall be in the range $max\{4, \gamma_0 + 1\} \leq \gamma* \leq 11$, being γ_0 the initial count exponent, another user-defined parameter, which shall be in the range $1 \leq \gamma_0 \leq 8$. The initial value of the counter $\Gamma(1)$ is 2^{γ_0}, while the initial accumulator value $\Sigma_z(1)$ is calculated according to Equation 4.2, where k'_z is a value in the range $0 \leq k'_z \leq D-2$.

$$\Sigma_z(1) = \lfloor \frac{1}{2^7}(3 \cdot 2^{k'_z+6} - 49)\Gamma(1) \rfloor \tag{4.2}$$

4.3.2 Block design

The sample-adaptive encoder is formed by two main modules, the compression engine and the bitpacker, together with a FSM that manages the complete encoding process, as shown in Figure 4.2. In addition, it includes the necessary logic to receive the configuration at runtime through an AMBA AHB slave interface and the input samples through a dedicate ad-hoc interface. Input and output data interfaces are implemented using a handshaking protocol, associating a *Valid* signal to each input/output data. The received configuration is checked to confirm that it is correct, raising an error otherwise. These values are sent to the rest of the modules in the design.

As soon as mapped residuals $\delta(t)$ are received, the compression engine calculates the code k for each input sample using the *createcdw* module, taking into account the current counter and accumulator values. The *opcode_update* module generates an operation code depending on the location of the sample in the image, developing different variations of this functional unit for each possible compression order (BIP, BSQ and BIL).

Then, these image statistics (i.e., the counter and the accumulator) 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 values of a specific sample with all its spectral components (i.e., in all the bands), necessary to compress the next one. All this processing chain has been developed as a highly-optimized pipeline, which allows to encode an input sample every clock cycle.

The last part of the encoding flow is the *bit_pack* module, which wraps the generated codewords according to the value of the *W_BUFFER_GEN* constant. Each time the output buffer is full, a *Valid* flag is asserted to indicate that the output bitstream can be captured.

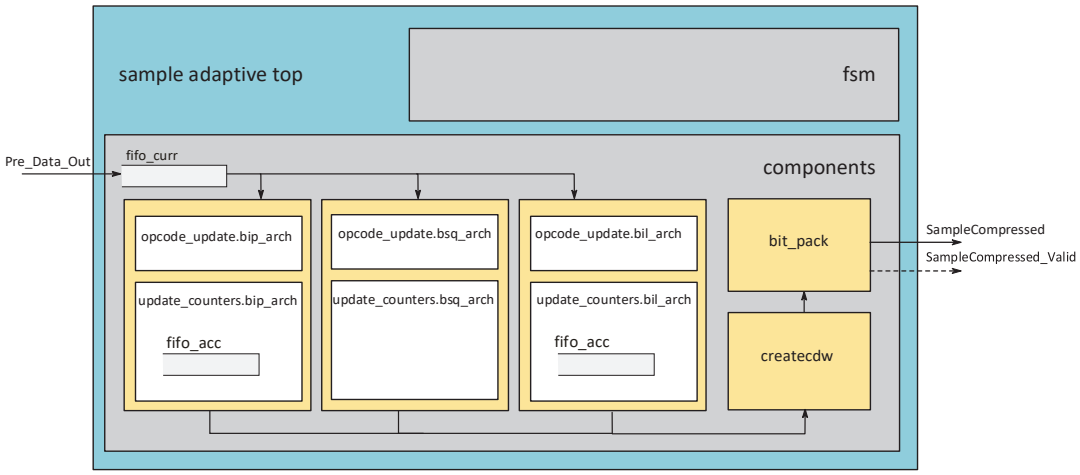


FIGURE 4.2: Block diagram of the CCSDS 123.0-B-1 sample-adaptive entropy coder

4.3.3 Block characterization

Table 4.3 summarises the sets of configuration values defined to synthesize the CCSDS 123.0-B-1 sample-adaptive encoder, with the purpose of analyzing parameters impact in both hardware occupancy and maximum clock frequency. Image features, such as size and input bit-depth D, are fixed to target AVIRIS scenes. Although the sample-adaptive

Table 4.3: Sets of synthesis configurations for the CCSDS 123.0-B-1 sample-adaptive encoder

Generic	Set1	Set2	Set3	Set4
EN_RUNCFG	0	0	1	1
Nx_GEN	512	512	512	512
Ny_GEN	680	680	680	680
Nz_GEN	224	224	224	224
D_GEN	16	16	16	16
W_BUFFER_GEN	32	32	64	64
INIT_COUNT_E_GEN	1	3	1	3
ACC_INIT_CONST_GEN	4	5	4	5
RESC_COUNT_SIZE_GEN	5	6	5	6
U_MAX_GEN	8	16	32	32

encoder is developed as a standalone solution some features, such as the statistics update, are dependent on the input image arrangement; in this way, all the proposed configuration sets are defined for BIP order, since it makes possible that in a full lossless compression chain, as the one presented next in Section 5.4, a whole throughput of one input sample per clock cycle can be reached. Parameters that change among configurations are the the width of the output buffer, *W_BUFFER_GEN*, and the ones specifically related the encoding process (i.e., the unary length limit U_{max}, the rescaling counter size $\gamma*$, the initial count exponent γ_0 and the accumulator initialization constant k).

Results in terms of maximum clock frequency are shown in Table 4.2 for Xilinx Kintex UltraScale XCKU040. In the same way, hardware occupancy is provided in terms of memory blocks, arithmetical units and logic resources for each one of the proposed parameter combinations. Since it is able to process one input sample (i.e., a mapped prediction residual $\delta(t)$) per clock cycle, the sample-adaptive encoder always achieves a throughput equal to the maximum clock frequency magnitude independently of the selected configuration. In terms of maximum clock frequency, a value up to 175.7 MHz is reached for Kintex UltraScale XCKU040 when Set4 is used for synthesis, though minimum differences (around the 10%) are obtained regarding the worst case, under Set3. It is observed that high values of *W_BUFFER* and U_{max} have not an influence in the maximum clock frequency achieved, while the relationship between γ_0 and $\gamma*$ implies a penalty.

In general terms, the sample-adaptive encoder allows real-time data processing while at the same time has a lower area footprint than the block-adaptive one, since the latter simultaneously computes multiple encoding options and works with groups of J samples.

In contrast, the sample-adaptive encoder just calculates a single encoding option for each input residual independently.

Although Set4 is the fastest in terms of maximum clock frequency for Kintex UltraScale XCKU040, it is also the most demanding one regarding hardware occupancy. Anyway, this area footprint is minimum, consuming just the 0.1% of registers and the 0.4% of the available LUTs in the device. DSPs are not used independently of the selected configuration set, while BRAM usage is negligible. This memory consumption is just conditioned by the way in which the statistics are stored under BIP order (i.e., an accumulator value per band z).

TABLE 4.4: CCSDS 123.0-B-1 sample-adaptive encoder - Synthesis on Xilinx Kintex UltraScale XCKU040

Parameters	**Total**	**Set1**	**Set2**	**Set3**	**Set4**
Block RAMs	600	1	1	1	1
DSP48	1920	0	0	0	0
Registers	484800	434	434	500	500
LUTs	242400	942	961	1007	1033
Maximum Frequency (Clk_S) (MHz)		167.2	167.1	159.6	175.7

4.4 CCSDS 123.0-B-2 hybrid encoder

4.4.1 Algorithm overview

With the aim of obtaining the lowest possible bit rate at the ouput of the encoding stage, the hybrid encoder follows an interleaved strategy, switching between two possible encoding methods, denoted as *high-entropy* and *low-entropy*. A single output codeword is generated by each processed sample under the high-entropy mode, while the low-entropy method, which prevails when losses are introduced in the prediction stage (i.e., near-lossless or lossy compression) or when low-entropy input data is received, may encode multiple samples in a single codeword, thus achieving lower compression rates than the other entropy coder alternatives previously described in this chapter.

The selected mode depends on code selection statistics (i.e., a counter and an accumulator), which are updated with every new sample. Values of already processed samples are added to the accumulator, while the counter registers the number of samples that have been

previously processed. As in the case of the sample-adaptive encoder, both statistics are rescaled at certain points by dividing by two, preserving the average estimated value. The least significant bit of the accumulator is outputted before rescaling, allowing to the decoder to invert the statistics update process. More details about how these statistics are incremented and rescaled can be found in [31], including the theoretical basis.

The high-entropy method is selected if the condition described in Equation 4.3 is met, where $\Sigma_z(t)$ and $\Gamma(t)$ are the accumulator and counter values in the target band z, respectively, and T_0 represents a threshold that determines if the sample under analysis should be encoded using the high-entropy process or the low-entropy one, and whose value is reflected in Table 4.5.

$$\Sigma_z(t) \cdot 2^{14} \geq T_0 \cdot \Gamma(t) \tag{4.3}$$

Under the high-entropy mode, the hybrid encoder works in a similar way than the sample-adaptive one described in Section 4.3.1, but coding the codewords in reverse order. In this way, GPO2 codes are used to code input samples individually depending on image statistics, which are independently computed for each separate band. In this case, the coding procedure is based in the next assumptions, depending on the relationship between input residuals $\delta(t)$ and the k index value:

1. If $\delta(t)/2^k < U_{max}$, the codeword is comprised by the k least significant bits of $\delta(t)$, followed by a one and $\delta(t)/2^k$ zeros.

2. Otherwise, the high-entropy codeword consists on the representation of $\delta(t)$ with D bits, followed by U_{max} zeros.

On the other side, the low-entropy mode uses one of the 16 variable-to-variable length codes to code each mapped residual $\delta(t)$. During encoding, each low-entropy code has an active prefix, which is a sequence of input symbols. The selection of the appropriate active prefix is based on Equation 4.4, where T_i represents the threshold of the largest code index i that can be used to code the low-entropy residual. These threshold values are defined in the standard, as reflected in Table 4.5.

$$\Sigma_z(t) \cdot 2^{14} < \Gamma(t) \cdot T_i \tag{4.4}$$

TABLE 4.5: Relationship between the code index, the input symbol limit and the threshold in the low-entropy mode

Code index i	Input Symbol Limit L_i	Threshold T_i
0	12	303336
1	10	225404
2	8	166979
3	6	128672
4	6	95597
5	4	69670
6	4	50678
7	4	34898
8	2	23331
9	2	14935
10	2	9282
11	2	5510
12	2	3195
13	2	1928
14	2	1112
15	0	408

The input symbol determines, following Equation 4.5, if the current mapped residual $\delta_z(t)$ is appended to the selected active prefix; on contrary, if the input symbol is the escape one (represented as X), the residual value $\delta_z(t) - L_i - 1$ is directly outputted to the bitstream and coded in the same way than it is done by the high-entropy method, being L_i the input symbol limit for that specific code, which is also predefined in the standard [31]. The index i is an integer in the range $0 \leq i \leq 15$, equivalent to the total number of variable-to-variable length codes available in the low-entropy mode. If after updating the active prefix is equal to a complete input codeword, as specified in the code table associated to that code, the corresponding output codeword assigned to that input word is appended to the output bitstream, and then the active prefix for that low-entropy code is reset.

$$i_z(t) = \begin{cases} \delta(t), \delta(t) \leq L_i \\ X, \delta(t) > L_i \end{cases} \tag{4.5}$$

Finally, a compressed image tail is appended at the end of the output bitstream, after all the mapped residuals have been encoded. This field contains the necessary information to decode the compressed bitstream in reverse order. Reverse decoding process becomes necessary because of the latency between the processing of a low-entropy mapped residual

and the output of the codeword that encodes it. In this way, memory-efficient hardware implementations can be efficiently developed. The image tail is comprised by the flush codeword, in increasing order, for each one of the 16 the active prefixes, in addition to the final value of the accumulator $\Sigma_z(t)$ in each spectral band. Then, the bitstream is filled with padding bits until a byte boundary is completed, adding a '1' bit followed by as many zero bits as necessary. This '1' bit is used by the decoder to identify the number of padding bits that shall be discarded before starting the decompression process. The compressed bitstream is completed with a header that specifies the user-defined parameters employed by the encoder, which must be known for a correct decompression.

4.4.2 Block design

The top-level block diagram of the hybrid encoder is shown in Figure 4.3, which is comprised by four functional units: the *Statistics* module, the responsible of initializing, updating and rescaling both the counter and the accumulator; the *Method Selection* unit that implements Equation 4.3 and consequently decides which encoding method is selected, depending of the current $\delta(t)$ value; the *Bitstream Generator*, which solves the bottleneck of outputting the resultant codeword generated by the hybrid encoder in a bit-by-bit manner (being possible to complete more than one byte in the same iteration or even let a byte incomplete), thus adapting data rates; and the two main modules that compute the high- and low-entropy methods, as described below.

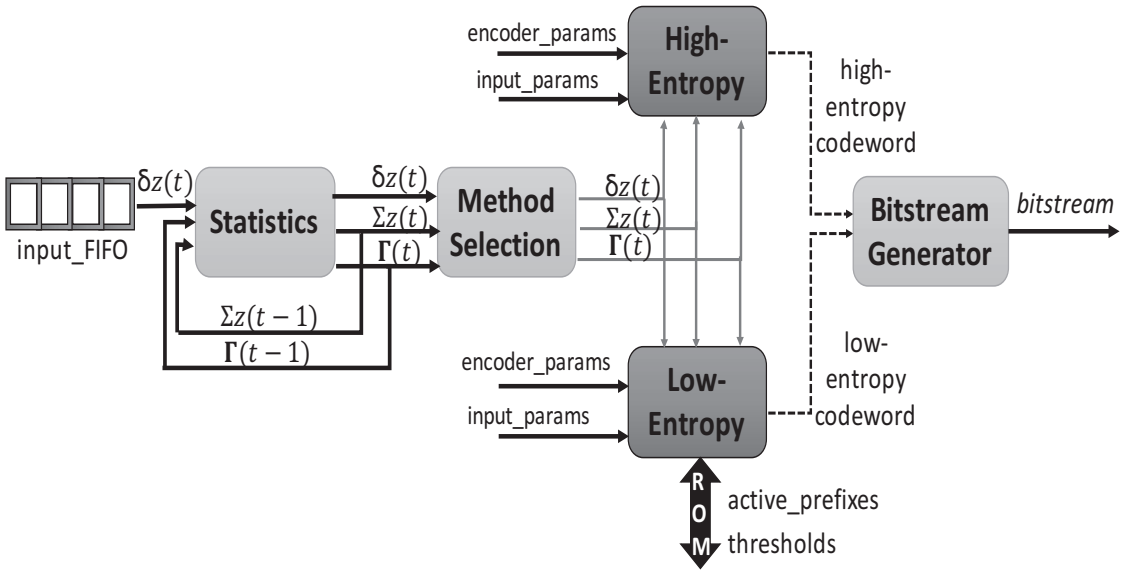


FIGURE 4.3: General overview of the hybrid encoder architecture

The hybrid encoder receives both the configuration and the input samples through AXI interfaces. This module receives the encoder parameters, the image size and the pixel resolution D through an AXI4-Lite interface. Then, the input samples are received from an intermediate FIFO, where the input mapped residuals $\delta(t)$ are stored. The output interface is based on a basic AXI4-Stream interface that uses a simple handshaking protocol by defining the *Valid* and *Ready* signals for dataflow control, in addition to the *Last* signal. which indicates the transmission of the last codeword. Input and output data interfaces can be also synthesized with an ad-hoc protocol, if it is required to be connected to other modules, such as a preprocessor, to conform more sophisticated compression solutions.

The variable-length codes (a single codeword for each high-entropy code or a codeword comprised by multiple low-entropy codes) are sent grouped in words of *W_BUFFER* bytes, parameter defined by the user. An extra flag, named End-of-Processing (EOP), is used to inform to the external bitpacker that the encoder has finished the bitstream generation. In addition, both the final state of the 16 low-entropy codes and the accumulator values for each band z are saved in FIFOs, which are accessed by the external bitpacker to properly generate the image tail.

4.4.2.1 High-entropy unit

This functional unit is responsible of encoding the mapped residual $\delta(t)$ using a high-entropy codeword, when Equation 4.3 is satisfied. This process employs RLL-GPO2 codes and the procedure is done based on the assumptions described in Section 4.4.1.

The high-entropy module is divided into two main parts: the computation of the k value and the writing of the resultant codeword in the output bitstream. The calculation of k has been done in an iterative way, trying to find the largest positive value of $k_z(t) \leq max\{D-2, 2\}$ that satisfies Equation 4.6. That loop is accelerated by following a pipelining strategy and avoiding the use of complex arithmetic operations, replacing divisions, which are not fully optimized by HLS tools, by logic shifts.

$$\Gamma(t)2^{k_z(t)+2} \leq \tilde{\Sigma_z}(t) + \lfloor \frac{49}{2^5}\Gamma(t) \rfloor \tag{4.6}$$

4.4.2.2 Low-entropy unit

This functional unit encodes the mapped residual $\delta(t)$ when Equation 4.3 is not satisfied. For doing so, mapped residuals are set into one of the 16 group of values identified by a code index, i. The management of the input-to-output and flush input-to-output codeword tables in hardware result very inefficient from a computational point of view, if the process is implemented as directly described in the standard. To overcome this issue, the information present in these tables as well as the searching patterns have been reorganized. For doing so, the information present in the input-to-output codeword table for each code index, i, and its corresponding flush input-to-output codeword table have been merged into two arrays, named APl_i and APv_i, respectively. Additionally, the corresponding AP_i is implemented as an index used to move inside APl_i and APv_i arrays (i.e., as an offset) and hence, it is initialized to 0 instead to a null sequence. For each $\delta(t)$, in this new approach, $i_z(t)$ is calculated as shown in Equation 4.7. As described in the standard, if $\delta(t)$ exceeds L_i, the residual value $\delta(t) - L_i - 1$ is directly encoded to the bitstream as it is done in the high-entropy mode, with a k value equal to 0.

$$i_z(t) = min(\delta(t), L_i + 1) \tag{4.7}$$

After calculating $i_z(t)$, AP_i is updated as shown in equation 4.8:

$$\mathrm{AP}_i = \mathrm{AP}_i + i_z(t) + 1 \tag{4.8}$$

Then, i and v are obtained as $\mathrm{APl}_i(\mathrm{AP}_i)$ and $\mathrm{APv}_i(\mathrm{AP}_i)$, respectively. Two actions can be carried out depending on the i value:

1. If i value is not 0, v value is written to the bitstream using i bits and AP_i is reset to 0.
2. If i value is equal to 0, AP_i is updated to v value ($\mathrm{AP}_i = v$).

After processing all the $\delta(t)$ values, the remaining active prefixes are added to the bitstream by directly coding the current $\mathrm{APv}_i(\mathrm{AP}_i)$ values using $\mathrm{APl}_i(\mathrm{AP}_i)$ bits. Following this simple strategy, the number of memory accesses and operations needed to process and update the active prefixes, and to obtain the output codewords according to the input ones is strongly optimized for a hardware implementation.

4.4.3 Block characterization

The CCSDS 123.0-B-2 hybrid encoder is developed using Xilinx Vitis HLS tool and mapped on Kintex UltraScale XCKU040. Unlike the CCSDS 123.0-B-2 predictor, where hardware occupancy depends on the user-defined parameter values, as demonstrated in Section 3.4.3, the encoder parameters do not have a significant impact on logic resources utilization. For this reason, the selected combination of encoder parameter values reflected in Table 4.6 is the one that achieves best results in terms of encoding performance. The baseline configuration is conditioned by AVIRIS, which is the target sensor, including image size and dynamic range D. The hybrid encoder is synthesized with a maximum clock frequency of 154 MHz.

Table 4.6: Baseline synthesis configuration for the CCSDS 123.0-B-2 hybrid encoder

Parameter	**Value**
Image parameters	
Columns, N_x	677
Lines, N_y	512
Bands, N_z	224
Dynamic Range, D	16
Encoding Order	BIL
Encoder parameters	
Unary Length Limit, U_{max}	16
Rescaling Counter Size, $\gamma*$	5
Initial Count Exponent, γ_0	1
Output Buffer Width, W_BUFFER	32

The only key parameter that can affect to the memory usage is the number of bands of the input hyperspectral image, N_z. This value determines the number of elements in the counter and accumulator vectors, since statistics are maintained per band when working in BIL order. However, as it is illustrated with the results of Table 4.7, a change in the N_z value for the situations under study does not modify BRAM consumption, since the size of each memory block in this technology is 36Kb and it is enough to store statistics independently of the number of bands considered for the input image. In total, just the 1.9% of BRAMs available in the device are used. The rest of BRAMs used by the hybrid encoder are associated to the storage of low-entropy values, such as input-to-output codeword and flush tables or the threshold values.

The consumption of DSPs supposes around the 0.2% of the total available in the device. These resources are mainly used to perform multiplications on Equations 4.3 and 4.4, calculated under the low-entropy mode. Finally, the LUTs usage is also reduced, consuming

an average of the 2.1% of the total for the different N_z values reflected in Table 4.7. In any case, the hardware occupancy of the hybrid encoder is considered negligible compared to other stages present in a full compression chain.

Table 4.7: CCSDS 123.0-B-2 hybrid encoder - Synthesis on Xilinx Kintex UltraScale XCKU040

Parameters	**Total**	**Baseline**	$N_z = 32$	$N_z = 128$	$N_z = 512$
Block RAMs	600	11.5	11.5	11.5	11.5
DSP48	1920	3	3	3	3
Registers	484800	2361	2352	2358	2364
LUTs	242400	4980	4966	4979	4981

4.5 Conclusions

Three entropy coding approaches have been presented throughout this Chapter, which are compliant with the CCSDS standards for on-board data compression. The block-adaptive encoder, originally defined in the CCSDS 121 universal lossless compression standard, is based on Rice coding. The sample-adaptive and the hybrid encoder alternatives are presented in the CCSDS 123 compression standard for multi- and hyperspectral images. Although both are based on Golomb codes, the hybrid encoder allows higher CRs by exploiting low-entropy data.

It has been demonstrated that the developed functional blocks can be successfully mapped on a Xilinx Kintex UltraScale XCKU040 FPGA, providing a high throughput to achieve real-time processing, together with a low area footprint. Results in other space-grade FPGA technologies (e.g. NanoXplore Brave NG FPGAs or Microsemi RTG4) have been obtained but not included so as not to expand the document unnecessarily.

Chapter 5

Modular solutions for on-board data compression

This Chapter presents the design and implementation of IPs for the compression of different kind of data collected and generated on-board satellites. These IPs are mainly based on functional blocks developed and presented in Chapters 3 and 4. Different solutions are proposed taking into account space mission constraints, such as data nature, expected Rate-Distortion ratio, area footprint or real-time processing demands. The solutions provided are compliant with the CCSDS compression standards, guaranteeing compatibility with decompression on ground. For each proposed compression chain, its structure is detailed and results are provided in several space-grade FPGAs to demonstrate the viability to be used in a EO mission. A performance analysis is also presented to demonstrate the viability of using an adapted version of the CCSDS 123.0-B-2 compression algorithm to process both panchromatic and RGB video in future space missions.

5.1 Outline

As mentioned throughout this Thesis, it is expected that future EO and exploration space missions integrate high-resolution sensors. The development of compression chains capable of handling and processing the high volume of data collected by these sensors is currently a challenge for the space industry, since there is a change of paradigm in terms of on-board computational demands and how hardware resources influence on both the overall payload performance and the global power consumption.

For this reason, different compression IPs have been developed in this Thesis for space missions. These IPs are comprised by two main functional blocks, a prediction-based preprocessor and an entropy coding stage. By combining the developed functional blocks, as depicted in Figure 5.1, it is possible to implement a suitable compression solution for a specific application or space mission. There is also the possibility of implementing a versatile compression solution capable of processing data acquired by multiple sensors (i.e., with different nature) just with a single compression core, reducing hardware occupancy and power consumption.

The selection of the suitable functional blocks is conditioned not only by the nature of the input data, but also by the space missions constraints, including the performance objectives in terms of throughput, hardware occupancy or power consumption. The combination of RTL and HLS design methodologies is also considered when a compromise between performance and development time is required, taking advantage of the strengths of each methodology to achieve design goals.

An additional advantage of the proposed modular approach is the reusability. This means that the independent functional blocks that have been developed for each compression stage (i.e., preprocessing and entropy coding) can be reused in future space missions for on-board compression. Proper behaviour is ensured, since these modules have been deeply characterized, at the same time that the space program can soften the scheduling thanks to avoiding a development from scratch.

Because the compression chains designed are compliant with CCSDS standards, a proper decompression and data management is ensured on ground. In addition, it is possible to obtain versatile solutions that can compress data from different nature with a single processing core by selecting the appropriate modules. In this way, ad-hoc compression

approaches are proposed depending on mission necessities at the same time that area utilization is optimized.

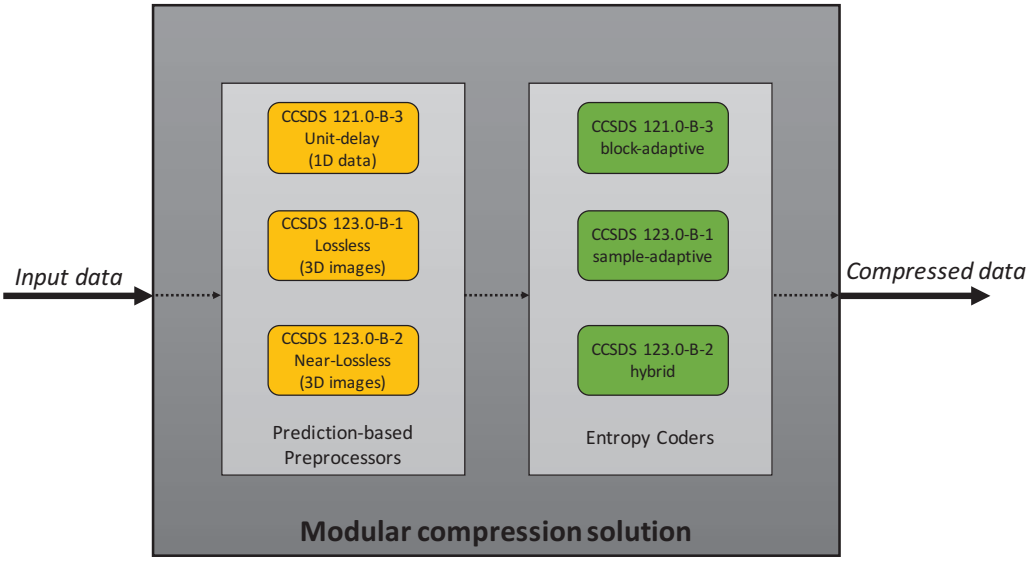


FIGURE 5.1: Concept of modular compression solution

5.2 Validation scenarios

To validate this modular compression approach some scenarios are defined, in which different compression solutions are proposed in the form of IP cores. The appropriate combination of functional blocks will be selected taking into account the specific constraints of each scenario. As it will be explained throughout this Chapter, all the developed functional blocks presented in Chapters 3 and 4, and reflected in Figure 5.1, have been employed to design the proposed compression chains. It is important to mention that the selection of these scenarios takes into account two main objectives: to be as completed as possible; and to be representative of real space missions.

The specific scenarios that have been considered to demonstrate the goodness of the proposed modular compression solution are listed below:

- **Lossless one-dimensional data and image compression**. This approach allows to compress one-dimensional data and 2D images by combining the unit-delay predictor with the block-adaptive encoder. As an example, this solution is used in the Lagrange ESA-funded program [183] and the SUNRISE III mission [184], This

results in a compression solution fully compliant with the CCSDS 121.0-B-3 lossless standard.

- **Lossless hyperspectral image compression**. This solution includes a 3D lossless predictor defined in the CCSDS 123.0-B-1 compression standard combined with any of the entropy coders developed in Chapter 4. As an example of this scenario the SHyLoC IP cores can be mentioned [58, 60], though SHyLoC does not use the hybrid encoder, since it is not a fully compliant solution.

- **Near-lossless hyperspectral image compression**. This scenario goes a step beyond of the lossless HSI compression approach, providing higher compression ratios by replacing the 3D lossless predictor by the one proposed in the CCSDS 123.0-B-2 compression standard. The introduction of losses can be controlled through certain user-defined parameters. This predictor can be combined with any of the entropy coders presented in Chapter 4, all of them compliant with the CCSDS 123.0-B-2 standard. A tailored version of this development will be integrated as part of the CHIME instrument processing chain, selecting the appropriate options and parameters to maximize the CR and without compromising both the throughput and the resources utilization.

- **Video compression**. A final scenario will be dedicated to the compression of video sequences on-board satellites. The proposed solution is based on the CCSDS 123.0-B-2 near lossless compression standard. An example for this scenario is the video sensor to be embarked on the instrument developed in the scope of the H2020 VIDEO project [63]. The solution provided will be able to compress both monochrome and RGB video sequences.

5.3 Lossless one-dimensional data and image compression

As a one-dimensional data compressor, a hardware implementation mainly comprised by the unit-delay predictor described in Section 3.2.2 and one of the entropy coders explained in Chapter 4 is proposed. Therefore, three alternative solutions arrive by combining the unit-delay predictor with the block-adaptive, sample-adaptive or the hybrid encoders. The top-level structure of the proposed IP core is reflected in Figure 5.2.

The combination of the unit-delay predictor with the block-adaptive encoder will be fully compliant with the CCSDS 121.0-B-3 standard. As an application example, this solution is considered for its integration in the Lagrange mission [183] and also in the SUNRISE III mission [184], in which a balloon-borne solar observatory is expected to be launched at the stratosphere. These use cases emerge in the context of a collaboration among our Group, the Instituto de Astrofísica de Canarias (IAC) and the Instituto de Astrofísica de Andalucía (IAA) [1].

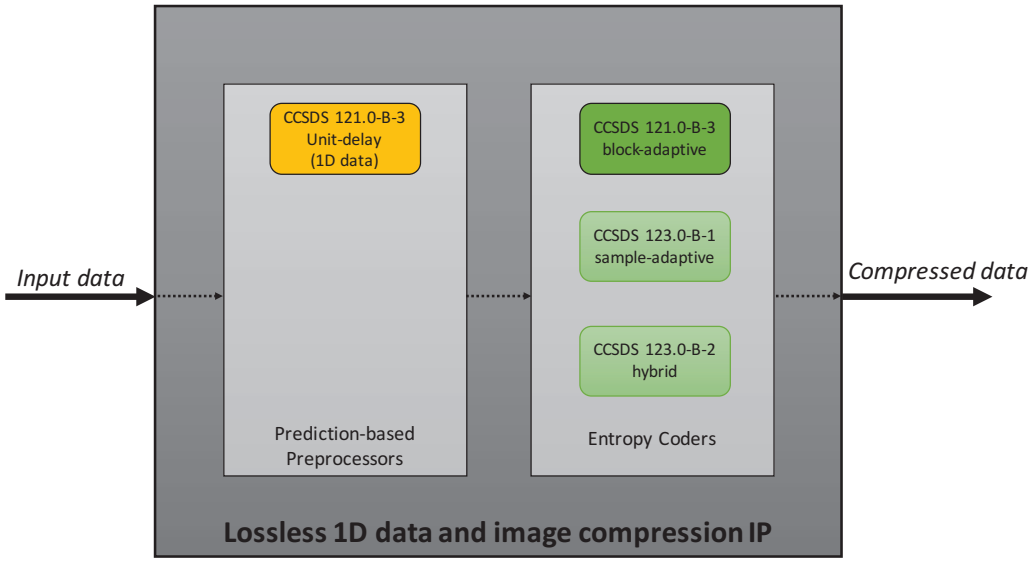


FIGURE 5.2: Block diagram of the lossless 1D data and image compression IP

5.3.1 System development

The overall diagram of the IP for this scenario is depicted in Figure 5.3. The IP includes, in addition to the unit-delay predictor and the block-adaptive entropy coder, a configuration engine to enable the setting of parameters allowed by the standard, both at compile-time or at runtime. This configuration engine controls that the logic internally dedicated by each module to receive and validate their configuration parameters works properly. The configuration is controlled by the *EN_RUNCFG* parameter; in case of it is disabled, the parameters are set at compile-time, reducing the overall complexity of the design. This solution supports signed samples and an input bit-depth D up to 32 bits per sample [60].

Regarding input and output data interfaces, the IP defines a wrapper based on a handshaking protocol, associating a *Valid* signal to each input/output data. Input samples are

[1] Characterization results for these missions are not provided due to confidential reasons

received through an ad-hoc parallel interface, while the compressed bitstream is produced at the output by using a similar interface, easing the connection with external modules in a plug&play fashion. Data flow is handled through the *ForceStop* (it forces the stop of a compression at any point) and the *Ready* signals at the input side, being the latter asserted when the IP is configured correctly and thus it is able to receive new samples to be compressed. The *Ready_Ext* input signal is used to inform that the external module responsible of managing the output data is ready to receive the compressed samples. Additionally, there are other signals used for data control, such as *FIFO_full*, *EOP* (it indicates that the compression of the last sample has started) or *Finished*, which is asserted when the compression has ended. The input signal *IsHeaderIn* is also defined to inform that the received data is part of a header, easing the substitution of the unit-delay predictor by an alternative preprocessing stage, if necessary to achieve specific performance goals. Finally, the *Error* signal is asserted in case of malfunction or an unexpected situation during the compressor performance.

At the beginning of the workflow, the IP receives the runtime configuration values through an AMBA AHB slave interface only if $EN\_RUNCFG = 1$; otherwise, compile-time configuration is used. After that, the *Ready* signal is asserted to inform that the IP is waiting for new input samples. The received configuration is stored in internal registers, read by the IP and then made available for the functional blocks that conform it. Then, the input samples are preprocessed by the unit-delay predictor prior to being coded.

The IP core can optionally create a header; in that case, the generated codewords are appended to the header and finally packed. Regarding mitigation techniques, the IP core includes EDAC for critical embedded memories located in the block-adaptive encoder, with capability to correct single errors and to detect double errors. EDAC is enabled by setting the corresponding generic parameter. Besides, states in the FSM are encoded using Gray coding and including safe statements, returning to the *idle* state in case of errors.

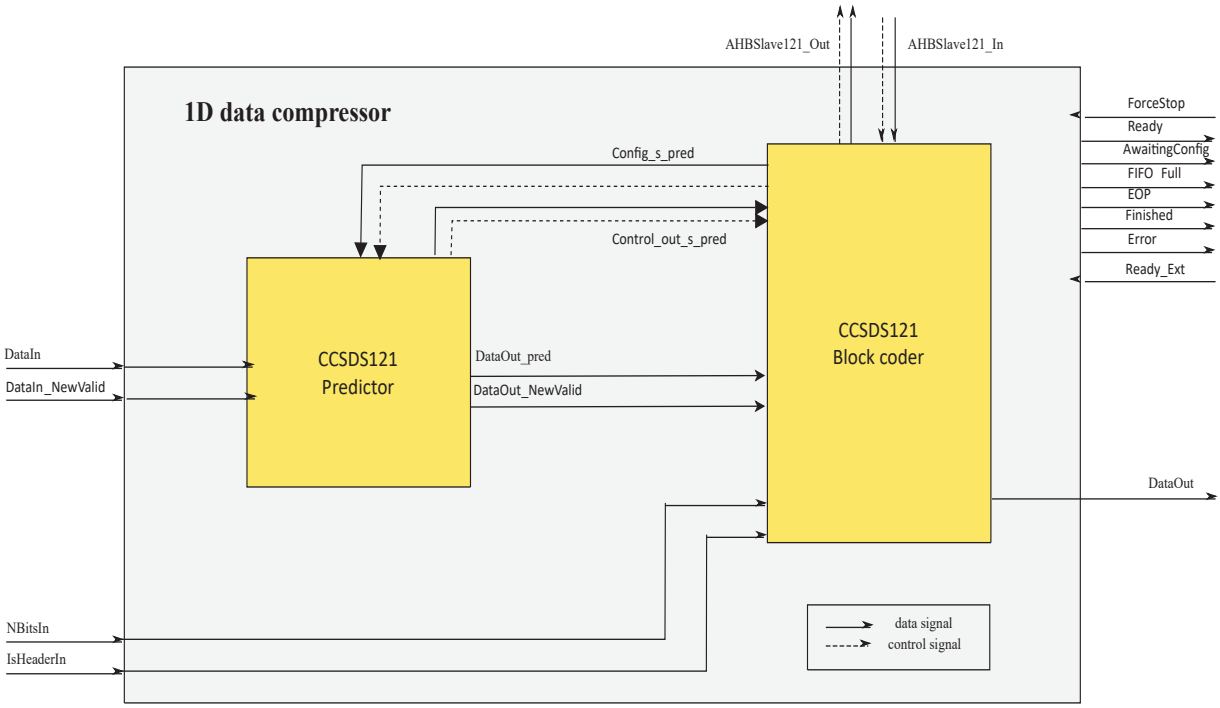


FIGURE 5.3: Schematic of the proposed IP core for lossless compression of one-dimensional data

5.3.2 Experimental results

Different sets of configuration parameters have been defined to analyse the impact of the IP parameter values on both hardware occupancy and maximum clock frequency. These synthesis sets are summarised in Table 5.1. The main parameters that change among configurations are the dynamic range of the input samples *D_GEN*; the block size *J_GEN*; and the width of the output buffer, *W_BUFFER_GEN*. In addition, the penalty for for managing signed samples is studied. Finally, the influence of the selected mode for the header generation is also taken into account. Regarding the maximum data size, it has been fixed to 2048 for each spatial dimension in case that the IP is used for 2D image compression.

The timing results in terms of maximum clock frequency are reflected in Tables 5.2, 5.3, 5.4 and 5.5 for Xilinx Virtex5QR XQR5VFX130 and Kintex UltraScale XCKU040, Microsemi RTG4 150 and NanoXplore NG-LARGE, respectively. In general, Set5 and Set7 achieve the highest values of clock frequency, which is clearly conditioned by the small values selected for the dynamic range D and, specially, for the block size J. Combining low values for both parameters (i.e., $D = 8, 16$ and $J = 8$) allows to reach that high clock frequency

TABLE 5.1: Sets of synthesis configurations for the lossless one-dimensional data compression IP core

Generic	Set1	Set2	Set3	Set4	Set5	Set6	Set7
EN_RUNCFG	0	1	1	1	0	1	0
D_GEN	32	16	32	32	16	16	8
IS_SIGNED_GEN	0	0	1	0	0	1	0
J_GEN	32	16	16	64	8	32	8
REF_SAMPLE_GEN	128	256	256	4096	512	256	256
W_BUFFER_GEN	64	32	64	64	48	32	8
HEADER_FORMAT_GEN	0	0	0	0	1	1	1

because of the reduction of the datapath length and the simplification of some internal operations, which are dependent on those parameter values.

Specifically, a maximum clock frequency of 176.3 MHz is obtained for the Kintex UltraScale XCKU040 under Set7, while up to 115.9 MHz are reached for Virtex5QR XQR5VFX130 under Set5. In the latter case, a minimum penalty in terms of maximum clock (around the 0.7%) is appreciated when selecting Set7 instead of Set5. The maximum clock frequency is also obtained for RTG4 150 and NG-LARGE when using Set7, achieving up to 79 and 34.1 MHz, respectively. As expected, Set4 is the configuration that obtains the lowest clock frequency, since the maximum allowed values of D and J (32 and 64, respectively) are used. Regarding throughput, as the whole IP is able to process one sample per clock cycle, the maximum throughput is equal to the maximum clock frequency achieved. In this way, a maximum throughput of 176.3 and 163.7 MSamples/s is obtained for Xilinx XCKU040 when storing input samples with 8 and 16 bits, respectively.

The hardware occupancy in terms of memory blocks, arithmetical units and logic resources is also provided in Tables 5.2, 5.3, 5.4 and 5.5 for Virtex5QR XQR5VFX130, Kintex UltraScale XCKU040, RTG4 150 and NG-LARGE, respectively, for each one of the proposed configuration sets. The results in terms of logic resources utilization are in consonance with the ones obtained in terms of timing. Set5 and Set7 have the lowest area footprint, since low values of D and J simplify some internal computations and reduce memory demand. When using Set7, just the 2.2% and the 0.7% of the available LUTs are used on Virtex5QR and Kintex UltraScale, respectively. Under the most restrictive configuration (i.e., Set4) these numbers increase up to the 13.7% and the 4.2% for XQR5VFX130 and XCKU040, respectively.

The maximum area occupation for Microsemi RTG4 150 and NanoXplore NG-LARGE is also observed for Set4, and it is around the 11.4% and the 13.1% of the available LUTs,

respectively. The memory utilization is the same in both Xilinx technologies (5 BRAMs), implying just the 1.7% of the total available on Virtex5QR. In the case of RTG4 150 and NG-LARGE, maximum BRAM utilization supposes the 10% and the 10.4% of the total available on the device, respectively, when Set4 is selected. DSPs consumption is also minimum on the FPGAs analysed for all the configuration sets, reaching a maximum utilization of the 1.9% and the 0.3% on XQR5VFX130 and XCKU040, respectively, under Set4.

TABLE 5.2: Lossless one-dimensional data compression IP core - Synthesis on Xilinx Virtex5QR XQR5VFX130

Parameters	**Total**	**Set1**	**Set2**	**Set3**	**Set4**	**Set5**	**Set6**	**Set7**
Block RAMs	298	2	0	1	5	0	1	0
DSP48	320	2	5	5	6	2	7	1
Registers	81920	2008	1625	2350	2706	1360	1827	808
LUTs	81920	7078	4670	8514	11231	3586	5827	1782
Maximum Frequency (Clk_S) (MHz)		95.4	110.1	94.5	62.0	115.9	96.9	115.1

TABLE 5.3: Lossless one-dimensional data compression IP core - Synthesis on Xilinx Kintex UltraScale XCKU040

Parameters	**Total**	**Set1**	**Set2**	**Set3**	**Set4**	**Set5**	**Set6**	**Set7**
Block RAMs	600	2	0	1	5	0	1	0
DSP48	1920	2	5	5	6	2	6	1
Registers	484800	2028	1632	2357	2754	1318	1830	813
LUTs	242400	6140	4313	7883	10994	3239	5252	1655
Maximum Frequency (Clk_S) (MHz)		151.2	160.5	137.9	120.1	163.7	141.1	176.3

TABLE 5.4: Lossless one-dimensional data compression IP core - Synthesis on Microsemi RTG4 150

Parameters	**Total**	**Set1**	**Set2**	**Set3**	**Set4**	**Set5**	**Set6**	**Set7**
Carry Cells	151824	2649	1401	2989	3091	1071	1604	435
Sequential Cells	151824	1820	1430	2014	2685	1090	1644	727
Block RAMs (RAM64x18)	209	21	11	19	21	13	14	9
DSP Blocks	462	5	6	6	6	5	6	4
LUTs	151824	9987	7362	12809	17364	5380	8727	2478
Maximum Frequency (Clk_S) (MHz)		55.5	58.8	56.0	35.0	78.1	65.1	79.0

TABLE 5.5: Lossless one-dimensional data compression IP core - Synthesis on NanoXplore NG-LARGE

Parameters	Total	Set1	Set2	Set3	Set4	Set5	Set6	Set7
Carry Cells	32256	5322	2367	5778	5938	1917	2411	939
Registers	129024	2777	1674	2862	3401	1471	1850	868
Block RAMs (48Kb)	192	20	13	20	20	13	13	11
DSP Blocks	384	1	2	2	4	1	3	1
LUTs	129024	10963	6806	11903	16871	5637	8104	2196
Maximum Frequency (Clk_S) (MHz)		21.1	25.2	19.5	16.5	25.5	20.7	34.1

5.3.3 Demonstrator set-up

A demonstrator has been developed in order to validate the behaviour of the proposed IP core in a realistic scenario from a functional point of view, simulating the system connectivity expected on-board satellites. The demonstrator implements the proposed one-dimensional data compressor with the configuration defined as Set5 in Table 5.5, on a NX1H140TSP development kit that mounts a NanoXplore NG-LARGE FPGA, providing a novelty to the state-of-the-art since it is, to the best of our knowledge, the first on-chip validation of a compression solution in this technology. The FPGA implementation is completed with the STAR-Dundee SpaceWire Codec IP [185], which belongs to the ESA's IP cores portfolio, to enable data transfers through a SpaceWire interface [186]; and some decoupling FIFOs, which are used to adapt data rates between both IP cores. The block diagram of the hardware design is reflected in Figure 5.4. Besides, a time-code control logic is included that allows to configure the SpaceWire Codec IP as both a time-code slave or master, useful to debug communication establishment between the host PC and the FPGA. This time-code unit is controlled by an input mapped to one of the available switches on the development board, while some status signals of the SpaceWire Codec IP are mapped to LEDs.

The whole design uses a single clock source of 20 MHz, generated internally by using one of the Phase-Locked Loops (PLL) available in the FPGA, which is used as the system clock of both IP cores as well as the transmission clock for the SpaceWire Codec IP. I/O ports of the STAR-Dundee SpaceWire Codec IP are mapped to one of the SpaceWire connectors available on the NX1H140TSP development kit. The other end of the communication link is a STAR-Dundee SpaceWire Brick Mk3 [187], which is connected to a host PC from where the Brick is configured, and where both the transmission and reception of data are controlled. The FPGA bitstream is received and loaded into the configuration memory

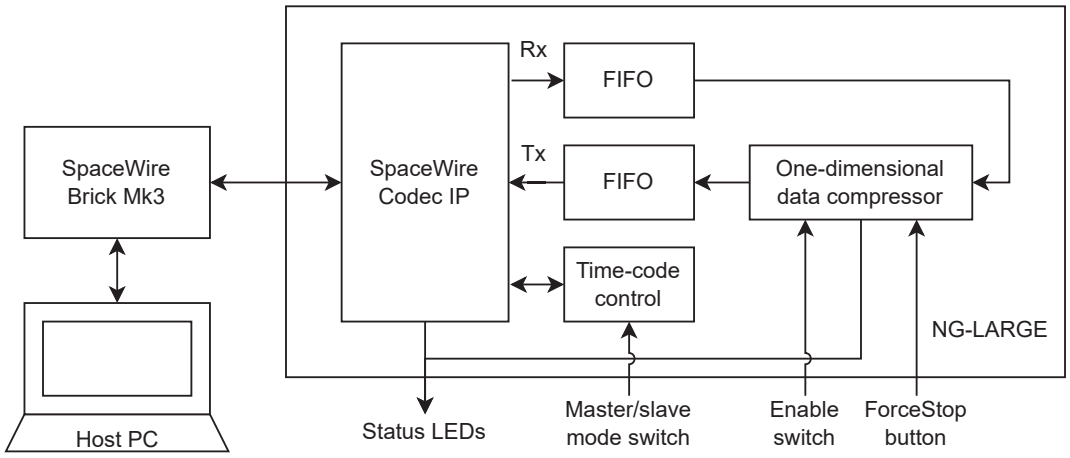


FIGURE 5.4: Block diagram of the hardware design developed for the IP core test set-up

by using a dedicated JTAG interface connected to the host PC, where that bitstream is generated. The STAR-Dundee SpaceWire Codec IP is configured in *autostart* mode, so the link is initiated by the host PC controlling the SpaceWire Brick. The link is successfully established when activated by the host PC, by means of the STAR-Dundee Device Configuration application. The whole test set-up is shown in Figure 5.5.

With this test set-up, a corpus of data sets has been successfully compressed by sending the original data through the SpaceWire link within the FPGA, where they are compressed by the IP. Compressed streams are sent back to the host PC through the SpaceWire brick to be compared against reference streams generated by the CCSDS 121.0-B-3 reference software developed in collaboration between our Group and ESA. In order to do so, the SpaceWire Brick Mk3 is configured with a data transmission speed of 10 Mbits per second. Once the link is established, the Brick detects a speed of 20 Mbits per second on the reception channel.

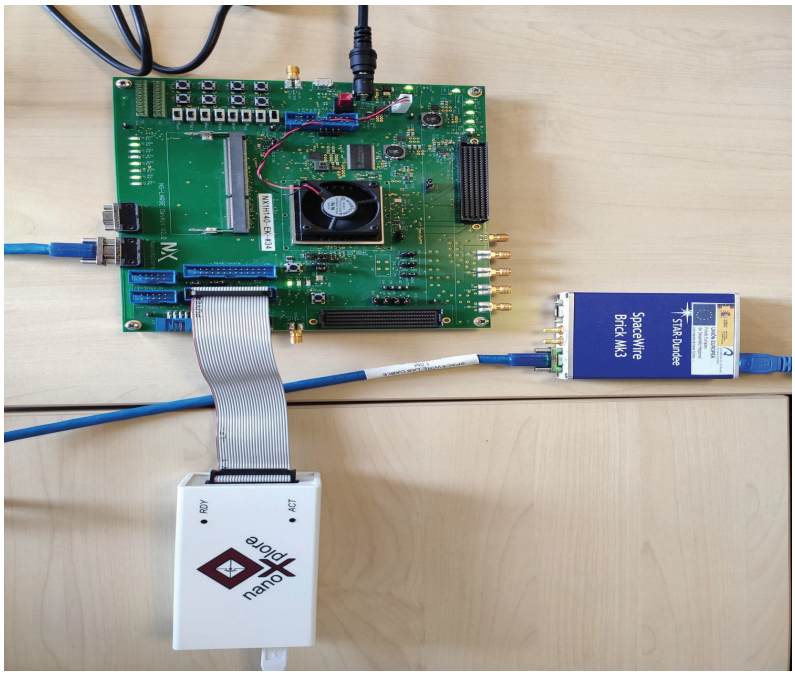


FIGURE 5.5: Overview of the IP test set-up, including the NX1H140TSP development kit and the STAR-Dundee SpaceWire Brick Mk3

5.4 Lossless hyperspectral image compression

This validation scenario provides solutions to compress hyperspectral images in lossless mode. As depicted in 5.6, the 3D predictor described in Section 3.3 and compliant with the CCSDS 123.0-B-1 standard is used. Although the choice of any of the three entropy coders presented in Chapter 4 would implement a valid lossless compression solution, only the selection of the block adaptive or sample adaptive encoders is compliant with the CCSDS 123.0-B-1 standard.

As an example, a hardware implementation fully compliant with the CCSDS 123.0-B-1 standard has been designed for lossless compression of multi- and hyperspectral images on-board satellites. Results are given for a number of solutions for typical spectroscopic sensors (e.g. Landsat, AVIRIS and AIRS).

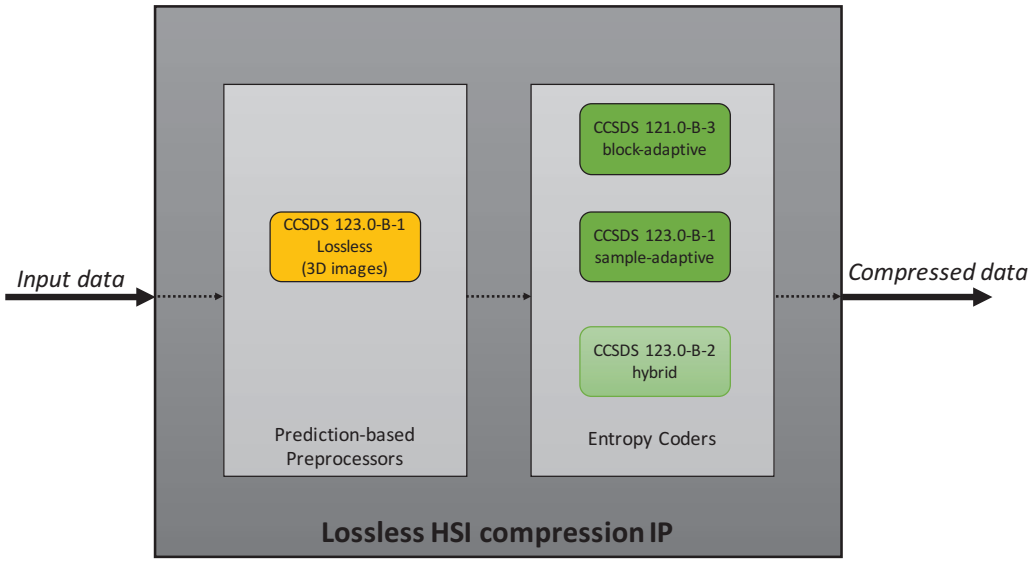


FIGURE 5.6: Block diagram of the lossless hyperspectral image compression IP

5.4.1 System development

As mentioned before, the proposed standard compliant solution is based on the CCSDS 123.0-B-1 3D lossless predictor and either the sample-adaptive or the block-adaptive encoder. The top module of the IP core includes, in addition to the prediction-based preprocessor and the entropy coding stage, the components depicted in Figure 5.7, which are necessary to receive the configuration and the input samples, and to send the compressed bitstream to the output. These modules are summarised below:

- **Configuration core**. It includes several units to perform the configuration of the whole compression chain. Runtime configuration values are received from the AMBA AHB slave interface, which includes two independent signals for data transfers (*AHBSlave123_In* and *AHBSlave123_Out*). The *clk_adapt* module is the responsible of adapting data rates with the rest of the system, if needed. The *interface_gen* module has a twofold purpose: on one hand, it assigns the configuration values, either from the ones received by the AHB slave interface, if runtime configuration is enabled ($EN\_RUNCFG = 1$), or from the user-selected parameters defined at compile-time, if runtime configuration is disabled ($EN\_RUNCFG = 0$); on the other hand, it checks that the selected configuration values are correct, raising an error otherwise. These values are sent to the rest of the modules in the design, including

the *header_gen* submodule, responsible of generating the predictor fields and also the ones associated to the selected entropy coding stage.

- **Control decoder**. It generates and interprets control signals to and from I/O ports and ensures that the compression chain is ready to receive samples to be compressed.
- **Dispatcher**. It receives the output from the compression engine and organizes it in FIFOs so that it can be sent to the output according to the selected configuration, including both the header and the compressed bitstream. This module will monitor the *Ready_Ext* signal and the status of the FIFOs to signal if it is necessary to interrupt the compression, when the receiver is not ready to accept the compressor output.

Input and output data are handled with ad-hoc interfaces, including a *Valid* flag. The data flow is managed by a handshake protocol, which includes the *Ready* signal to inform that the IP can receive new samples. The *Ready_Ext* input signal is asserted to notify that any available output data can be flushed, since the receiver is available. Additional signals are defined for control purposes, including the *ForceStop* to suspend the current compression; *FIFO_full*, to indicate the state of the FIFOs responsible of providing and receiving the input and output samples,respectively; *EOP*, which informs that the compression of the last sample has started; or *Finished*, which is asserted when the compression has ended. The output signal *IsHeaderOut* is included to indicate that the output data is part of the header, easing the substitution of the sample-representative encoder by an alternative entropy coder, if desired. Finally, the *Error* signal is asserted in case of an unexpected behaviour during the compression.

The IP core is able to compress samples in BIP, BSQ and BIL orders, since all these arrangements are supported by the main functional elements (i.e., the predictor and the sample-adaptive encoder). The order is selected at compile-time with the *PREDICTION_TYPE* generic parameter that determines the hardware architecture that is implemented, taking into account data dependencies and memory demands. In case that the selected predictor architecture makes use of an external memory to store intermediate compression statistics, an AMBA AHB Master interface is instantiated, providing independent channels for data reception and transmission (*AHBMaster123_In* and *AHBMaster123_Out*, respectively). Also, the possibility of enabling or disabling EDAC at compile-time is included. The user can select to use EDAC for internal, external or both types of memories.

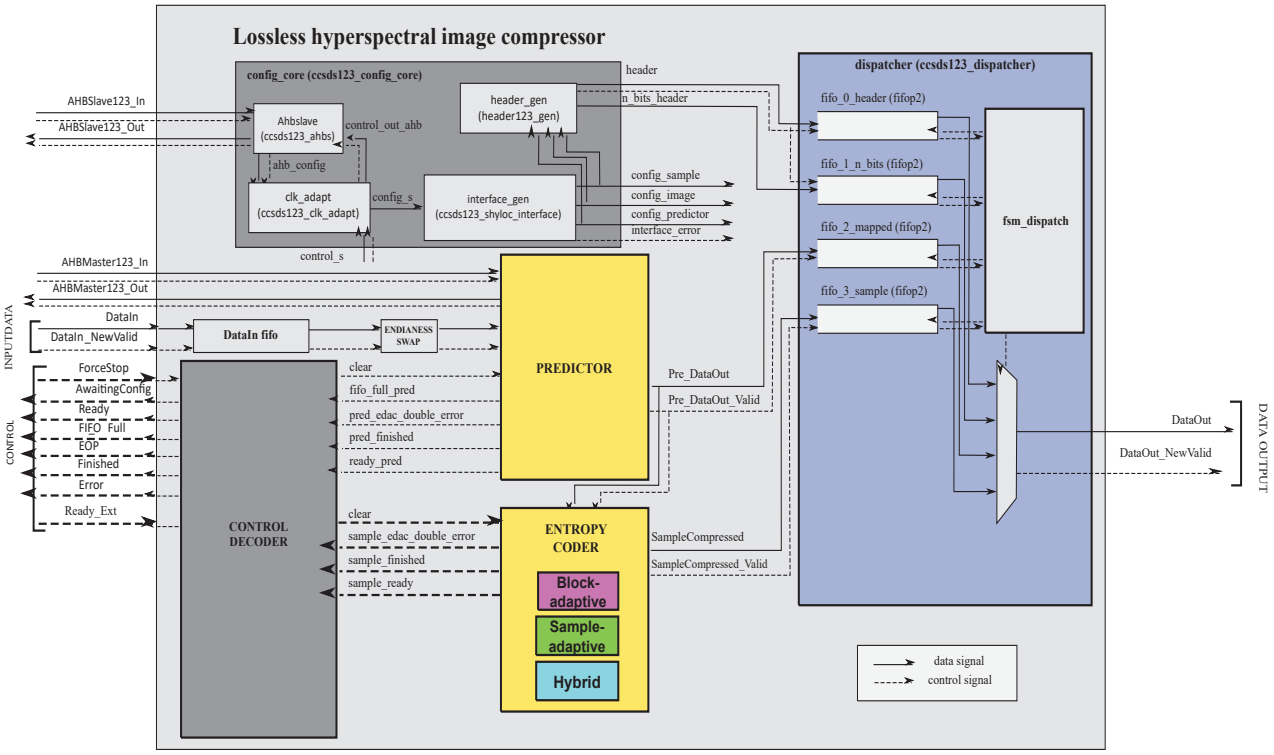


FIGURE 5.7: Internal overview of the proposed IP core for lossless hyperspectral image compression and connectivity among functional units

By default, the sample-adaptive encoder is implemented. When the block-adaptive encoding option is used, the block-adaptive encoder is connected in a plug&play manner thanks to the generic interfaces defined for both stages. The selection of the appropriate entropy coding stage is done through a VHDL generic during compile-time configuration.

5.4.2 Experimental results

The IP core has been evaluated using up to 20 different sets of synthesis parameters (15 in compile-time and 5 in runtime configuration), trying to cover a broad range of cases and verifying all the predictor architectures. Table 5.6 shows the values used during the synthesis for some relevant configuration parameters, excluding image dimensions which are dependent on the analyzed scenario. Real spectroscopic sensors, such as Landsat, AVIRIS and the Atmospheric Infrared Sounder (AIRS), have been considered to obtain representative results for multi- hyper- or ultraspectral images, respectively. Main features of these scenes are summarised in Table 5.7. As entropy coding option, just the sample-adaptive alternative is considered for synthesis purposes.

Results for the IP core in terms of maximum clock frequency are shown in Tables 5.8, 5.8, 5.8 and 5.8 for Xilinx Virtex5QR XQR5VFX130 and Kintex UltraScale XCKU040, Microsemi RTG4 150 and NanoXplore NG-LARGE, respectively. These results are provided for all the developed predictor architectures, whose selection depends on input samples arrangement, and for the different scenarios proposed in Table 5.7. For the AVIRIS sensor, two different synthesis were performed, one allowing just compile-time configuration and the other enabling also runtime configuration.

The maximum clock frequency achieved for Virtex5QR is above 109 MHz for all the configurations implemented, with the exception of the BSQ architecture under the ultraspectral scenario, where this maximum frequency just reaches 95 MHz. When targeting Kintex UltraScale XCKU040, clock frequencies above 119 MHz are obtained for all the developed predictor architectures in all the targeted scenarios. For the Microsemi RTG4 technology, the system clock frequencies range from 73 to almost 95 MHz. Finally, clock frequencies

Table 5.6: Sets of synthesis configurations for the lossless hyperspectral image compression IP core

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

Table 5.7: Scenarios considered for the IP performance analysis

Instrument	Image size			Bit depth
	Nx	Ny	Nz	
Landsat	1024	1024	6	8
AVIRIS (calibrated)	677	512	224	16
AIRS	90	135	1501	12-14

between 27 and 41 MHz are obtained for NG-LARGE. Besides, the IP core has been developed 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 for all the FPGA technologies under study in all the implemented architectures. However, not all the architectures provide the same throughput, being BIP the one that can reach the maximum throughput of one sample per clock cycle. For this reason, maximum throughput is 133.8 and 151.6 MSamples/s on Virtex5QR (with runtime configuration disabled) and XCKU040, respectively, targeting the hyperspectral scenario.

Synthesis results in terms of resources utilization are summarised in Figures 5.8, 5.9, 5.10 and 5.11 for the FPGA technologies under study. In general, the IP uses low logic resources, which proves the low complexity of the proposed compression chain. The usage of DSP units, LUTs, registers and carry cells is almost constant for each FPGA technology, with slight differences depending on the implemented compressor architecture, the target image size or the possibility of configuring at run-time or not. Differences are more noticeable for the memory utilization, which is mainly determined by the predictor architecture. For those architectures that do not make use of external storage (BIP and BIL), the image size has a high impact on BRAMs usage. Concretely, the size of an spectral line ($N_x N_z$) conditions that storage utilization since it fixes the size of memory elements, such as the FIFOs that store the adjacent samples for the local sum and differences calculation during the prediction. The number of P previous bands used for prediction also has an influence in the BRAMs consumption, since it determines 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 operations. 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. Up to the 40% and the 20% of memory resources are used in Xilinx Virtex5QR and Kintex UltraScale

Table 5.8: Lossless hyperspectral image compression IP core - Maximum frequency on Xilinx Virtex5QR XQR5VFX130

Config	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	109.4	109.4	107.6	109.4	125.8
AVIRIS	138.3	109.4	110.8	109.4	109.4
AIRS	109.4	109.4	95.0	109.3	109.3
Runtime config.	109.3	140.8	110.4	111.9	110.3

Table 5.9: Lossless hyperspectral image compression IP core - Maximum frequency on Xilinx Kintex UltraScale XCKU040

Config	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	149.4	157.6	139.5	166.4	164.1
AVIRIS	151.6	150.3	140.7	160.6	152.7
AIRS	149.7	138.2	119.9	152.8	126.2
Runtime config.	151.6	149.8	135.1	148.0	149.8

Table 5.10: Lossless hyperspectral image compression IP core - Maximum frequency on Microsemi RTG4 150

Config	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	94.9	94.2	77.8	94.7	91.9
AVIRIS	74.8	84.1	77.8	73.7	85.4
AIRS	87.2	83.1	77.5	84.2	86.5
Runtime config.	84.0	83.6	79.0	84.6	81.5

Table 5.11: Lossless hyperspectral image compression IP core - Maximum frequency on NanoXplore NG-LARGE

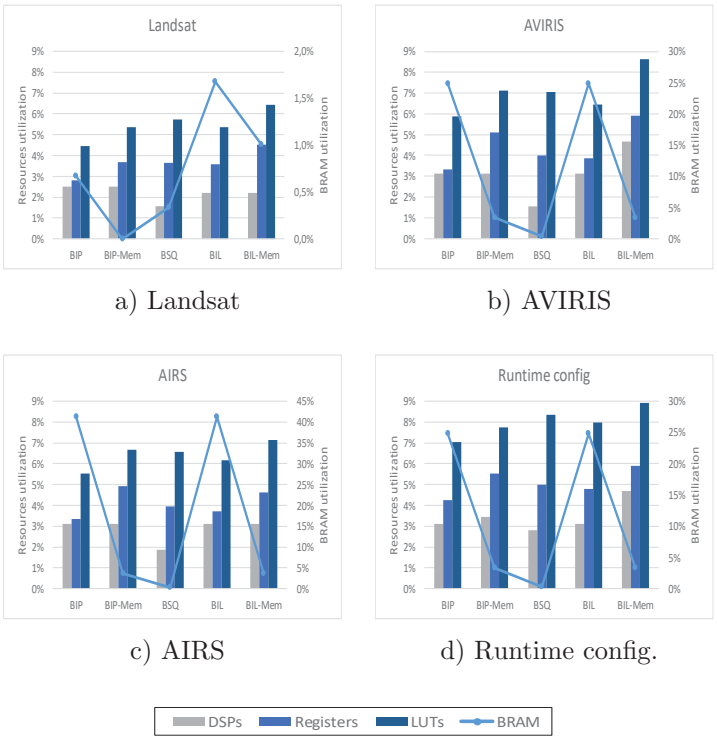
Config	BIP	BIP-MEM	BSQ	BIL	BIL-MEM
Landsat	37.1	40.6	38.9	37.0	33.5
AVIRIS	38.9	39.8	28.9	31.1	27.8
AIRS	30.2	36.3	30.7	31.5	32.3
Runtime config.	31.5	30.9	27.6	29.2	27.8

XCKU040, respectively, when targeting a ultraspectral scenario. With respect to the DSPs and LUTs utilization, all the architectures and configurations provide similar results for Virtex5QR, with ranges between 1.5% and 4.7% for DSPs, and between 4.5% and 9% for LUTs. In Xilinx technologies, the DSP utilization tends to be lower for the BSQ architecture, due to its serial implementation. The BIP and BIL architectures use less LUTs compared with the others, due to the absence of the AHB interface to communicate with an external memory, and the LUT utilization slightly increases with the image dimensions in all the implemented architectures.

In the case of Microsemi RTG4, the memory demands when compressing ultraspectral scenes (e.g. AIRS) are so high that exceed the available FPGA internal storage, as shown in Figure 5.10. This forces to use an external memory for that specific scenario. 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 interface increases the resource usage of all these elements in about 1%. When targeting NG-LARGE, the memory usage grows up to a maximum of 80% for the AIRS ultraspectral scene. Regarding logic resources, LUTs consumption is in the range of 2.5% - 6%, while DSPs are under the 2% for all the cases.

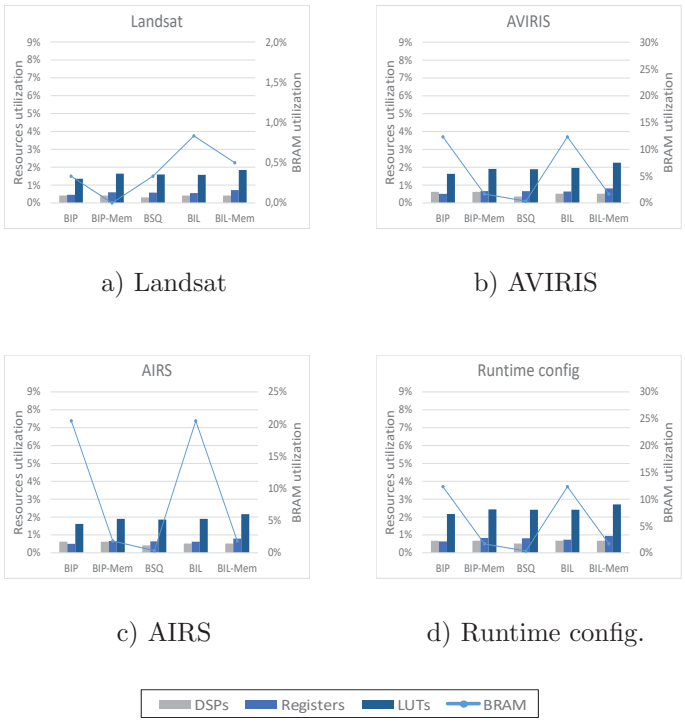
In general, it can be observed that the inclusion of the AHB configuration interface increases the LUT utilization only about a 1% for all the FPGA technologies under study, compared to the results obtained for AVIRIS just configuring the compressor at compile-time.



a) Landsat b) AVIRIS

c) AIRS d) Runtime config.

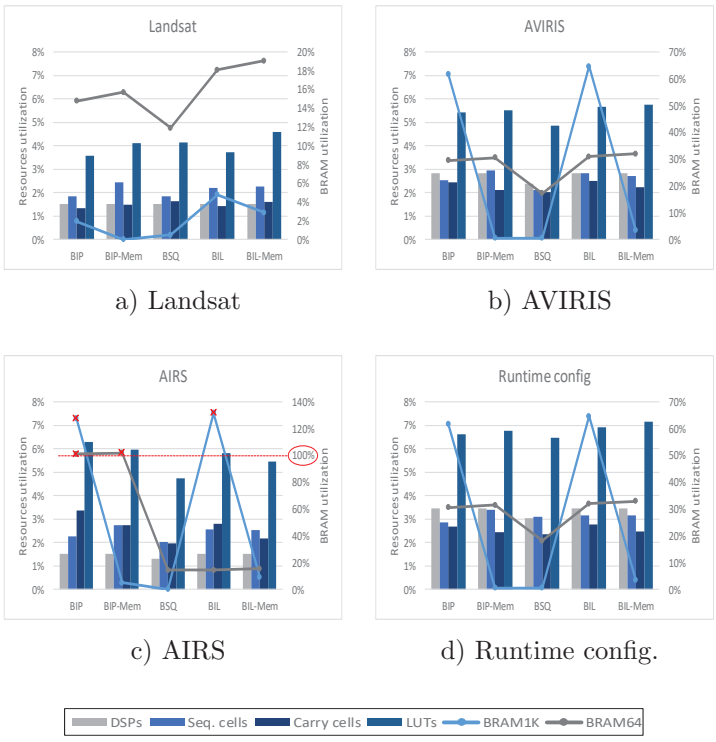
FIGURE 5.8: Lossless hyperspectral image compression IP core - Resources utilization on Xilinx Virtex5QR XQR5VFX130 [60]



a) Landsat b) AVIRIS

c) AIRS d) Runtime config.

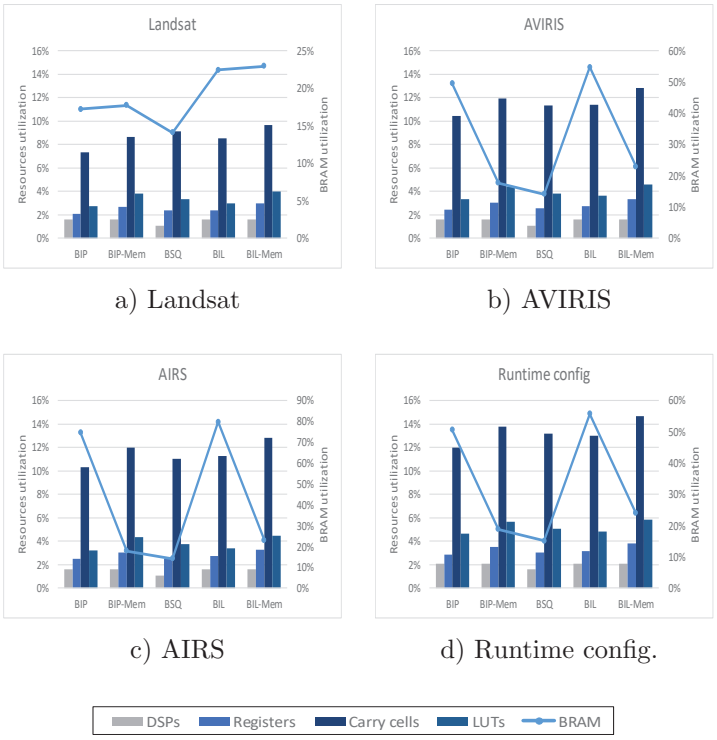
FIGURE 5.9: Lossless hyperspectral image compression IP core - Resources utilization on Xilinx Kintex UltraScale XCKU040



a) Landsat b) AVIRIS

c) AIRS d) Runtime config.

FIGURE 5.10: Lossless hyperspectral image compression IP core - Resources utilization on Microsemi RTG4 150 [60]



a) Landsat b) AVIRIS

c) AIRS d) Runtime config.

FIGURE 5.11: Lossless hyperspectral image compression IP core - Resources utilization on NanoXplore NG-LARGE [60]

5.4.3 Comparison with state-of-the-art implementations

Since the CCSDS 123.0-B-1 lossless compression standard is becoming interesting for future space missions that embark spectroscopic sensors, there are several implementations of this algorithm on FPGAs in the specialized literature. Nonetheless, the comparison of the developed IP core with other FPGA implementations available in the state-of-the-art is not easy, because the presented solution allows a high quantity of architectures and configurations that results in different performance capabilities. This flexibility 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, targeting even in some cases commercial devices.

For comparison purposes, the IP architecture that achieves the best throughput for the AVIRIS sensor ($D = 16$) is considered, in order to provide a fair comparison with the existing FPGA implementations [157, 159, 188–191]. Besides, resources utilization (if available) is

analyzed for the mentioned implementations. Although there are other implementations available in the specialized literature, they are based on a parallel approach, instantiating multiple compressor copies that work simultaneously to maximize throughput. For this reason, they are not considered as part of this comparison, focusing the attention on those works where the performance of just a single compression instance is analysed. This comparison is summarised in Table 5.12.

The implementation proposed by Santos et al. [188] focuses on obtaining a low-complexity architecture in terms of resources utilization, in order to fit in older generations of space-grade FPGAs. 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. In [189], 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. Bascones et al. [190] propose an implementation that works also in BIP order and it is optimized to use only internal memory storage. Using this scheme, they achieve a throughput of almost 48 Msamples/s when it is implemented on a Xilinx Virtex-7 XC7VX690T.

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. [157]. 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. 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 addition to implementations targeting space-grade FPGA technologies, a comparison is also done with implementations of the CCSDS 123.0-B-1 algorithm on commercial

Table 5.12: Comparison with CCSDS-123 FPGA implementations

Implementation	Order	P	D	Encoder	Device	LUTs	FFs	BRAMs	Throughput (MSamples/s)
HyLoC, Santos et al.[188]	BSQ	3	16	Sample	Virtex XQR5VFX130	2342	1535	0	11.3
Keymeulen et al. [189]	BIP	3	13	Golomb-Rice	Virtex SX50T	12697	1586	8	40
Bascones et al. [190]	BIP	0-15	16	Sample	Virtex XC7VX690T	-	-	-	47.6
Tsigkanos et al. [157]	BIP	3	16	Sample	Virtex FX130T	9462	9990	83	213
Pereira et al. [191]	BIP	3	16	No	Zynq-7020	2244	630	0	20.4
Fjeldtvedt et al. [159]	BIP	0-15	16	Sample	Zynq-7020	3012	2528	84	147
Developed IP	All	0-15	16	Sample/Block	Virtex XQR5VFX130	4809	2736	74	138.3
Developed IP	All	0-15	16	Sample/Block	Zynq-7020	4619	2765	74	151.1

FPGAs. In this way, Pereira et al. [191] propose a low-complexity implementation of the prediction stage (column-oriented local sum and reduced prediction mode are selected), targeting a Xilinx Zynq-7020 and achieving a maximum throughput of 20.4 MSamples/s. This architecture works in BIP order, storing the weights in an internal memory but out of the processing core, in order to share these values among multiple copies of the compression instance, if needed for higher throughput. The work of Fjeldtvedt et al. [159], also targeting a Xilinx Zynq-7020, proposes a BIP architecture with a Sample Delay module that works in a similar way that the Slice Buffer proposed by [157]. Different pipelining strategies are applied in the prediction module in order to reduce delay, splitting the critical path located in the local sum calculation and in the weights updating. With this scheme, they achieve a maximum throughput of 147 MSamples/s.

The 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-7020 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 and the BIP architecture is selected, since it obtains the best results in terms of throughput. If higher performance is required, several measures can be taken by the user, such as breaking the critical path with intermediate registers, or instantiating several copies to work in parallel. Although our implementation of the CCSDS 123-0-B-1 standard exhibits lower throughput than the implementation presented in [157], it represents a good trade-off among timing performance, resources utilization, versatility and portability.

5.5 Near-lossless hyperspectral image compression

This validation scenario provides compression of hyperspectral images in near-lossless mode. As depicted in Figure 5.12, the CCSDS 123.0-B-2 near-lossless predictor can be combined with any of the entropy coders developed and explained in Chapter 4. In this way, a full compression solution based on the CCSDS 123.0-B-2 standard is provided, which can be used to demonstrate the viability of this compression solution for different space programs that embark spectroscopic sensors and need from high CRs.

As an example, the compression IP of the CHIME instrument is presented. Taking into account the CR required in this mission, it is not feasible to use a lossless compressor. As entropy coder, the block-adaptive encoder is used, because it has a lower area footprint

and a higher throughput than the hybrid one. Besides, it will be possible to achieve CRs under 1 bpp, which is the reason why the sample-adaptive encoder has been discarded for this implementation.

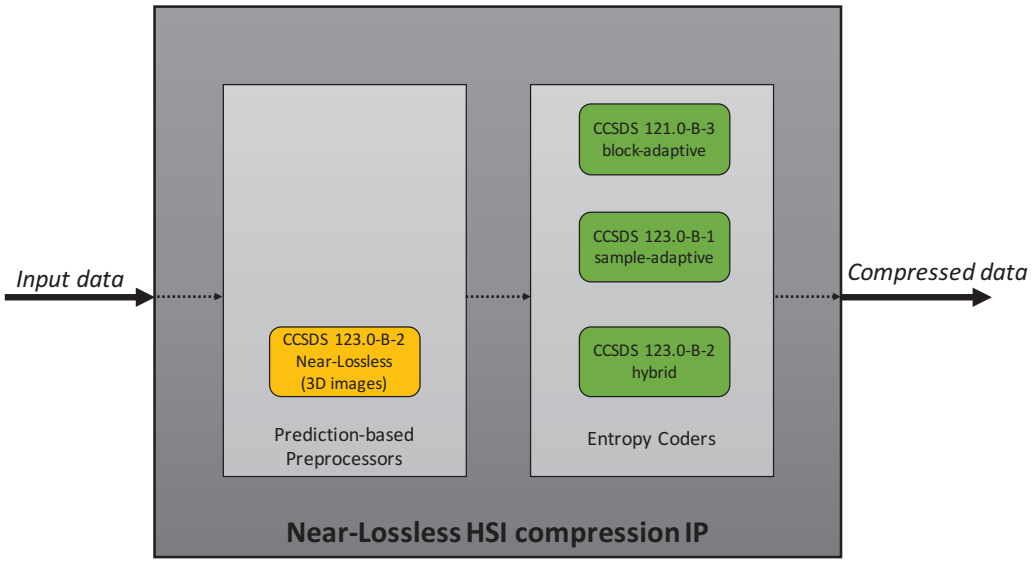


FIGURE 5.12: Block diagram of the near-lossless hyperspectral image compression IP

5.5.1 Compression solution fully compliant with the CCSDS 123.0-B-2 standard

The solution proposed for this scenario is an IP fully compliant with the CCSDS 123.0-B-2 standard, including all the possible alternatives for the prediction and local sum steps and the whole range of values for each one of the parameters defined by the standard, which can be configurable at compile-time or at runtime.

5.5.1.1 System development

A global overview of the whole design is shown in Figure 5.13, including the different interfaces used for configuration and interconnection among modules. I/O interfaces implement the AXI4-Stream protocol to speed up data transfers to/from the compression solution. In addition to the main compression chain, comprised by the predictor and the hybrid encoder, respectively described in Sections 3.4.2 and 4.4.2, two extra modules are included and described below:

- **Header generator**. This module is responsible of generating the appropriate header fields, including both predictor and hybrid encoder information, to allow a correct decompression of the generated bitstream. The length of this header directly depends on the compression configuration. Image, predictor and encoder parameters are sent to this module through an AXI4-Lite interface, while the generated header is sent through a lightweight custom interface to the bitpacker. This module works in parallel to the main compression datapath, since its behaviour is essentially sequential. In this way, the global latency of the system is balanced.

- **Bitpacker**. It is the last stage of the compression chain, taking the output of both the header generator and the hybrid encoder modules to generate the compressed image, formed by the header, the codewords and the image tail, created also in this module to reduce the latency of the hybrid encoding stage. The two input data interfaces of this module, one to receive the header coming from the header generator and other to receive the bitstream from the hybrid encoder, are implemented with custom interfaces, include the necessary control signals for dataflow management. Taking into account the value of the *Valid* signals of both input interfaces, the IP is able to identify if it should be ready to receive the header or the bitstream, appending both information properly at the output interface, which is implemented with an AXI4-Stream interface. When the *EOP* flag generated by the hybrid encoder is asserted, the bitpacker recognizes that both the header and the bitstream have been received, and it is time to generate the image tail, as it is indicated in Section 4.4.1. For this purpose, it accesses to the two FIFOs where the final state of both the 16 low-entropy codes and the accumulator values for each band Σ_z have been stored.

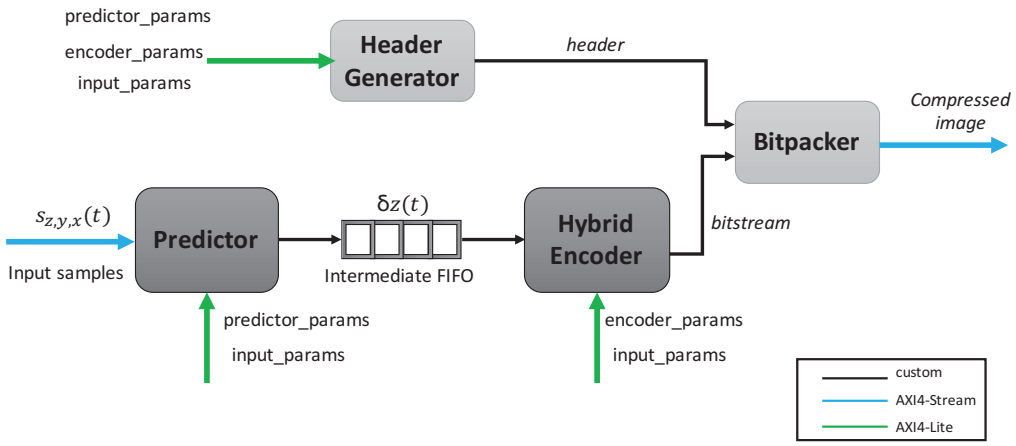


FIGURE 5.13: Top-level hierarchy of the near-lossless hyperspectral image compression solution fully compliant with the CCSDS 123.0-B-2 standard

5.5.1.2 Experimental results

Several tests have been performed by combining 7 different images, including AVIRIS scenes and synthetic images to debug corner cases, with multiple configuration sets, ensuring that at least the different local sum and prediction options are covered in both lossless and near-lossless modes. In the latter case, both band-dependent and band-independent errors are used.

Then, the proposed compression solution has been synthesized targeting the Kintex UltraScale XCKU040 FPGA. The baseline configuration for synthesis purposes is summarised in Table 5.13, restricting supported image dimensions and the D value to the features of AVIRIS scenes, since they are the ones used for validation purposes. The rest of parameters values have been selected after an exhaustive parameter tuning on software, selecting the configuration that maximize CR under lossless mode. The optimal combination of local sum and prediction modes is defined taking into account the results previously presented in Tables 3.8 and 3.9 in Chapter 3. The absolute error value can be modified, depending on the requirements of the target application.

Area consumption of the whole compression chain is summarised in Table 5.14, specifying the resources utilization of each stage. As it can be observed, the predictor is the critical module in terms of both memory and logic resources usage, consuming the 12.7% of BRAMs and the 4.1% of LUTs available in the target device.

In general, the results for this full approach are as expected, since the prediction stage is the one that performs more complex operations, such as the ones performed under near-lossless compression in the feedback loop. The consumption of DSPs supposes around the 3.3% of the total available in the device. These resources are mainly used to perform the different multiplications present in the design, such as the multiplication by the precomputed inverse performed in both the quantizer and the mapper, and the dot product between the local differences vector $U_{z,y,x}$ and the weights vector $W_{z,y,x}$, both performed in the prediction stage; or to compute Equations 4.3 and 4.4, calculated under the low-entropy mode in the hybrid encoder.

All the modules that conform the compression chain have been successfully synthesized with a targeted maximum clock frequency of 125 MHz, the maximum reached by the predictor, which is the bottleneck of the system.

TABLE 5.13: Main configuration parameters of the CCSDS 123.0-B-2 proposed IP used for synthesis purposes

Parameter	**Value**
Image parameters	
Columns, N_x	677
Lines, N_y	512
Bands, N_z	224
Dynamic Range, D	16
Encoding Order	BIL
Predictor parameters	
Bands for Prediction, P	3
Local Sum Mode	Narrow Neighbour-Oriented
Prediction Mode	Full Prediction
Weight Resolution, Ω	16
Sample Adaptive Resolution, Θ	2
Sample Adaptive Offset, ψ_z	1
Sample Adaptive Damping, ϕ_z	1
Error Method	Absolute
Absolute Error Bit-depth, D_a	8
Absolute Error Value, A_z	4
Hybrid Encoder parameters	
Unary Length Limit, U_{max}	16
Rescaling Counter Size, $\gamma*$	5
Initial Count Exponent, γ_0	1

TABLE 5.14: Near-lossless hyperspectral image compression IP - Resources utilization on Xilinx Kintex UltraScale XCKU040

	36Kb BRAM	DSP48E	Registers	LUTs
Predictor	76 (12.7%)	54 (2.8%)	6054 (1.3%)	10087 (4.1%)
Hybrid Encoder	9 (1.5%)	9 (0.5%)	2498 (0.5%)	4286 (1.8%)
Header generator	0 (0%)	0 (0%)	2976 (0.6%)	2478 (1.0%)
Bitpacker	0 (0%)	0 (0%)	387 (0.1%)	334 (0.1%)
Total	**85 (14.2%)**	**63 (3.3%)**	**11915 (2.5%)**	**17185 (7.0%)**

5.5.1.3 Demonstrator set-up

The developed IP core has been mapped on a Xilinx Kintex UltraScale XCKU040-2FFVA1156E FPGA, mounted on a KCU105 evaluation board for validation purposes. In addition to the compression chain, the test-setup is completed with a MicroBlaze

embedded microprocessor to manage IP initialization and test behaviour; the necessary AXI infrastructure to interconnect the different modules; and a Direct Memory Access (DMA) module, which handles data transactions between the compression chain and the off-chip memory, where the input images are located prior to start the tests. The access to that external memory is done through a dedicated DDR4 memory controller. Input images are loaded into the external RAM from an SD card, by using the Xilinx *xilffs* library, which runs on the MicroBlaze soft processor and provides the necessary software functions to interact with that storage device. A specific AXI module is also used to manage modules initialization and configuration through AXI4-Lite interfaces. An overview of the whole test set-up is shown in Figure 5.14. The inclusion of these modules in the design implies a resources utilization overhead of 15% of LUTs, 9% of registers and 13% of dedicated memory, compared to the area consumed just by the compression chain.

The whole validation set-up also runs at a clock frequency of 125 MHz, except the DDR4 controller, which is fed with a dedicated clock at 300 MHz for high-speed accesses to the off-chip memory. The bottleneck of the processing chain is the predictor, due to the feedback loop implemented to support near-lossless compression. In this implementation, the predictor is able to generate a prediction residual $\delta(t)$ every 7 clock cycles. The other modules have a latency of 1 clock cycle including the hybrid encoder, since it always

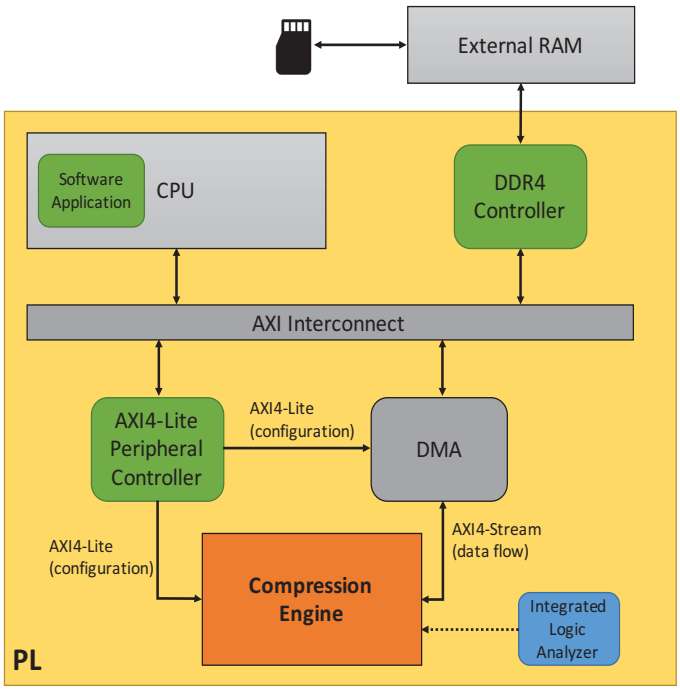


FIGURE 5.14: Validation set-up for the CCSDS 123.0-B-2 compliant IP

selects the high-entropy method for the AVIRIS scenes used for validation; otherwise, its latency will be variable, depending on the number of samples that are coded by using the low-entropy mode. The latency of the header generator, which has a sequential behaviour, is overlapped with the rest of the compression solution by executing both processes simultaneously. Therefore, the latency of the whole compression chain is 7 clock cycles to fully process an input sample. Taking into account that the presented solution has a linear behaviour, the throughput of the system is imposed by the predictor and it can be calculated as following reflected, considering as sample a 16-bits input pixel of an AVIRIS scene:

$$Throughput = \frac{1}{T_{predictor} \cdot \frac{1}{f}} = \frac{1}{7 \cdot \frac{1}{125 \cdot 10^6}} = 17.86 \quad (MSamples/s), \tag{5.1}$$

being $T_{predictor}$ the number of clock cycles taken by the predictor to generate a mapped residual $\delta(t)$ and f the system clock frequency, fixed to 125 MHz.

Although this throughput could prevent the use of this solution in real-time applications, it can be improved by placing multiple instances of the compression chain in parallel, since there is enough margin in terms of resources utilization, as previously reflected in Table 5.14. This mechanism is also robust against radiation effects, since an error in one image segment implies that it is lost but not the rest of the image, which can be recovered during the decompression. This scheme has been successfully evaluated in [192], obtaining different results in terms of compression performance and reconstructed image quality, depending on the partitioning pattern.

5.5.1.4 Evaluation of the HLS approach

Finally, Table 5.15 shows a comparison between the full HLS implementation of the CCSDS 123.0-B-2 standard and the VHDL implementation of the CCSDS 123.0-B-1 standard presented in Section 5.4. Results obtained by the VHDL implementation are presented for BIL order, for a more realistic comparison with the HLS approach [193]. As it can be observed, there is an increment in the logic resources utilization due to the introduction of the quantization feedback loop in the HLS approach, to be able to compress in near-lossless mode. The use of the new hybrid encoder instead of the sample-adaptive one described in Issue 1 of the standard also implies a higher logic resources usage. This is added to the fact that generally HLS implementations are not capable to map the developed model in

the available resources in an optimal way, as it is done in an RTL description. This area overhead is around +188% of LUTs and +15% of BRAMs. The high difference in the use of DSPs, around +79%, is derived from the quantizer inclusion, which uses multiplications. Besides, architectural modifications to support near-lossless compression also imply a reduction of -70% in terms of throughput.

The main strength of the presented work is the short development time, thanks to use the HLS design methodology, obtaining a functional model +75% faster (i.e., in less time) than following an RTL strategy. In this way, it is demonstrated the benefit of HLS to provide a behavioural description of the final system that, though it does not reach an optimized model in terms of area utilization and performance, it can served as a starting point for prototyping purposes. It is intended that future FPGA implementations of the CCSDS 123.0-B-2 near-lossless compression standard takes the presented approach as worst case, since it is expected that VHDL descriptions always obtain better results in terms of resources usage and throughput than one developed following an HLS design strategy.

Table 5.15: Comparison between CCSDS 123 VHDL and HLS implementations

Implementation	Development time (months)	Encoder	LUTs	FFs	DSPs	BRAMs	freq. (MHz)	Throughput (MSamples/s)
CCSDS 123.0-B-1 (VHDL)	24	Sample	5975	3599	13	74	152	59.4
CCSDS 123.0-B-2 (HLS)	6	Hybrid	17185	11915	63	85	125	17.86

5.5.2 Compression solution based on the CCSDS 123.0-B-2 standard for CHIME

The HLS model developed for CHIME is a tailored version of the predictor presented in the previous Section, implementing only the predictor features that achieve the best results in terms of compression ratio to meet mission goals. This optimal configuration was extracted after an exhaustive parameter tuning, whose main results are presented in Section 5.5.2.1.

The CCSDS 121.0-B-3 block-adaptive encoder introduced in Section 4.2 is used as entropy coder, since it achieves a compromise among compression demands and area utilization. In this way, a combined approach that uses both HLS and VHDL design methodologies is

employed to obtain a complete compression solution for this specific space mission in the available development time (i.e., 3-4 months).

5.5.2.1 Parameter tuning

The CCSDS 123.0.B-2 compression standard presents a great amount of configuration parameters that influence the compression performance. For the CHIME program, most of these parameters have been set to fixed values to simplify the hardware implementation, while at the same time optimizing the compressor performance according to the mission requirements. Some choices are imposed by the mission sensor, such as the spectral and spatial resolution per line, as well as the sample processing order in BIL. Other choices are motivated by the goal of obtaining a throughput as high as possible, such as the settings for the number of P previous bands used for the prediction, the local sum and the prediction mode. These decisions has been made after performing a parameter tuning to find the configuration that allows to maximize the compression ratio for the representative test vectors. These test vectors are based on AVIRIS scenes extended in the x-axis to achieve 2048 pixels per line, as expected for the CHIME sensor, and presenting different levels of cloud coverage.

The baseline configuration used for the parameter tuning is the one reflected in Table 5.16. Firstly, the predictor is configured in lossless mode, in order to identify the suitable parameter values to maximize CR. This process is done in an incremental way, determining first the predictor parameters that have a higher influence in the compression performance and then the ones related to the entropy coding stage.

The first test battery exercises the 8 possible combinations of local sum and prediction mode. Doing this, the combination of narrow neighbour-oriented local sum with reduced prediction mode is identified as the best one, obtaining maximum CRs of 4.0378.

After fixing these parameters, then it was turn for analysing the impact of the P value in the CR. Since it is demonstrated that values higher than 3 do not provide an improvement in terms of compression performance [181], the tuning has been done for P values from 0 to 3, as shown in Figure 5.15. Although the results for $P = 1$ are slightly better than for $P = 3$, the latter is finally selected since results for $P = 1$ are outliers probably due to image construction by mirroring In addition, it is expected a higher image quality after decompression by employing $P = 3$ than $P = 1$.

TABLE 5.16: CCSDS123.0-B-2 predictor configuration for CHIME parameter tuning

Parameter	Value
Image parameters	
Columns, N_x	2048
Lines, N_y	2649
Bands, N_z	224
Dynamic Range, D	16
Encoding Order	BIL
Predictor parameters	
Bands for Prediction, P	3
Local Sum Mode	Narrow Neighbour-Oriented
Prediction Mode	Full
Register size, R	64
Weight Resolution, Ω	16
Weight Update Scaling Exponent Interval, t_{inc}	64
Weight Scaling Exponent Interval Initial, v_{min}	6
Weight Scaling Exponent Interval Final, v_{max}	9

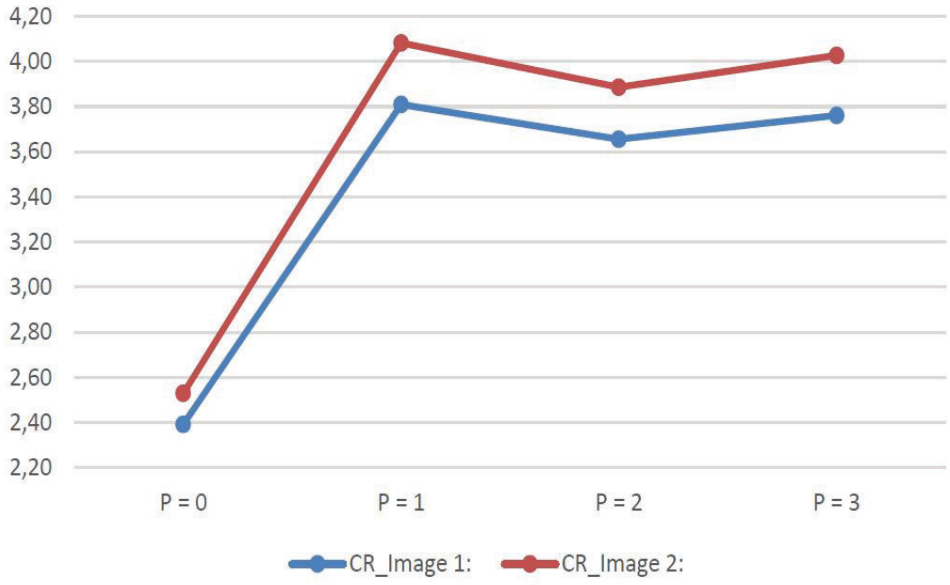


FIGURE 5.15: Compression ratio as a function of the P value

Then, suitable values for v_{min}, v_{max} and t_{inc} are obtained. Those optimal values are 0, 3 and 2048, respectively, though the latter does not affect performance. Under this configuration, a maximum CR of 4.3776 is obtained in lossless compression. Regarding the register size R, improvements are not observed for $R \geq 43$, so 48 is finally selected, as it is the closer byte boundary.

Weight resolution Ω has been swept from 8 to 18 in steps of two. In addition, the maximum allowed value (i.e., 19) is also included in the tests. Results are reflected in Figure 5.16 for

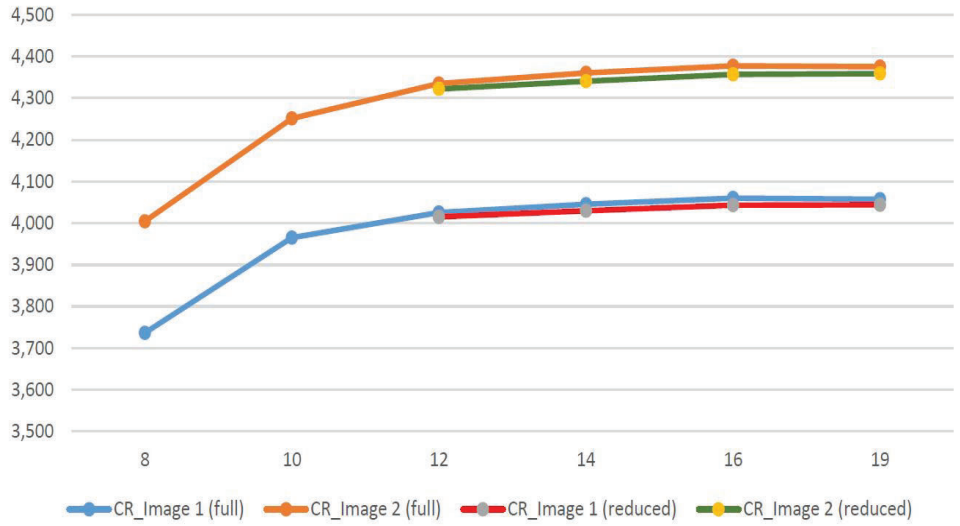


FIGURE 5.16: Compression ratio as a function of the Ω value

two of the employed test vectors and considering prediction in full and reduced mode for each one of them. As it is observed, best results under full mode are obtained for $\Omega = 16$, while $\Omega = 19$ provides higher CRs when reduced prediction mode is selected. Since the CR improvement for increasing Ω from 16 to 19 is not significant (from 4.3573 with $\Omega = 16$ to 4.3589 with $\Omega = 19$), 16 is selected as final value for the sake of hardware simplicity.

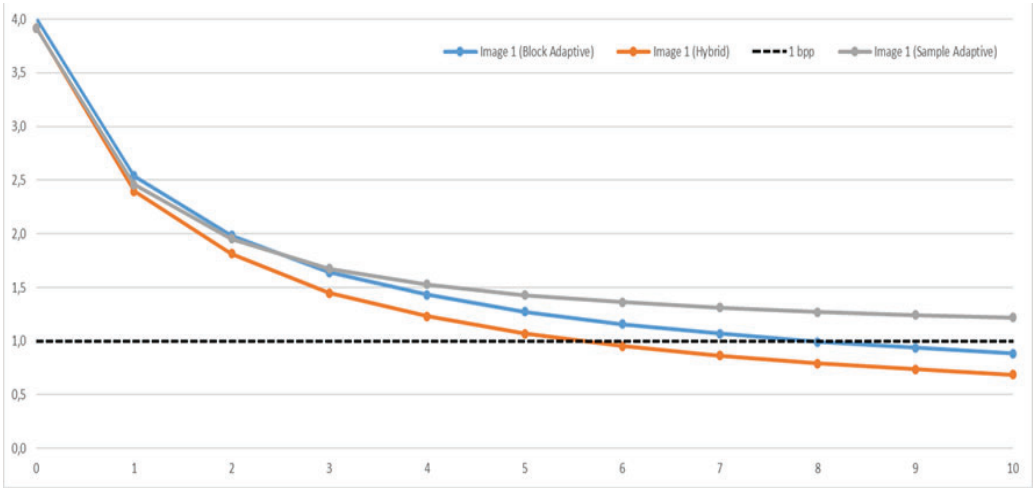
Regarding sample representative parameters, our findings demonstrate that optimal values depend on the target image nature and the error introduced under near-lossless compression. As an example, for a sample representative resolution Θ equal to 4, the optimal offset value ψ_z is dependent on the absolute error limit a_z, as reflected in Table 5.17. If lower resolutions are used, these values are adjusted dividing them by 2 for each bit of resolution removed. In addition, it has been demonstrated that suitable values of the damping ϕ_z depends on the chosen value of the resolution Θ, being this relationship the one expressed by Equation 5.2.

$$\phi_z = 2^{\Theta-2} \tag{5.2}$$

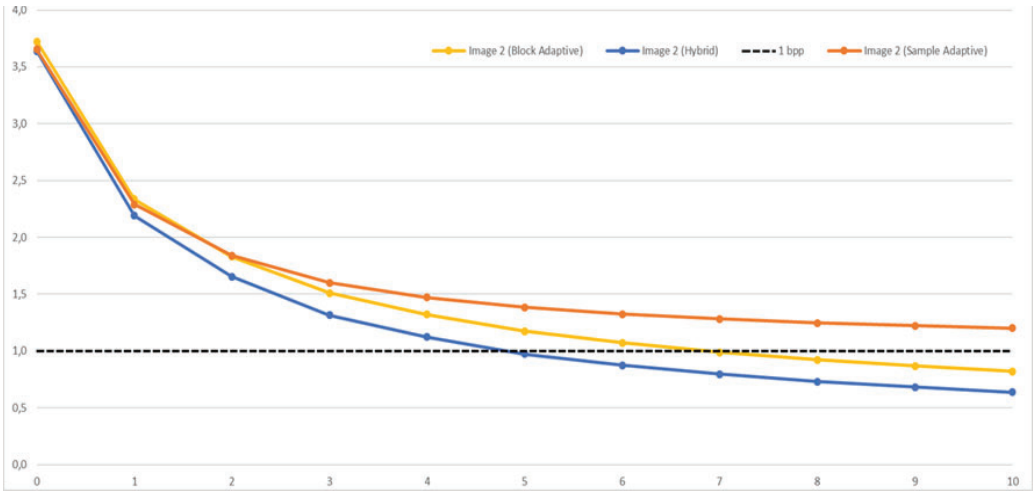
TABLE 5.17: Optimal offset and damping values depending on the absolute error limit a_z, when $\Theta = 4$

a_z	**1**	**2**	**3**	**4**
Damping, ϕ_z	4	4	4	4
Offset, ψ_z	3	4	6	7

Finally, the compression performance of the three available options for the entropy coding stage is compared. As it is shown in Figure 5.17, the hybrid encoder outperforms the other two encoding alternatives, specially for high absolute error limits ($a_z \geq 4$). This is because mapped prediction residuals $\delta(t)$ tend to be closed to 0 at the same time the maximum error limit defined during the prediction increases, exploiting in this way the low-entropy mode introduced in the hybrid encoder. The sample-adaptive encoder is clearly the worst option, since it is not able to achieve a CR lower than 1 bpp from a theoretical point of view. Differences in terms of achieved CR between the hybrid and the block-adaptive encoder are reduced for high absolute error values, since the latter incorporates a *zero-block* option.



Test Vector 1



Test Vector 2

FIGURE 5.17: Comparison of the encoder performance for CHIME test vectors.

For this reason, the block-adaptive encoder has been selected for this implementation, since results are considered acceptable. Besides, a VHDL implementation, previously presented in Section 4.2.2, is available at the beginning of the CHIME program, which is widely verified compared to the developed hybrid encoder. In addition, the block-adaptive encoder ensures a throughput of 1 sample per clock cycle, providing real-time capabilities that are not guaranteed when using the hybrid encoder under low-entropy mode. The combination of parameter values that achieve best results for the block-adaptive encoder in terms of CR is the one summarised in Table 5.18.

TABLE 5.18: Parameter values for the block-adaptive encoder to achieve the highest CR

Parameter	Value
Block size, J	4
Codeset	Basic
Reference sample interval, r	4096

5.5.2.2 System development

The block diagram of the developed approach for the CHIME pre-development phase is shown in Figure 5.18. It is mainly comprised by two stages: a tailored version of the CCSDS 123.0-B-2 predictor and the block-adaptive encoder presented in Section 4.2. In addition, a third-party AXI-to-AHB bridge has been included to preserve the original configuration interface of the block-adaptive encoder.

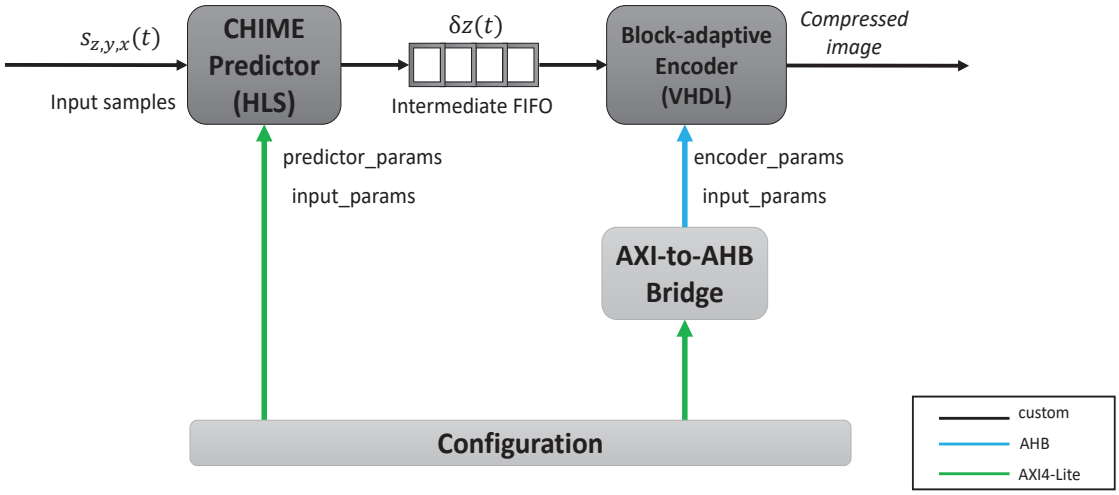


FIGURE 5.18: Block diagram of the proposed compression approach for CHIME pre-development phase

Since the target device is a Kintex UltraScale XCKU040 FPGA, the predictor can be modelled using Xilinx Vitis HLS tool. As it is observed in Figure 5.19, this module has been developed with custom interfaces, based on a simple handshaking protocol to be easily connected with the block-adaptive entropy coder. To do that, specific directives have been defined to the Vitis HLS tool, in order to synthesize I/O ports as ad-hoc interfaces, avoiding the automatic definition of AXI signals. The predictor accesses to the input FIFO, external to the compressor, to receive a new sample as soon as the internal pipeline allows, ensuring that there are not collisions with the processing of the previous one. Since this module generates the CCSDS header fields related to the prediction, a dedicated output, denoted as *Is_header_out*, is defined to send this partial header to the block-adaptive encoder, which receives it and appends the encoder fields.

Several predictor parameters are fixed, taking into account the results of the parameter tuning presented in Subsection 5.5.2.1. The selected configuration is summarised in Table 5.19. With respect to the quantizer settings, only band-dependent absolute error limits are supported, while sample representative parameters (i.e., the offset ψ_z and the damping ϕ_z) are fixed as band-independent for simplicity. The absolute error is implemented with a resolution of 6 bits, which allows a range of values between 0 and 63, runtime configurable. The sample representative settings are also fixed, with the exception of the band-independent sample representative offset ψ_z. Lossless compression is enabled by setting the absolute error limit A to 0. For disabling the samples reconstruction stage, it is enough with setting the sample representative offset ψ_z or the resolution Θ to 0. In addition to these two parameters, the number of lines in the image N_y is also configurable, and a bypass option is provided to avoid the compression process. Runtime parameters

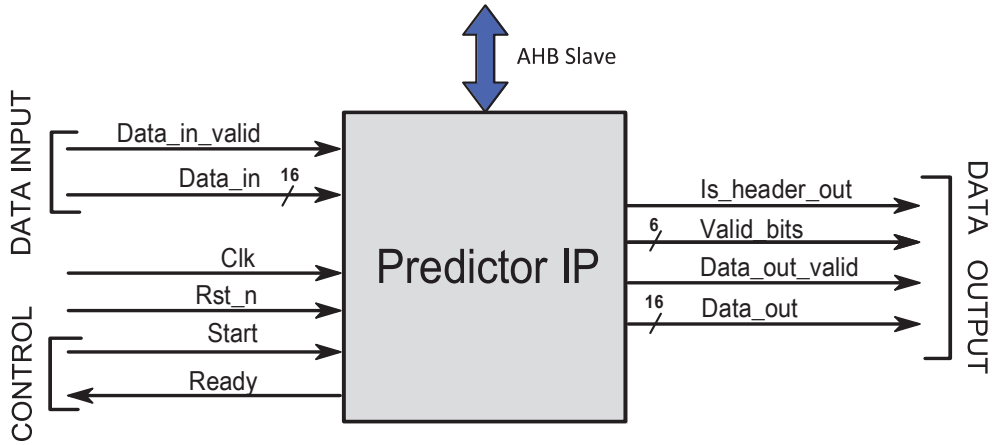


Figure 5.19: CHIME predictor - Overview

are reflected in Table 5.20 and they are configured through a dedicated AXI4-Lite slave interface.

Table 5.19: CHIME predictor - Constant parameters

Parameter	**Value**
Image parameters	
Columns, N_x	2048
Bands, N_z	256
Dynamic Range, D	16
Encoding Order	BIL
Predictor parameters	
Bands for Prediction, P	3
Local Sum Mode	Narrow Neighbour-Oriented
Prediction Mode	Reduced
Register size, R	48
Weight Resolution, Ω	16
Weight Update Scaling Exponent Interval, t_{inc}	2048
Weight Scaling Exponent Interval Initial, v_{min}	0
Weight Scaling Exponent Interval Final, v_{max}	3
Sample Representative Resolution, Θ	2
Sample Representative Damping, ϕ_z	1
Absolute Error Bit-Depth, A_z	6

Table 5.20: CHIME predictor - Runtime configurable parameters

Parameter	**Allowed values**	**Bits taken**
Number of lines, N_y	[1:256]	8
Bypass	[0,1]	1
Band-dependent Absolute Error, A	[0:63]	6
Sample Representative Offset, ψ_z	[0:3]	2

5.5.2.3 Experimental results

A pipelining strategy has been followed in the prediction stage, overlapping internal prediction stages in absence of data dependencies and thus improving throughput. The selection of the narrow neighbour-oriented local sum together with the reduced prediction mode benefits this approach, since data dependencies with the sample at the left of the current one in the same band $s_{z,y,x-1}$ are removed, which is a main constraint under BIL order. However, the HLS tool does not properly analyse data dependencies among internal stages so, though some tasks are performed in parallel (i.e., the processing of the next sample is started without finishing the prediction of the current one), maximum theoretical

throughput is not achieved. The optimized predictor version is able to process an input sample (16 bits precision) each 11 clock cycles at a frequency of 100 MHz. This feature is still far of the final target of 1 clock cycle per sample for CHIME, but it is expected to be achieved by a purely VHDL description of the CCSDS 123.0-B-2 predictor. In any case, this model allows to validate the behaviour of the proposed compression chain as a prototype at early stages of the mission workflow, and it can be considered as a worst case for a future VHDL implementation.

Results in terms of resources utilization are shown in Table 5.21. As it can be observed, the limiting factor is the memory usage of the predictor, which is clearly conditioned by the input image size. This is because all the preprocessed samples of the previous spectral line ($N_x \cdot N_z$ samples) should be stored to compute the prediction of the current spectral line. For the CHIME implementation, a total of 144 BRAMs are used just to store this memory. Nonetheless, even when the predictor has been designed by means of HLS techniques, hardware occupancy is considered acceptable, since the design can be mapped in the target FPGA including mitigation strategies based on modular redundancy.

Table 5.21: CHIME IP - Resources utilization on Xilinx Kintex UltraScale XCKU040

	36Kb BRAM	DSP48E	Registers	LUTs
Predictor (HLS)	167 (27.8%)	33 (1.7%)	7388 (1.5%)	9737 (4.0%)
Block-adaptive Encoder (VHDL)	4 (0.7%)	4 (0.2%)	1725 (0.4%)	4944 (2.0%)
Total	**171 (28.5%)**	**37 (1.9%)**	**9113 (1.9%)**	**14681 (6.0%)**

5.5.2.4 Demonstrator set-up

Figure 5.20 shows the demonstrator set-up used for validation purposes [194]. In this picture, the main hardware components of the board and the used external interfaces are highlighted. The main elements of the demonstrator are summarised next:

- **DUT board.** The KCU105 evaluation board from Xilinx has been used to implement the Design-Under-Test (DUT) [195]. This board mounts a Kintex UltraScale XCKU040-2FFVA1156E FPGA, where the design is loaded for its validation. This board contains all the hardware needed, excepting the interfaces to send and receive the images. To implement the data interfaces, extra hardware is added by using a FMC connector.

- **Test Board.** This mezzanine board has been mounted in the DUT board. This board allows for bridging the USB 3.0 host (connected to the Monitoring and Control

PC) to a FIFO bus. The FPGA in the DUT board interfaces this FIFO to receive and transmit the image samples. The USB 3.0 to FIFO bridge allows to send the images and receive the compressed bitstream at a maximum data rate of 2.1 Gbps.

The Monitoring and Control PC includes different software tools to perform several functions, such as programming the FPGA through the JTAG link; a GUI to control and monitor the test execution; reading the files containing the images to be sent and controls the USB interface to transmit this information to the test mezzanine; recovering the compressed data from the USB link, saving them into an output file to be post-processed; and automation of all the tasks needed to run one test and to verify compression results.

Validation has been initially done with small images to speed up the debugging process. After that, two test vectors have been used, compressing both in lossless mode and additionally one of them under lossy compression [194]. In all the cases, the output files obtained by the demonstrator are identical bit-by-bit to the ones produced with the CNES software [141]. In addition, the throughput has been measured, obtaining the same results expected from the theoretical analysis (i.e., 11 clock cycles per sample), which correspond to the predictor throughput, the critical stage of the compression chain. Therefore, the execution of these tests has been considered successful and, consequently, the HLS-VHDL combined implementation has been probed to be in accordance with the initial mission requirements.

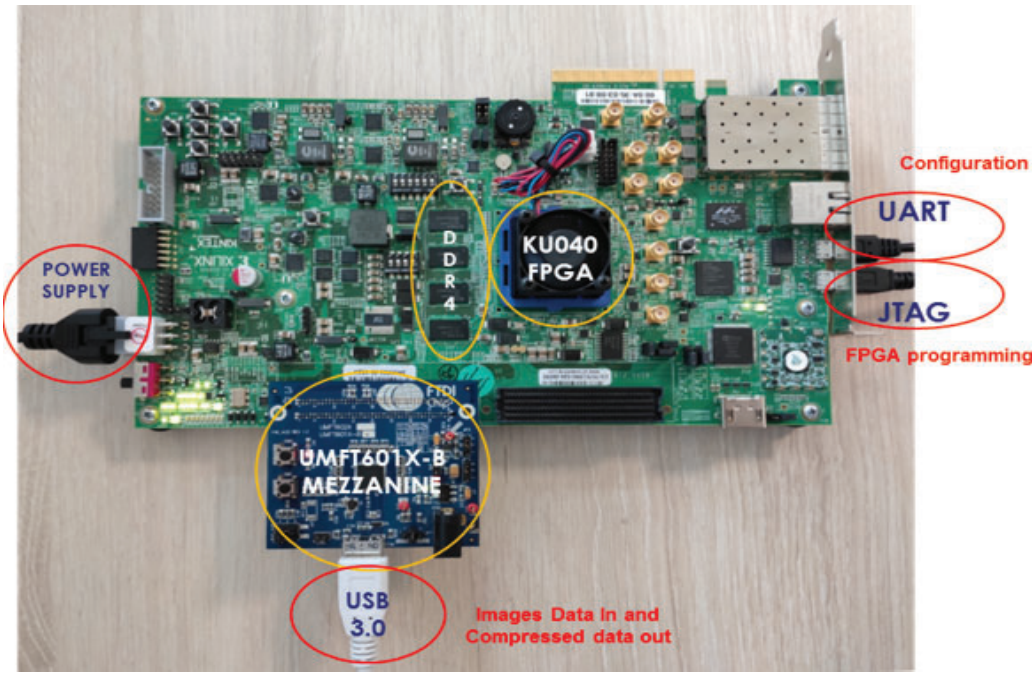


FIGURE 5.20: Validation set-up for CHIME (extracted from [194])

5.6 Video compression solution based on the CCSDS 123.0-B-2 algorithm

Since video sensors are integrated on-board satellites little by little, the space industry is looking for low-complexity compression solutions that can be efficiently implemented on embarked hardware. In this way, video acquisition is not compromised by neither the on-board storage capacity nor the limitations in the downlink bandwidth with ground.

The VIDEO project proposes the development of a next-generation instrument for Earth observation, capable to perform high-resolution video monitoring on an extremely wide scene, with the purpose of recognizing and tracking objects. The processing finishes by compressing the useful data in order to minimize the downloaded information to ground [63].

The contribution to this project presented in this Section is related to the compression of the video acquired by the VIDEO instrument. As commercial video encoders used on ground are complex to be embarked on satellites, even if tailored solutions are considered, an alternative standpoint is presented in this Thesis. Concretely, two different solutions are proposed, depending on the video nature (i.e., panchromatic or RGB). Both approaches take the CCSDS 123.0-B-2 algorithm as baseline, which is adapted for video compression. In this way, a reusable solution is proposed to compress not only multi- and hyperspectral images, but also panchromatic and RGB video with a unique processing core, reducing hardware occupancy and power consumption. This kind of versatile compression solutions could be useful for next-generation space missions that embark both spectroscopic and video sensors, sharing the compression core and thus optimizing on-board computational resources.

5.6.1 Application of the CCSDS 123.0-B-2 algorithm for panchromatic video compression

5.6.1.1 Proposed approach

As it has been aforementioned in Section 3.4, the CCSDS 123.0-B-2 algorithm is conceived for multi- and hyperspectral image compression. However, it can be easily adapted to perform panchromatic video compression, since this kind of video just has a wide spectral

component in the visible range. To do this, the spectral dimension z defined in the standard is replaced by the temporal domain t, using for the inter-prediction the neighbouring samples in the P previous frames, instead of the vicinity in the previously processed spectral bands, as it is done originally in the local differences calculation of the prediction stage. This transformation is reflected in Figure 5.21. In addition, intra-prediction is done by calculating the local sum in the current spatial frame, as it was explained in Section 3.4.1. This intra-prediction, though limited in comparison with the one used by commercial video encoders, can be considered as a low-complexity alternative for video sequences with mainly global movement, as it is expected in Remote Sensing applications. This prediction-based solution, though is not able to achieve compression ratios in the same order than transform-based approaches, such as H.264 and H.265 codecs, is capable of offering a fine-grain control of the losses introduced in the compression process, at the same time that eases the hardware implementation.

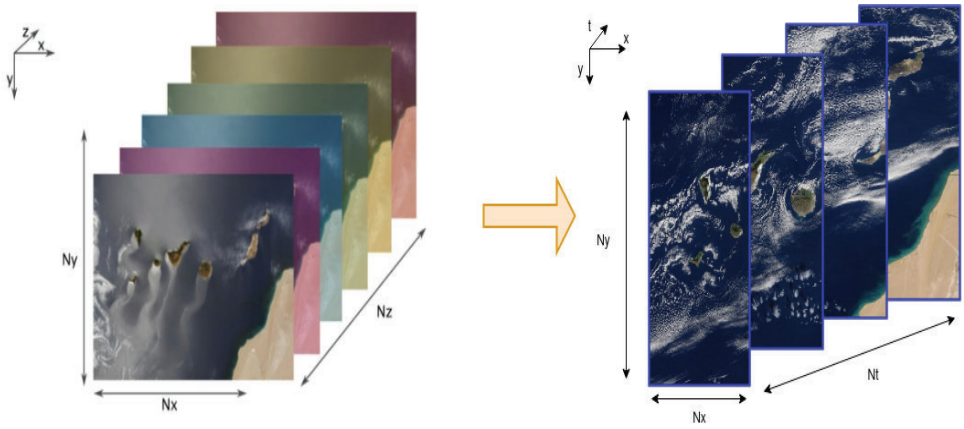


FIGURE 5.21: Conversion from the spectral to the temporal domain

5.6.1.2 Experimental results

A dataset comprised by 6 video sequences has been used for carrying out the quality assessment of the CCSDS 123.0-B-2 algorithm for panchromatic video compression. Concretely, 4 of these scenes are from Planet SkySat database [22] and the other 2 have been acquired in our facilities using an IDS uEye sensor [196]. All of them present a relatively low global movement with different levels of local movements, trying to be representative of a real Remote Sensing scenario.

All the video sequences used in the experiment are comprised of 150 frames and are stored using 8 bits per pixel. Each frame corresponding to the *Burj Khalifa* video from SkySat

has 1920x1080 spatial pixels. The frames corresponding to the other 3 videos from SkySat, *Calbuco Volcano*, *Turkey* and *Las Vegas*, have a spatial resolution of 1280x720 pixels. Finally, the spatial resolution of the videos collected by the IDS uEye sensor is 1280x1024 pixels.

Since the algorithm has a high quantity of configurable parameters that influence in the achieved CR, the solution must be characterized to find the optimal configuration for panchromatic video compression. A fine-grain parameter tuning is done, fixing first the predictor parameters, and afterwards, the ones related to the hybrid encoder.

The selected compressor configuration is summarised in Table 5.22, keeping in mind the goal of maximizing CR without degrading in excess the reconstructed video quality.

Regarding the selected prediction mode, it was observed that the *reduced* mode provides better results for static video sequences (i.e., with a reduced local movement), while the *full* mode is preferable for scenes with a considerable presence of local movement. These results were expected, since the *full* prediction mode considers more neighbouring samples for its computation than the *reduced* one, improving in this way the compression under high local movement. In both cases, the *wide neighbour-oriented* local sum outperforms

Table 5.22: Relevant compressor parameters

Parameter	**Value**	**Description**
Video parameters		
D	8	Bit-width of input pixels
ENDIANNESS	0	Little Endian
IS_SIGNED	0	Unsigned samples
Predictor parameters		
P	3	Previous bands used for prediction
R	48	Register size
Ω	16	Weight resolution
v_{min}	0	Weight update initial value
v_{max}	3	Weight update final value
t_{inc}	64	Weight update change interval
Θ	2	Sample representatives resolution
ϕ	1	Damping
ψ	1	Offset
D_a	7	Absolute error bit-depth
A	[0:32]	Band-independent absolute error
Encoder parameters		
U_{max}	32	Unary Length Limit
$\gamma*$	4	Rescaling Counter Size
γ_0	1	Initial Count Exponent

the other available options, though the penalty when the *narrow neighbour-oriented* one is selected is around the 10%-15%. This latter option may be preferable to prioritise the throughput instead of the CR in hardware implementations, since it reduces data dependencies with adjacent samples in the same frame. The impact of varying the P value has been also evaluated, concluding that values higher than 3 do not provide a compression improvement, as it is shown in Figure 5.22 and as it was also concluded for hyperspectral image compression [179, 181]. At the same time, employing a low P value simplifies hardware implementation, since some memory elements are dependent on this parameter.

Other critical parameters, such as the ones involved in the samples representatives calculation (i.e., the resolution Θ, the damping ϕ and the offset ψ), are also in consonance with the optimal values detailed in [181], when the compressor is characterized with multi- and hyperspectral images.

Results in terms of compression ratio versus reconstructed video quality, measured by using the PSNR, are summarised in Figure 5.23. Only absolute error values are reflected for simplicity, because both the compression ratio and the PSNR are similar when relative error limits are used or when both are combined, since the algorithm always take the minimum value between both options.

The absolute error is used as frame-independent (i.e., the same error is applied to all the frames in the video sequence) and its value is swept from 0 (lossless) to 31, by defining a

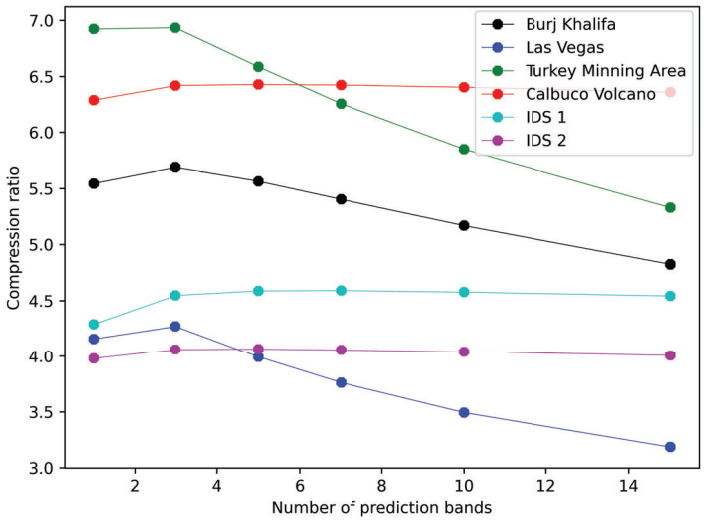


FIGURE 5.22: Panchromatic compression - Influence of P value in the compression ratio under lossless mode

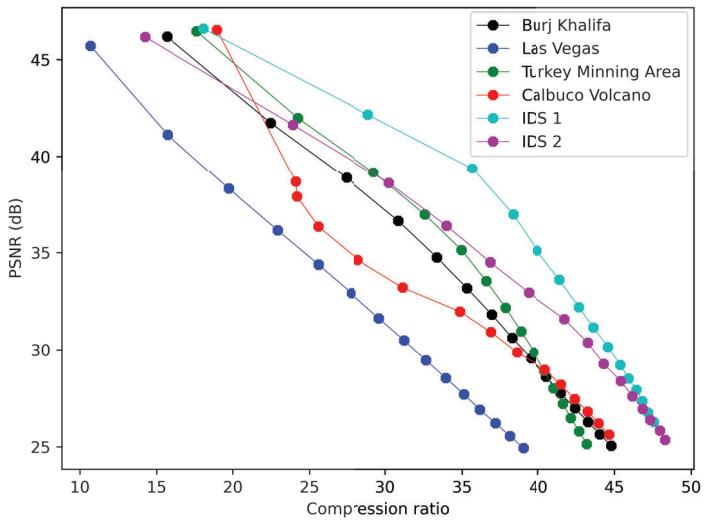
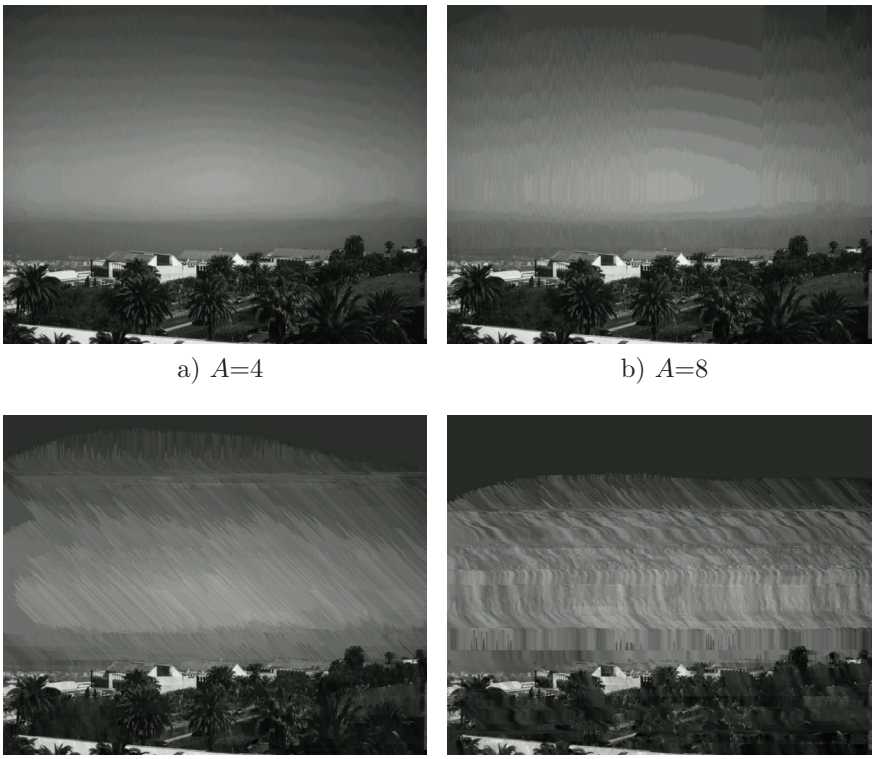


FIGURE 5.23: Panchromatic compression - Relationship between compression ratio and video quality

maximum error bit-depth of 5. It is observed that for compression ratios up to 25 (i.e., 0.32 bits per pixel), the reconstructed video obtains PSNR values over 35 dB, providing sequences without any visual impact because of compression effects. The worst results are obtained for *Las Vegas* sequence, the one with highest local movement within the used dataset. It is also remarkable the difference in terms of compression performance for the video samples with higher and lower amount of local movement, obtaining the latter CRs around 50 (i.e., approximately 0.16 bits per pixel), with a PSNR equal to 25 dB.

Figure 5.24 shows a frame of the *IDS 1* sequence, applying different absolute error limits. As it can be observed, a clear degradation is not appreciated from a visual point of view until the bottom-left frame ($A = 16$), with a CR around 40 (i.e., 0.2 bits per pixel) and a PSNR equal to 37 dB. In the first and second frame ($A = 4$ and $A = 8$, respectively), just some degradation is distinguished at the top of the picture. Nevertheless, object shapes are preserved even when $A = 32$, obtaining in this case a CR around 47 with a PSNR equal to 27 dB.



a) A=4 b) A=8

c) A=16 d) A=32

FIGURE 5.24: Visual aspect of the decompressed frames from the *IDS 1* video sequence when compressing it with different maximum errors values

5.6.2 Adapting the CCSDS 123.0-B-2 algorithm to compress RGB video

5.6.2.1 Proposed approach

5.6.2.1.1 Using a single CCSDS-123 compression core for each color channel

When the CCSDS 123.0-B-2 algorithm is used to compress hyperspectral images, it has been demonstrated that highest RD ratios are obtained for hyperspectral images with a large number of bands [197]. This happens due to the fact that when the number of bands increases, the spectral channels at which subsequent bands are sensed get closer and hence, adjacent bands start to be strongly correlated. Accordingly, each sample can be accurately predicted using the values of the same pixel in previous bands. However, when compressing multispectral images with higher spectral distances between consecutive bands, the spectral prediction is not so efficient.

This is also applicable to Remote Sensing applications focused on video acquisition, where a high spatial resolution results in video sequences with a reduced global movement between subsequent frames, and where the moving objects are very small in comparison to the spatial dimension of the captured scene. Due to this, the correlation between subsequent frames for the same color channel in an RGB video could be strong, while the correlation between the 3 color channels for a single frame would not be so high. Hence, each of the color channels could be treated as an independent gray-scale video and can be independently compressed using a CCSDS 123.0-B-2 compressor, as it is shown in Figure 5.25.

Following this procedure, both panchromatic and RGB video sequences can be compressed using the same CCSDS 123.0-B-2 processing core, without carrying out any further modification, just by replacing the spectral dimension z by the temporal dimension t, and independently compressing each color channel as a gray-scale video.

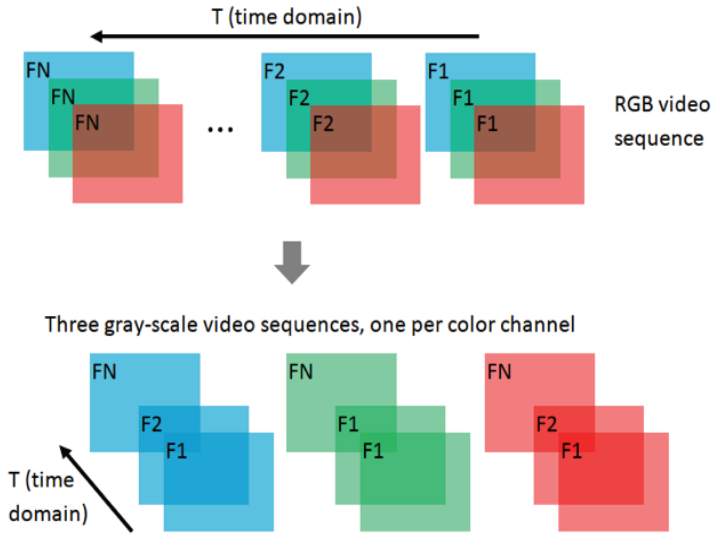


FIGURE 5.25: Compression strategy for RGB video, treating each color channel as independent frames

5.6.2.1.2 Transformation to the YCbCr domain

Although the previous strategy is viable for RGB videos, its compression performance is limited. So, further optimizations can be added to increase the overall compression performance. In order to keep the compression solution fully compliant with the CCSDS 123.0-B-2 standard and to make it flexible so it can be adapted to multiple requirements, each optimization added here on is implemented into the entire compression chain as an individual stage, which can be optionally used attending to the necessities of the targeted

application, resulting in three different compression alternatives. Figure 5.26 graphically describes the entire compression chain, including these three compression alternatives and the individual blocks that encompasses them, which are also detailed next:

- **RGB to YCbCr spectral transformation.** This block carries out the spectral transformation of each video frame from the RGB to the YCbCr color space. By doing so, while each video frame still has three spectral channels, most of its spatial information is concentrated into the first one, the luma (Y), which represents the brightness of the scene, while the chroma bands (Cb and Cr) stores the color information. The conversion is done by applying Equations 5.3, 5.4 and 5.5, as it is described in the Recommendation ITU-R BT.709-6 [198]. This kind of transformation is commonly used in most of the standard video compressors, such as the H.264 specification [115], to introduce a higher level of losses in the Cb and Cr components, since they store information less perceptible to the human eye, allowing in this way to increase CR without degrading reconstructed video quality. Similarly, a higher error limit can be applied in the CCSDS 123.0-B-2 compressor when using it to individually compress the Cb and Cr frames, while compressing the Y frames with a lower error or even in lossless mode. This preprocessing stage is graphically described in the second row of Figure 5.26 and it is only executed in the compression alternatives 2 and 3.

$$Y = 0.2627 \cdot R + 0.6780 \cdot G + 0.0593 \cdot B \tag{5.3}$$

$$C_B = \frac{B - Y}{1.8814} \tag{5.4}$$

$$C_R = \frac{R - Y}{1.4746} \tag{5.5}$$

- **Spatial subsampling of the Cb and Cr bands**. In addition to the previous step, a spatial subsampling of the Cb and Cr channels can be applied by using a bilinear interpolation, reducing in this way their width and height. This strategy is also commonly applied in most of the commercial video compressors in order to increase the overall CR achieved at the cost of introducing some spatial blurring in the Cb and Cr channels, without introducing significant distortions from a visual

point of view. One further advantage of applying this preprocessing stage is that the computational burden of the subsequent compression stages for the Cb and Cr channels is reduced, decreasing at the same time the resources utilization needed on the final hardware implementation. In the proposed approach, the size of the Cb and Cr channels has been reduced by a factor of 2 in both their width and height dimensions, thus lessening the overall amount of data to be further processed from these channels by a factor of 4. By introducing this stage in the compression chain, it is also possible to adjust the global latency, being able to compress the luma channel with a compression instance, at the same time that both chroma components are compressed one after the other by using an additional compression core. This preprocessing stage is graphically described in the last row of Figure 5.26 and it is only executed in the last compression alternative (i.e., alternative 3).

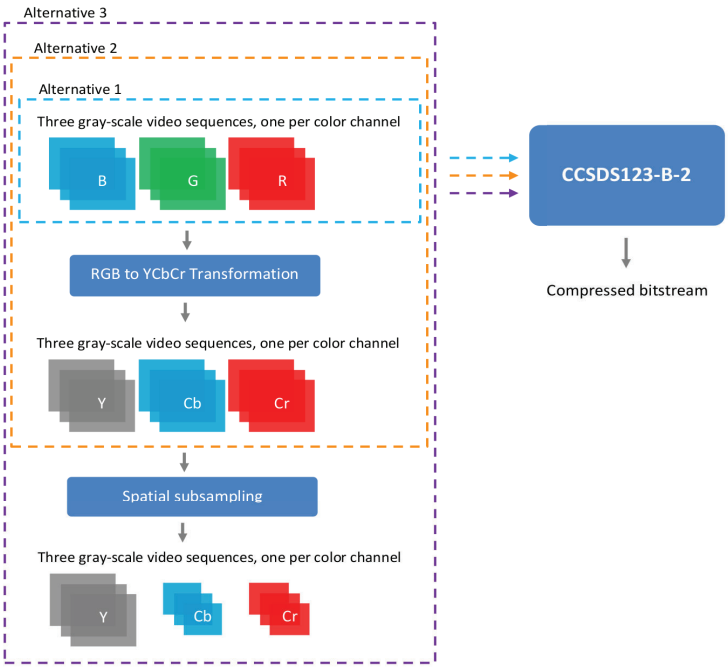


FIGURE 5.26: Full processing chain for RGB video compression

5.6.2.2 Experimental results

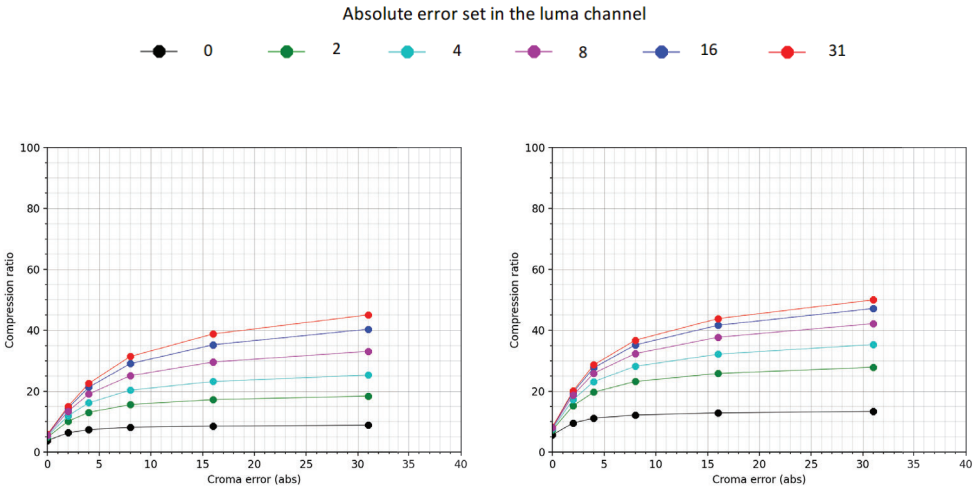
4 video sequences have been used for carrying out the quality assessment of the CCSDS 123.0-B-2 standard for RGB video compression. The first two sequences have been acquired in our facilities using an IDS uEye sensor [196], providing scenes comprised by 100 frames that are stored using 8 bits per pixel and with a spatial resolution of 1280x1024 pixels.

These scenes were captured trying to simulate a Remote Sensing application, where an object with a constant movement is appreciated, while the rest of the scene remains almost static. Two additional sequences extracted from the Stanford Drone Dataset have been also employed [199], known as *Death Circle* and *Nexus* sequences. These videos have been also trimmed to 100 frames, with a spatial resolution of 1400x1904 pixels and a precision of 8 bits per pixel. Unlike video sequences acquired in our facilities, videos from Stanford Dataset present a high local movement, helping to characterize the behaviour of the compression solution under demanding scenes in terms of displacement.

After an exhaustive parameter tuning to characterize the different CCSDS 123.0-B-2 compression parameters, the same configuration used for panchromatic video compression is selected, which has been summarised in Table 5.22, since it is also the one that provides the best results in terms of RD ratio for RGB videos. Full prediction mode has been selected, since it was identified as the best option not only for eminently static scenes, but also for the ones with a considerable local movement. This is because it considers a pixel vicinity for the prediction in the current frame, in addition to the pixel at the same position than the current one but in P previous frames, which is the only one employed under reduced mode. Regarding the local sum method, the wide neighbour-oriented local sum outperforms the other available options.

A sweep is firstly performed to analyse the impact in the CR of the applied absolute errors on the luma and chroma channels, using values in the range $0 \leq A \leq 31$ in steps of power-of-two. This study has been performed for Alternatives 2 and 3, since both of them apply RGB to YCbCr spectral transformation, and the obtained results are shown in Figures 5.27 and 5.28, respectively. The case $A = 0$ represents lossless compression, achieving CRs around 5 for IDS videos, and between 8 and 10 for Stanford sequences under Alternative 2. Compression ratio in lossless mode is improved under Alternative 3, reaching up to 10 for IDS videos, and between 9 and 12 for the Stanford sequences.

As it is shown in Figure 5.27, the chroma errors slightly increase compression ratio for all the images in the dataset, while in Figure 5.28 is appreciated that compression performance stalls when high error values are applied to chroma components, specially when low absolute error values are applied to the luma. This is due to the fact that the amount of Cr and Cb data to be compressed is just the half than the Y data due to the spatial subsampling. Additionally, Cb and Cr channels present lower entropy and can be more efficiently compressed even with low maximum errors, resulting in less compressed



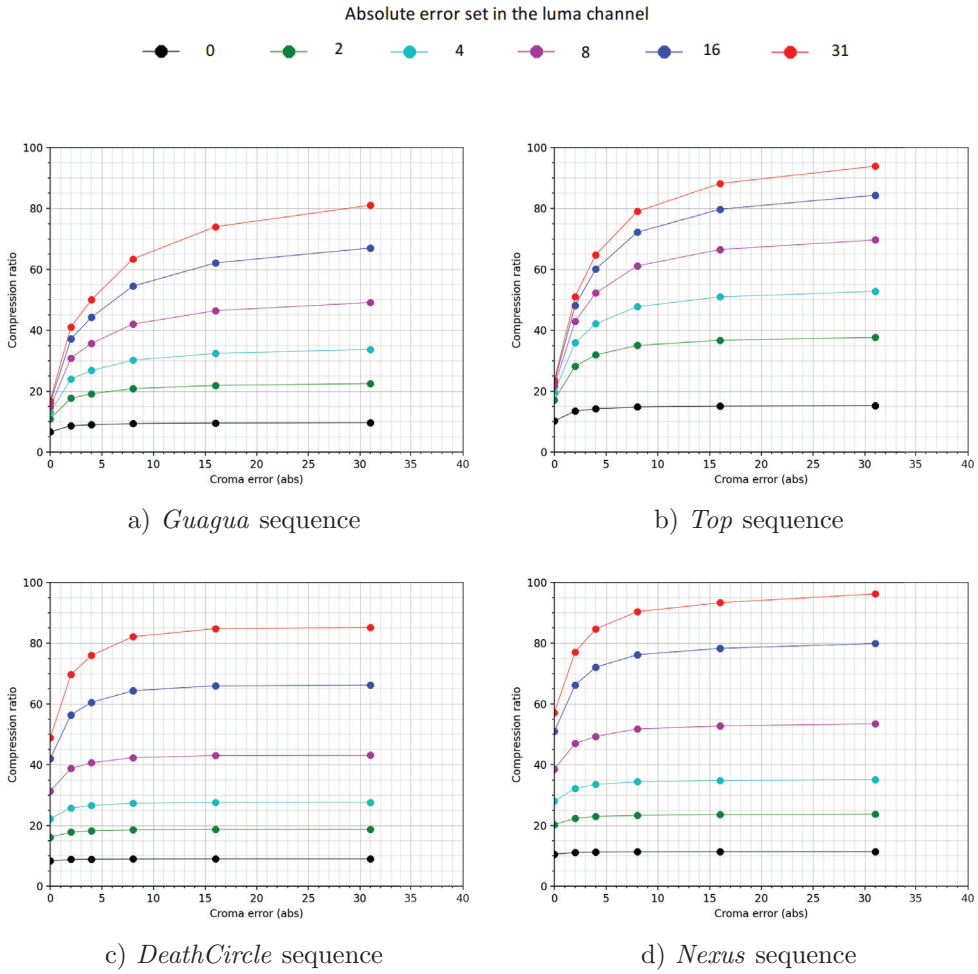
a) *Guagua* sequence b) *Top* sequence

c) *DeathCircle* sequence d) *Nexus* sequence

FIGURE 5.27: Relationship of the the chroma and luma errors with the compression ratio (Alternative 2)

information and becoming negligible in comparison to the part of the compressed bitstream corresponding to the Y channel.

On the other side, the luma error emerges as the key value to increase the compression performance, allowing maximum compression ratios for the IDS sequences between 37 and 94 when $A_Y = 2$ and $A_Y = 31$, respectively, under Alternative 3. Similar results are obtained for the Stanford sequences, reaching maximum compression ratios between 24 and 96 when $A_Y = 2$ and $A_Y = 31$, respectively. These results are obtained for the four video sequences by fixing both chroma errors to the maximum allowed in these tests (a value of 31). These results are considerably worse under Alternative 2, obtaining maximum compression ratios for the IDS sequences between 27 and 50 when $A_Y = 2$ and $A_Y = 31$, respectively, and between 20 and 54 for the Stanford videos when $A_Y = 2$ and $A_Y = 31$, respectively.



a) *Guagua* sequence

b) *Top* sequence

c) *DeathCircle* sequence

d) *Nexus* sequence

FIGURE 5.28: Relationship of the the chroma and luma errors with the compression ratio (Alternative 3)

Results in terms of compression ratio versus reconstructed video quality, measured in terms of PSNR, are shown in Figures 5.29, 5.30 and 5.31 for Alternatives 1, 2 and 3, respectively. In these tests, only frame-independent absolute errors are used. In the case of Alternative 1, the same absolute error is applied to each color channel, since in this approach the RGB to YCbCr spectral transformation is not performed, compressing directly RGB video in raw format. Purely lossless situation ($A_{Y,Cb,Cr} = 0$) is not shown for Alternative 1, since PSNR tends to infinite. As it can be seen for all the experiments performed, the decompressed video quality decreases as the maximum error fixed for the luma and chroma channels increases. This tendency is more notable in the case of the Stanford sequences when applying low absolute errors to the luma channel ($A_Y \leq 4$), under Alternatives 2 and 3.

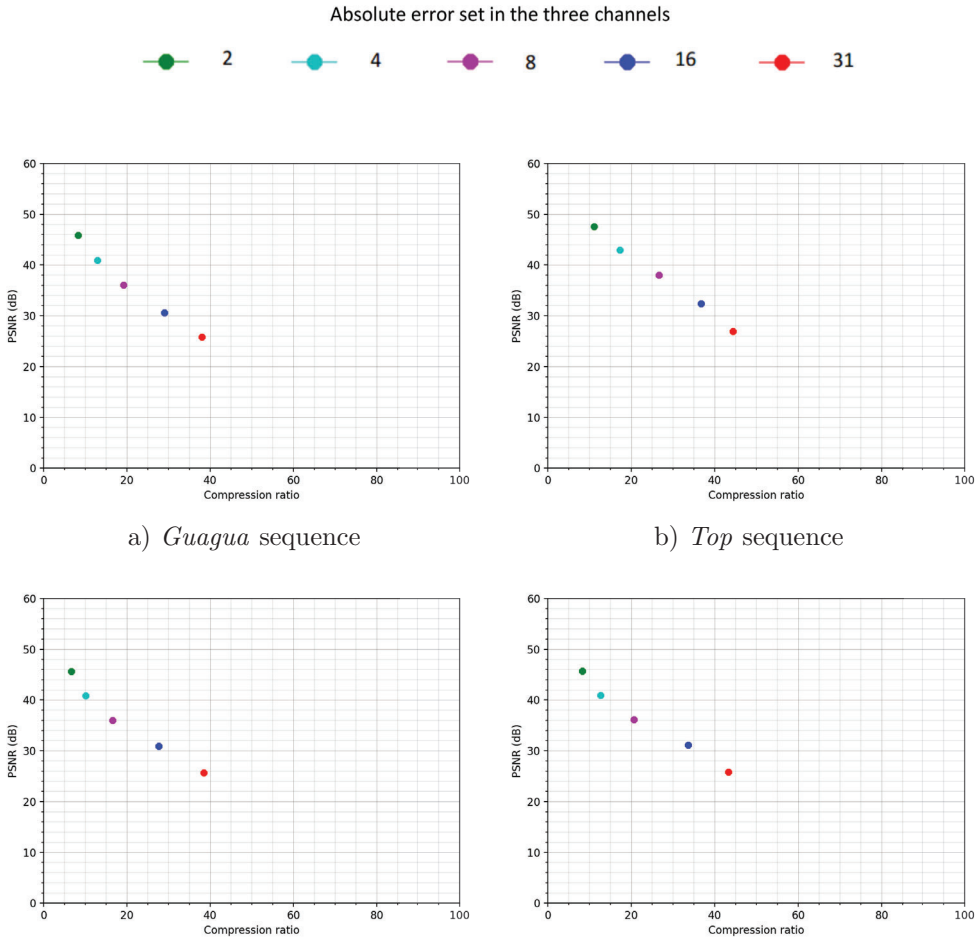
In the lossless scenario ($A_{Y,Cb,Cr} = 0$), a considerable video quality is reached for all the analysed video sequences. Under Alternative 2, around 52 dB are obtained for all the video sequences in the dataset. In the case of Alternative 3, PSNR values are lower, being around 34 dB and 36 dB for the *Guagua* and *Top* sequences, respectively, and still 52 dB for the two videos of the Stanford dataset.

Best results in terms of PSNR are obtained for Alternative 1, at the expense of a penalty in terms of CR. Maximum compression ratio of 38 is obtained for *Guagua* and *Death Circle* sequences, while this value is increased up to 44 for *Top* and *Nexus* videos. These results are obtained for $A_{Y,Cb,Cr} = 31$ and guaranteeing an acceptable visual quality, since PSNR ≥ 26 dB in all the cases.

Compression ratios up to 20 (i.e., 0.4 bits per pixel) and 28 (i.e., approximately 0.29 bits per pixel) are obtained for the *Guagua* and *Top* sequences, respectively, when fixing $A_Y = 4$ and $A_{Cb,Cr} = 8$ under Alternative 2. For the case of *Death Circle* and *Nexus* videos, maximum CRs of 22 (i.e., 0.36 bits per pixel) and 26 (i.e., around 0.31 bits per pixel) are obtained respectively, using the same approach and identical error values.

In general terms, higher compression ratios are obtained under Alternative 3. Fixing also the error limits to $A_Y = 4$ and $A_{Cb,Cr} = 8$, maximum CRs up to 30 (i.e., around 0.27 bits per pixel) and 47 (i.e., 0.17 bits per pixel) are achieved for the *Guagua* and *Top* sequences, respectively. Compression results are almost 27 (i.e., around 0.3 bits per pixel) and 34 (i.e., approximately 0.24 bits per pixel) for *Death Circle* and *Nexus* videos, respectively, by applying the mentioned error values to the three spectral channels. Under this configuration, compression has not effect yet over the video quality (approximately 30 dB for IDS sequences and 32 dB for Stanford videos) from a visual inspection.

When error limits are increased to $A_Y = 8$ and $A_{Cb,Cr} = 16$, higher compression ratios are achieved. For Alternative 2, maximum CRs of 29 and 38 are obtained for *Guagua* and *Top* sequences, respectively. In the case of *DeathCircle* and *Nexus* videos, CRs up to 31 and 36 are reached, respectively, specifying the same error limits. In all these test cases, video quality is over 25 dB, ensuring visual fidelity. Compression results under this error configuration but for Alternative 3 achieve maximum values of 47 and 66 for the *Guagua* and *Top* sequences, respectively, while video quality is still over 25 dB in both cases. The same video quality level is achieved for the Stanford sequences applying the same error limits, with compression ratios up to 43 and 53 for *DeathCircle* and *Nexus* sequences, respectively.



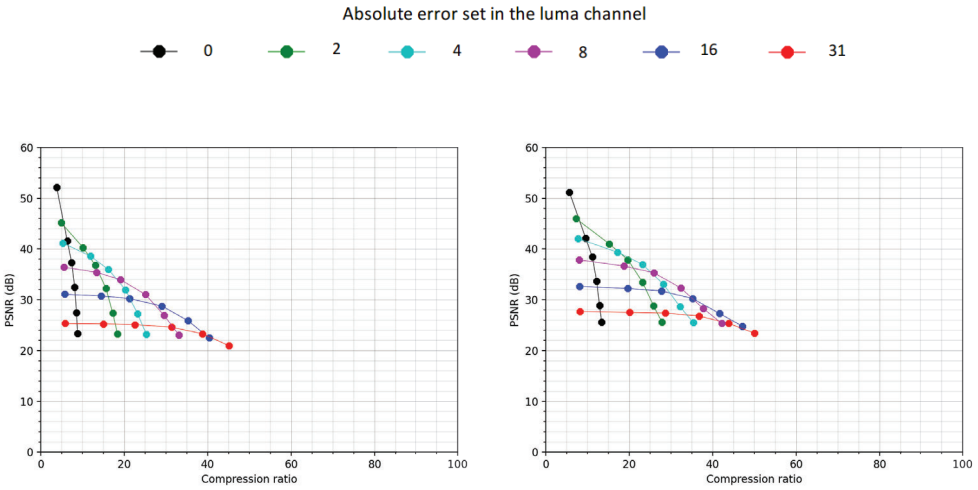
a) *Guagua* sequence b) *Top* sequence

c) *DeathCircle* sequence d) *Nexus* sequence

FIGURE 5.29: Relationship between the compression ratio and video quality, measured by PSNR, for original RGB video (Alternative 1)

In addition to the PSNR, which is a mathematically-defined measure of the video quality, results are also provided in terms of UIQI [200]. This metric tries to provide a quality measurement approach independent of the images under test, the viewing conditions or the observer point of view. I has been observed that under Alternative 3, UIQI values higher than 0.92 have been obtained for all the tests performed for the different video sequences in the dataset. These results are considered satisfactory, since these values are closed to the maximum allowed (i.e., 1 is just achieved under lossless compression).

From a visual point of view, video degradation is clearly appreciated in Alternative 3 for higher compression ratios (e.g., for $A_Y \geq 16$), though object shapes are still preserved. Some examples are provided in Figures 5.32 and 5.33, where a frame of the *Death Circle* and *Guagua* sequences is shown, respectively, applying in each case different error levels to the luma and the chroma channels. As it is reflected in both pictures, degradation is not



a) *Guagua* sequence

b) *Top* sequence

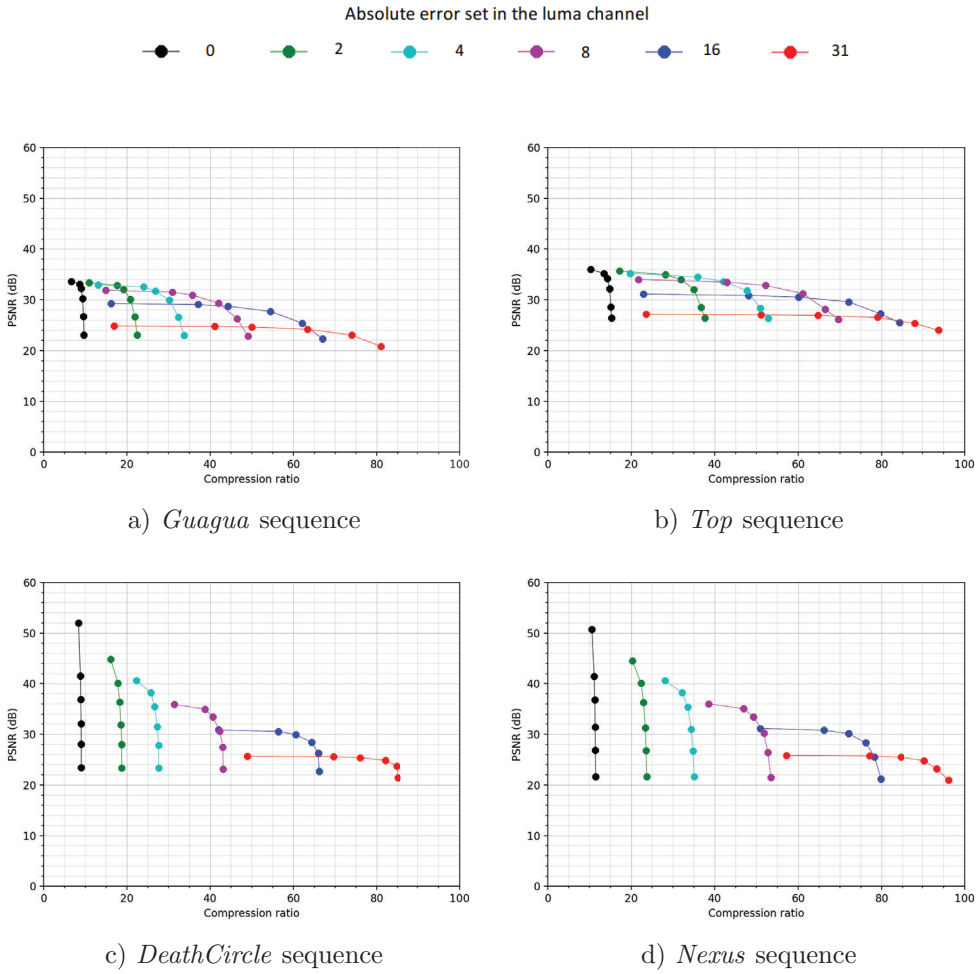
c) *DeathCircle* sequence

d) *Nexus* sequence

FIGURE 5.30: Relationship between the compression ratio and video quality, measured by PSNR, after applying RGB to YCbCr transformation (Alternative 2)

visually appreciated until the third frame, when a slight degradation is observed at the top of the frames, applying error values of $A_Y = 16$ and $A_{Cb,Cr} = 4$. Under this configuration, CRs of 61 (i.e., 0.13 bits per pixel) and 44 (i.e., 0.18 bits per pixel) are obtained for each video sequence, respectively. PSNR values are around 30 and 28 dB for the *Death Circle* and *Guagua* sequences, respectively, while UIQI ≥ 0.99 in both cases.

The highest level of degradation is observed in the last frame, when $A_Y = 31$ and $A_{Cb,Cr} = 8$. In this point, brightness losses are combined with colour saturation, obtaining a distorted sequence where objects are still clearly identified in the scene. In addition, some artifacts are observed, related to the way in which the prediction is carried out by the CCSDS 123.0-B-2 algorithm. In the full mode, which is the selected one for the tests performed, the predictor uses the previous vertical, horizontal and diagonal neighbour samples to predict the current one, and this may result in diagonal artifacts. The level of degradation



a) *Guagua* sequence

b) *Top* sequence

c) *DeathCircle* sequence

d) *Nexus* sequence

FIGURE 5.31: Relationship between the compression ratio and video quality, measured by PSNR, after applying RGB to YCbCr transformation and chroma subsampling (Alternative 3)

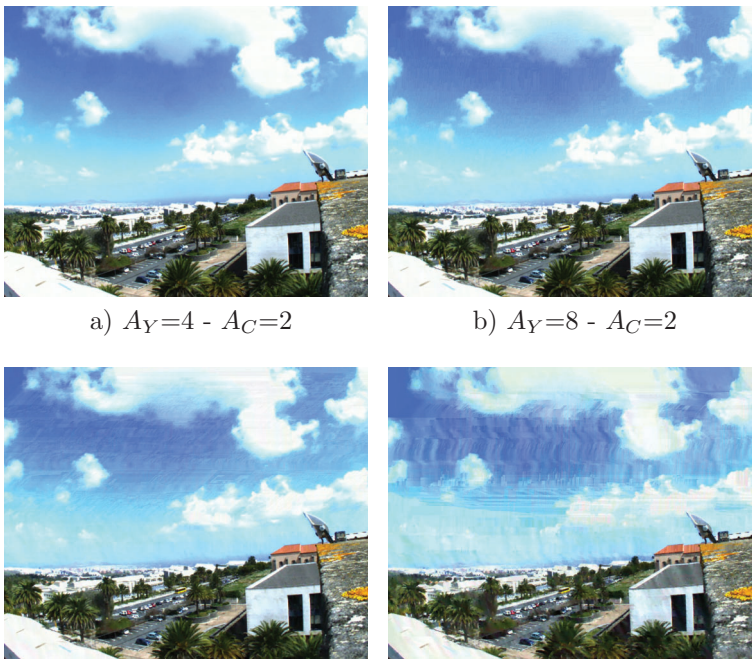
introduced by these artifacts depends on the absolute error set in the prediction. Since higher errors are being set for the chroma channels, these artifacts tend to has a higher impact in the image colour than in the image shapes, which are preserved by the luma channel. Under this configuration, compression ratios of 83 (i.e., approximately 0.1 bits per pixel) and 64 (i.e., around 0.13 bits per pixel) are obtained for the *Death Circle* and *Guagua* sequences, respectively, with a PSNR around 25 dB in both cases. Applying this error configuration, UIQI values of 0.967 and 0.981 are obtained for *Death Circle* and *Guagua* sequences, respectively.

The obtained results demonstrate the goodness of the proposed solution for Remote Sensing applications, having achieved compression ratios up to 39 (i.e., 0.21 bits per pixel) in the experiments carried out in this work without observing any degradation by visual inspection. Higher compression ratios can be achieved at the cost of decreasing

the decompressed video quality. While in this case the present objects are still clearly distinguishable in the decompressed video, the trade-off between compression ratio and decompressed video quality has to be set to meet the targeted application requirements.

a) A_Y=4 - A_C=2 b) A_Y=8 - A_C=2 c) A_Y=16 - A_C=4 d) A_Y=31 - A_C=8

FIGURE 5.32: Decompressed frames from the *DeathCircle* video with different maximum errors values for the Y channel (A_Y) and for the Cb and Cr channels (A_C).



a) A_Y=4 - A_C=2 b) A_Y=8 - A_C=2

c) A_Y=16 - A_C=4 d) A_Y=31 - A_C=8

FIGURE 5.33: Decompressed frames from the *Guagua* video with different maximum errors values for the Y channel (A_Y) and for the Cb and Cr channels (A_C).

5.7 Conclusions

In this Chapter, the concept of modular compression solution is demonstrated through its applicability to four different validation scenarios: lossless one-dimensional data and image compression; lossless and near-lossless hyperspectral image compression; and near-lossless video compression. The proposed compression chain for each one of the validation scenarios is mainly comprised by a prediction-based preprocessor and an entropy coder, selecting the appropriate alternative from the functional blocks developed and presented in Chapters 3 and 4.

Characterization results are provided for lossless compression scenarios (i.e. one-dimensional data and hyperspectral imaging), including maximum clock frequency and resources utilization in a variety of space-grade FPGAs, including the novel BRAVE family. The near-lossless hyperspectral image compression scenario has been successfully validated just on the Xilinx Kintex UltraScale XCKU040 FPGA, because of its main functional blocks have been designed following an HLS workflow and that device is the target one on the CHIME program. Finally, the video compression approach has been verified at algorithmic level, demonstrating the viability of using the CCSDS 123.0-B-2 algorithm not only to compress multi- and hyperspectral images, but also panchromatic and RGB video sequences. This derives in a versatile compression solution capable of compressing data with different nature in future space missions that embark different kind of sensors, reducing at the same time the area overhead.

Chapter 6

Conclusions and future work

This Chapter summarises the contributions of the Thesis to the field of hardware implementations of low-complexity algorithms for on-board data compression. The modular strategy proposed, based on the development of functional blocks, allows the user to create suitable solutions for the target application. The proposed scheme provides the basis to design efficient systems to compress one-dimensional data, HSI and for video sequences on-board satellites. In addition, future research lines are drawn, which will complement and improve the work presented in this Thesis.

6.1 Conclusions

Embarking high-resolution sensors on-board satellites is becoming increasingly popular in imaging space missions, since these sensors are powerful tools to acquire a high amount of information about the observed scene. However, both on-board storage resources and the downlink bandwidth to the ground stations are limited, making difficult the proper handling of the collected data in raw format (i.e., as it is acquired). For this reason, employing on-board compression techniques becomes mandatory to efficiently reduce the volume of the acquired data, such as multi- and hyperspectral images, video sequences or a non-formatted data stream (i.e., one-dimensional data).

The need for on-board compression has motivated space companies and academic institutions to dedicate efforts to design efficient compression techniques that could be deployed on-board. The implementation of low-complexity compression algorithms on hardware available on-board satellites is a constant challenge, mainly because the solutions must meet strict requirements in terms of low power consumption and high frequency (real-time processing capabilities is demanded by many applications) providing, on the other hand, high compression performance (i.e., high image quality and high compression ratio). Besides, on-board solutions have to be robust against the effect that ionizing radiation has on electronics which operate in a harsh environment, such as the outer space. Techniques such as Triple Modular Redundancy, scrubbing, EDAC, etc. alleviate this problem at the expense of impose more constraints to the designed systems.

This Thesis tries to solve these problems, making a contribution to the development of efficient hardware implementations for on-board data, hyperspectral image and video compression. The Thesis approach consists on the use of a modular strategy providing a number of functional blocks (mainly prediction-based blocks and entropy coder blocks) that can be combined to form a complete compression solution. These functional blocks are designed and fully characterized, allowing the user to accelerate the design process by simply concatenating the selected blocks required for a specific application. Moreover, the functional blocks have a high level of configuration, easing its adaptation to the mission requirements (sensor type, hardware occupancy, performance, etc.). The Thesis is focused in the use of FPGAs as hardware platforms, as they are an attractive and available option for the space industry. In addition to a high performance, flexibility and low power consumption, FPGAs also has reduced development costs compared to ASICs and even multiprocessor circuits.

The following sections detail the results of the Thesis and relate them to the objectives initially proposed and described in Chapter 1.

6.1.1 State-of-the-art analysis

There are several compression algorithms available in the specialised literature, including both lossless and lossy, to compress one-dimensional data, and multi- and hyperspectral images on-board satellites, with different levels of complexity and performance. Regarding video compression, there are not (to the date) specific algorithms proposed to compress video in space, except for tailored versions of video encoders used on ground, such as H.264 and H.265, used with a reduced set of features to adapt their architectural complexity to the embarked hardware. The feasibility of a one-dimensional data, hyperspectral image and video compression algorithm to be implemented on hardware available on-board is not commonly addressed in the state-of-the-art. Moreover, architectural studies focused on how to reduce hardware occupancy or to increase performance are not discussed in-depth.

As part of this Thesis, an exhaustive analysis of the available literature in the on-board data compression research topic has been performed, distinguishing different compression solutions depending on their performance and the nature of the data handled. From this study of the state-of-the-art, the current trends in the development of on-board data compression techniques are identified, as well as the most commonly employed hardware technologies to implement compression solutions. The focus is on compression algorithms for one-dimensional data and 3D images, since the interest of the space industry is moving from traditional images in the spatial domain to HSI, which provides extra information in several wavelengths for EO missions.

This analysis narrows down the compression algorithms to those that exhibit a good compromise between complexity and compression efficiency. In conclusion, the contributions of the Thesis are focused on prediction-based decorrelators and entropy coders based con Golomb-Rice codes. Specifically, those algorithms proposed by the CCSDS, the international organism responsible of publishing standards for the development of data systems for space missions, are taken into consideration.

6.1.2 Functional blocks

The main contribution of the Thesis is the development of basic functional blocks as the basis to design compression systems. They are fully configurable and can be used with any sensor type. These functional blocks are fully verified and characterized for a range of FPGAs qualified for space missions. The contributions can be summarised as follows:

- Three different prediction-based blocks have been developed, all of them compliant with CCSDS standards. The unit-delay predictor, based on the CCSDS 121.0-B-3 standard, is focused on one-dimensional data decorrelation. The 3D predictors compliant with the CCSDS 123.0-B-1 and 123.0-B-2 standards provide both spatial and spectral decorrelation when processing multi- and hyperspectral images under lossless and near-lossless compression, respectively. Data dependencies present on the processing of the possible input sample arrangements have been studied in detail, in order to overcome them to achieve a maximum throughput of one sample per clock cycle under specific configurations. For this reason, different predictor architectures have been proposed and implemented. As a result, these blocks are highly configurable (providing a set of parameters to tailor it to the application) and adaptable to the different sensors in the market. The blocks have been fully characterized and results are provided for Xilinx Kintex UltraScale XCKU040 FPGA. These results (and results on other FPGAs not present here for the sake of brevity) demonstrate the goodness of the proposed functional blocks to fit well on a space-grade FPGAs.

- Three entropy coding blocks have been developed, which are also compliant with CCSDS standards. The block-adaptive encoder, originally defined in the CCSDS 121 universal lossless compression standard, is based on Rice coding. The sample-adaptive and the hybrid encoder alternatives are presented in the CCSDS 123.0-B-2 near-lossless compression standard for multi- and hyperspectral images. Although both are based on Golomb codes, the hybrid encoder allows higher CRs by exploiting low-entropy data. It has been demonstrated that the developed functional blocks can be successfully mapped on a Xilinx Kintex UltraScale XCKU040 FPGA, providing a high throughput to achieve real-time processing, together with a low area footprint. The hybrid encoder block was designed using the HLS methodology.

6.1.3 Solutions provided

With the functional blocks developed in the Thesis, a number for different systems can be designed. The Thesis provides the following examples:

- **A one-dimensional data compression solution mainly comprised by an unit-delay predictor block and an encoder block**. Among the entropy encoders designed, the use of the block-adaptive encoder is recommended because the solution will be fully compliant with the CCSDS 121.0-B-3 universal lossless compression standard. The performance in terms of throughput is high enough to achieve real-time capabilities for next-generation sensors, reaching up to 176.3 and 163.7 MSamples/s for Xilinx Kintex UltraScale XCKU040 when processing input samples with a dynamic range of 8 and 16 bits, respectively. Hardware occupancy is reduced, using up to the 4.2% of the LUTs available in the XCKU040 FPGA under a restrictive configuration ($D = 32$ and $J = 64$) and with a negligible consumption of internal memory resources (around the 0.8%).

- **A lossless multi- and hyperspectral image compressor formed by a 3D predictor block and an encoder block**. The 3D predictor is the one defined in the CCSDS 123.0-B-1 lossless compression standard. Selecting the predictor block that process hyperspectral images in BIP order, a throughput of 1 sample per clock cycle is achieved. Although the hybrid encoder can be selected, it would not be a solution compliant with the CCSDS 123.0-B-1 compression standard. In this case, and with the sample-adaptive encoder, a maximum performance of 151.6 MSamples/s is obtained for Xilinx Kintex UltraScale XCKU040 when processing AVIRIS images (16 bits per sample). Logic resources consumption is reduced (around the 3% of available LUTs in a XCKU040 FPGA), while the memory usage is clearly dependent on image size (e.g. the 12% of BRAMs in a XCKU040 FPGA for AVIRIS).

- **A near-lossless multi- and hyperspectral image compressor formed by a 3D predictor block and an encoder block**. The 3D predictor is the one defined in the CCSDS 123.0-B-2 near-lossless compression standard and the entropy coder could be anyone of the alternatives described in the same standard, though the hybrid one has been selected for validation purposes. Both functional blocks were developed using a HLS methodology. This is, to the best of our knowledge, the first full functional solution based on the CCSDS 123.0-B-2 standard available in the state-of-the-art [62]. Logic resources consumption is around the 7% of available LUTs, the

BRAMs usage is a 14.5% for a XCKU040 FPGA processing a AVIRIS scenes at 125 MHz. A comparison between HLS and VHDL designs is presented in section 5.5.1.4, where the conclusion is that although the area footprint of an HLS design is about the double of a VHDL design, the maximum design frequency is similar. Finally, these HLS functional blocks can be combined with VHDL blocks to produce a specific solution (e.g. combining the predictor designed in HLS with the block-adaptive encoder designed in VHDL, approach used for the CHIME instrument).

- **A monochrome video compressor based on the CCSDS 123.0-B-2 near-lossless algorithm**. The approach consists in using the temporal domain t to predict the information of the subsequent video frames instead of the previous spectral channels z, as it is done in the CCSDS 123.0-B-2 standard. Results show the goodness of the proposed solution, achieving compression ratios up to 30 with a considerable video quality (more than 32 dB of PSNR), which has been checked by visual inspection after reconstruction. These results satisfy video compression demands on current space missions, together with a reduced architectural complexity and an acceptable reconstructed video quality.

- **An RGB video compressor based on the CCSDS 123.0-B-2 near-lossless algorithm**. The monochrome video compression solution has been extended to compress RGB video sequences. To achieve this goal, the RGB video is split into 3 data sequences, one per color channel, and independently processed within the CCSDS 123.0-B-2 compressor. Additionally, in order to increase the overall compression performance without significantly decreasing the video quality, two extra preprocessing stages have been added, including an RGB to YCbCr spectral transformation and a spatial subsampling of the chroma channels. These steps also reduce the overall computational burden, since the amount of data to be processed by the CCSDS 123.0-B-2 compressor is reduced. In the experiments carried out, the compression ratios reached were as high as 39 (i.e., 0.21 bits per pixel), without observing a high degradation by visual inspection. Higher compression ratios can be achieved at the cost of decreasing the decompressed video quality, needing to define a trade-off between both metrics to meet the targeted application requirements. This solution provides a clear advantage versus commercial video encoders, such as H.264 and H.265, though achieved compression ratios are clearly lower but still acceptable for on-board video compression requirements.

Based on the functional blocks provided, a general system able to compress on-demand one-dimensional data, multi- and hyperspectral images, and panchromatic and RGB video sequences is feasible. The advantage to have only a single processing core, with reduced footprint and low power consumption, can be a very interesting option for many space missions. Moreover, the capability of some FPGAs to be reconfigured on-board will allow to adapt its functionality to unforeseen situations or simply to instrument aging during the mission life, simply by replacing one of the implemented building blocks of the design or its configuration.

6.1.4 Validation

As a result of the Thesis, the solutions provided (and summarized in the previous section) are used in a number of research projects dedicated to the design of the compression units for different instruments. Among them, the following can be mentioned:

- A one-dimensional data compressor was designed for Lagrange ESA-funded program [183] and the SUNRISE III space mission [184] in collaboration with the Instituto de Astrofísica de Canarias (IAC)and with the Instituto de Astrofísica de Andalucía (IAA). The solution is compliant with the CCSDS 121.0-B-3 standard and work with a clock frequency of 150 MHz on the Xilinx Kintex UltraScale XCKU040 FPGA.

- A lossless/near-lossy compressor was developed for CHIME instrument in collaboration with Thales Alenia Space in France (TASiF) and Thales Alenia Space in Spain (TASiS). This implementation was based on the CCSDS 123.0-B-2 standard modified to use different error values for normal pixels and cloud pixels. A band-dependent error is applied for both kind of pixels. The CCSDS 121.0-B-3 block-adaptive encoder presented in Section 4.2 was used as entropy coder. A combined methodology using HLS and VHDL was used in the design and a prototype was done on a Xilinx Kintex UltraScale XCKU040. This validates the use of this compressor in the CHIME instrument in the Copernicus program.

- An RGB video compressor is under development in the context of the H2020 VIDEO project. The RGB sensor is provided by Pyxalis and the objective of the project is to compress Full HD video (1920x1080 pixels) in real-time. This implementation is based on the CCSDS 123.0-B-2 standard and extended with some preprocessing stages to

be capable of handling RGB video sequences. This hardware implementation will be mapped on a Xilinx Kintex UltraScale XCKU040 FPGA.

6.1.5 Summary

The use of compression in space missions (mainly in Earth Observation) is not a trivial decision. Scientists usually prefer raw data because the captured scenes are unique and cannot be reproduced, hence introducing data losses can harm the results of the experiment. This Thesis contributes to providing an important flexibility to compression data on a specific mission, allowing the user to select the appropriate solution keeping in mind the mission requirements and the use the collected data will have on ground.

With the accomplished research work, the main objectives proposed in this Thesis are achieved. Low-complexity solutions for on-board data, multi- and hyperspectral images, and monochrome and RGB video compression have been proposed, obtaining results in terms of timing and area utilization. These solutions have been mapped on space-grade FPGAs and the design decisions made, together with the inherent parallelization features of this technology, have contributed to successfully overcome initial forecasts drawn at the beginning of the Thesis. The results have been properly evaluated in terms of computational performance and hardware occupancy, taking into account also the available development time to provide the compression solution. Some of the proposed approaches have been validated directly on-chip, demonstrating in this way their viability to work on embarked hardware. Two different design methodologies have been employed, RTL and HLS (even combining both), keeping in mind restrictions in terms of scheduling, since the proposed solutions are candidates to be implemented as part of different space programs. Difficulties found during the design process have been identified and solved when possible, which were mainly related to data dependencies present in the datapath of the implemented algorithms. All these contributions are expected to help to reduce design costs and to provide efficient compression solutions in terms of hardware occupancy and computational performance for future space missions, in which high-resolution sensors are expected to be embarked for observation purposes.

From the technical point of view, we can conclude that both the CCSDS 121 and 123 algorithms are good candidates for on-board data compression implementations on space-grade FPGAs, showing a low hardware occupancy compared with other compression algorithms available in the state-of-the-art that are specifically though for space applications.

They have been successfully mapped in a wide variety of RHBD FPGA technologies, including the novel BRAVE family. Throughput results extracted for CCSDS 121.0-B-3 and CCCSDS 123.0-B-1 proposed solutions demonstrate their suitability to be used on-board for real-time applications. The achieved results are possible thanks to the algorithm nature and the parallelization capabilities provided by FPGAs in absence of data dependencies. This is not currently feasible for the CCSDS 123.0-B-2 approach, whose internal dependencies among internal prediction stages must be studied to considerably increase throughput. Nonetheless, this latter solution can exploit the possibility of replicating the compression instances in order to achieve higher throughput, since logic resources utilization has still some margin to allow it.

6.2 Further research work

Although compression is a well established area in ground applications, its use in space missions is limited and most of the times forced by the on-board memory or downlink rate limitation. It is foreseen that this will change in the near future. The advent of more precise instruments and more on-board processing capabilities will enforce the compression as a must to have on-board application. The direct reuse of the solutions implemented on ground applications will not work on space missions, mainly because the compression objectives will be different (data quality will be generally more important than in ground applications) and the limited availability of power on-board will prevent the use of the most advanced computational solutions in the state-of-the-art. Hence, this research line will continue in the next years as the missions demand increase. In Europe, ESA and the European Commission are aware of this reality publishing projects calls in this area. In particular, the continuation of this research will be framed within the projects awarded to the Group (all funded either by European Space Agency or the European Union).

The first research line is to achieve more efficient hardware implementations for hyperspectral image compression. The efforts will be focused on developing a VHDL description of the CCSDS 123.0-B-2 near-lossless compression standard. The main objective of this design will be to enhance the throughput compared to its HLS provided in this Thesis, by obtaining a cycle-accurate model that allows a fine control of the datapath latency. This activity will be done in the context of the *Lossless/lossy multispectral & hyperspectral compression IP core* project (ESA Contract No. 4000136723/22/21/NL/CRS). This

project is leaded by IUMA in a consortium with Thales Alenia Space Spain, National and Kapodistrian University of Athens, and Universitat Autònoma de Barcelona.

Different predictor architectures will be proposed depending on both the input samples arrangement and the target performance in terms of throughput. In addition, the three entropy encoding alternatives will be available. The hybrid encoder must be completely developed in VHDL, while the block- and the sample-adaptive options, described in Sections 4.2 and 4.3, respectively, can be reused as they are. It is expected that this VHDL description will be able to compress at a maximum rate of 1 input sample per clock cycle for certain configurations under BIL order (the one less restricted in terms of data dependencies under near-lossless mode) by exploiting the inherent parallelism of the algorithm in absence of data dependencies in the processing of two consecutive input samples. Moreover, the developed solution must still have an acceptable hardware occupancy to fit well on space-grade FPGAs, considerably reduced to the logic resources and memory consumption reported by the HLS development. A reduced hardware occupancy is crucial, since it must be enough margin to implement redundancy techniques to provide robustness against radiation effects during the space mission lifetime. All these improvements in terms of performance should be done guaranteeing at the same time full compliance with the standard, providing all the possible configuration options and working modes.

A tailored version of this compressor will be described in VHDL in the scope of the ESA project RFP/1-9941/19/NL/NA entitled *Copernicus HPCM (High Priority Candidate Missions) - CHIME (Copernicus Hyperspectral Imaging Mission for the Environment) phases B2, C/D and E1 (prototype and recurrent satellites)*, led by Thales Alenia Space in France.

A second research line will be related to video compression in the space and it will be funded in the ESA project *Efficient Video Compression for space* (ITT AO/1-1-10954/21/NL/MGu), in which our Group will collaborate with Thales Alenia Space in Spain that acts as project leader. This Thesis concludes that the proposed solution for video compression based on the CCSDS 123.0-B-2 multi- and hyperspectral image compression standard is promising at algorithmic level. A flexible solution was provided for next-generation space missions to compress both hyperspectral images and video sequences using the same processing core. Nonetheless, its performance is still far from video encoders typically used on ground, specially to reach high CRs. For this reason, it is proposed to develop a tailored version of the well-known H.264 encoder to accomplish video compression demands on-board satellites. Firstly, an HLS model will be developed

to perform a space design exploration, in order to identify what parts of the encoder architecture are candidates to be removed from the datapath, which can be dispensable in a Remote Sensing scenario (e.g. motion estimation or the use of B slices). To do this, the impact in terms of CR must be evaluated. After completing this stage, it is expected to obtain a customized high-level implementation that fits well on the hardware resources available on-board. Taking this HLS model as starting point, the next stage is to develop a VHDL description, equivalent to the prior from a behavioural point of view. This VHDL development will be optimized to obtain higher throughput and a reduced hardware occupancy compared to the HLS model.

Since the proposed solution for RGB video compression employs a processing instance per color channel, it is not feasible to apply it to multispectral video, which makes use of 6-10 spectral bands. It is expected that this technology will be interesting for the space industry in the near-future, because of the high quantity of information collected in both the spectral and the temporal domain, useful for many scientific applications on ground. For this reason, as a future line of this Thesis, alternative compression approaches are currently under study for this novel technology, including transform-based approaches, such as the one proposed for RGB video compression, or solutions based on lightweight convolutional networks, which must take into account on-board hardware restrictions.

It is intended to validate on-chip both hyperspectral image and video compression proposed solutions not only on the Xilinx Kintex UltraScale XCKU060 FPGA, but also on the novel BRAVE family. These SRAM-based FPGA devices are expected to be the reference in the near-future for the space industry, because of their high reprogrammability, high performance, high resources availability, reduced costs and low power consumption.

Finally, a third research line is opened with the European project proposal PDHT-NG (Payload Data Handling and Transmission - New Generation) entitled *Innovative and Flexible New Generation of Payload Data Handling for New Space Applications*, which is currently under evaluation by the European Commission. The main idea is the same developed in this Thesis but extended to a higher level of abstraction. This project will create a multi-mission technology that can address a large range of missions, from high-end satellite Copernicus to low-end constellation and will be compatible with high accuracy sensors (visible up to video, Infra-Red, hyperspectral, ultraspectral, Lidar, Radar, etc.). It will only rely on European technology and will contribute to EU non-dependence for the development of Earth Observation technologies. The basic idea is to develop a standard hardware platform (based on the COTS Kalray MPPA, Massively Parallel Processor Array

Architecture, Manycore) and a development platform. The applications (compression, object detection, monitoring, etc.) will be deployed seamlessly in the system through this development platform. The partners are the following ones: Thales Alenia Space in France (Leader), Thales Systems Romania, Teletel Greece, KP Labs Poland, Edisoft Portugal, Orbital EOS Spain and University of Las Palmas de Gran Canaria.

Appendix A

Sinopsis en español

En este Capítulo se proporciona una visión general del trabajo de investigación realizado en esta Tesis Doctoral. En concreto, se resaltarán las contribuciones realizadas en el campo de las implementaciones sobre FPGAs de algoritmos de compresión de datos unidimensionales, imágenes multi- e hiperespectrales, y sequencias de vídeo a bordo de satélites, poniendo especial énfasis en el concepto de modularidad para generar una solución de compresión óptima para la aplicación final.

A.1 Introducción

Las aplicaciones de teledetección se han hecho muy populares para la industria espacial durante las últimas décadas, ya que los sensores de alta resolución permiten obtener información útil para fines de vigilancia, caracterización y detección, entre otros. Este tipo de sensores, comúnmente utilizados en misiones de observación de la Tierra, también están ganando interés para la exploración espacial, y están siendo considerados por las agencias espaciales para estudiar la Luna o la superficie de Marte.

Los sensores de alta resolución de última generación están concebidos para adquirir el mayor área posible, sin degradar la resolución ni la calidad de los píxeles. También se puede incorporar información adicional en el dominio espectral y/o temporal para obtener mayor nivel de información sobre el objetivo analizado. Sin embargo, esa gran cantidad de información debe ser transmitida, almacenada o procesada. Además, se espera que los sensores de imagen de nueva generación aumenten en los próximos años tanto la resolución espacial como la espectral.

En este sentido, la compresión de datos surge como una solución para esquivar estas limitaciones, reduciendo el volumen de datos antes de enviarlos a tierra. Sin embargo, la reducción de datos sigue siendo un reto para la industria espacial, ya que los satélites no cuentan con suficiente capacidad de cálculo y almacenamiento para gestionar tal volumen de información, y el ancho de banda del enlace descendente con las estaciones terrestres es limitado para transferir esos datos en crudo (es decir, tal y como se adquieren).

Las técnicas de compresión pueden ser con o sin pérdidas. La compresión sin pérdidas preserva toda la información presente en los datos originales, que puede recuperarse completamente durante el proceso de descompresión. Por esta razón, la compresión sin pérdidas resulta interesante para la comunidad científica, ya que mantiene la fidelidad de los datos adquiridos por el sensor. Por otro lado, la compresión con pérdidas produce ratios de compresión más altos al introducir pérdidas en la cadena de compresión. Como consecuencia, la información recuperada no es idéntica a los datos captados por el sensor. En un punto intermedio se encuentra la compresión casi sin pérdidas, que consigue ratios de compresión más altos que las técnicas sin pérdidas sin llegar a los niveles de la compresión con pérdidas. La compresión casi sin pérdidas supone un buen compromiso entre los ratios de compresión alcanzados, la calidad de la imagen y la complejidad del algoritmo. En cuanto a la naturaleza del algoritmo de compresión, en el estado del arte destacan claramente dos técnicas: los enfoques basados en la predicción y los basados en transformada. Los métodos

de compresión basados en transformada reducen la información redundante presente en los datos de entrada transformando la información del dominio espacial a una representación alternativa, como la dimensión espectral, con el fin de decorrelacionar eficientemente los datos. Por otra parte, las soluciones basadas en la predicción se basan en el cálculo del valor del píxel actual como una suma ponderada de los píxeles en su vecindad espacial, espectral o temporal. Aunque el ratio de compresión que se consigue es peor que la de los algoritmos basados en transformada, los enfoques basados en la predicción ofrecen un buen equilibrio entre el ratio de compresión alcanzado y la complejidad del algoritmo.

La compresión de datos sigue considerándose un reto para la industria espacial. Se necesitan algoritmos y plataformas de procesamiento eficientes y de baja complejidad a bordo para comprimir esa enorme cantidad de información en tiempo real. Este ha sido un tema candente para la comunidad científica durante las últimas décadas, principalmente porque las restricciones computacionales presentes a bordo de los satélites obligan a los algoritmos de compresión a cumplir ciertos criterios de diseño. Al mismo tiempo, es muy importante preservar la calidad de los datos reconstruidos tras la descompresión, ya que será imposible capturar los mismos datos dos veces.

Esta es la razón por la que el Comité Consultivo para los Sistemas de Datos Espaciales (CCSDS), una organización internacional formada por las principales agencias espaciales del mundo para definir un procedimiento común para el desarrollo de sistemas de datos e información espaciales, ha publicado diferentes normas de compresión. Estos algoritmos se dirigen a datos de distinta naturaleza (datos 1D, 2D y 3D) y proponen distintas técnicas de compresión (es decir, sin pérdidas o con pérdidas), pero siempre cumplen la condición de una complejidad algorítmica reducida. Estas normas pretenden establecer un marco común para el desarrollo de soluciones de compresión a bordo y disponer de una solución universal para la descompresión de la información en tierra. En cuanto a la compresión de datos unidimensionales, imágenes e imágenes hiperespectrales, el CCSDS ha publicado los siguientes estándares: a) CCSDS 121.0-B-3 (Compresión de datos sin pérdidas) [27], b) CCSDS 122.0-B-2 (Compresión de imágenes 2D) [28], c) CCSDS 122. 1-B-1 (Preprocesamiento espectral basado en transformada para la compresión de imágenes multiespectrales e hiperespectrales) [29], d) CCSDS 123.0-B-1 (Compresión de imágenes multiespectrales e hiperespectrales sin pérdidas) [30] y e) CCSDS 123.0-B-2 (Compresión de imágenes multiespectrales e hiperespectrales con baja complejidad sin pérdidas y casi sin pérdidas) [31]. Existen también otros algoritmos específicamente diseñados para la compresión de datos a bordo de satélites en la literatura especializada, tanto basados en técnicas de predicción como en transformada en el dominio espectral. Sin embargo, estos

algoritmos no propocionan soluciones estándar, precisando de decompresores customizados para cada aplicación o misión espacial.

La compresión a bordo requiere no sólo algoritmos eficientes que proporcionen la compresión y la calidad reconstruida deseadas, sino también una implementación física adecuada en el hardware disponible que preserve el comportamiento correcto de la solución implementada, que debe coexistir con otras funcionalidades en la misma carga útil. Este hardware debe contar con ciertos requisitos para trabajar en el entorno espacial, como un reducido consumo de potencia y resistencia a fallos provocados por la radiación. Tradicionalmente, los algoritmos de procesamiento en general se implementan a bordo de satélites como software que se ejecuta en procesadores de propósito general. Luego se comenzó a introducir a bordo circuitos de alto rendimiento basados en ASICs, ya que ofrecen un equilibrio entre rendimiento y consumo de energía, debido a que están totalmente optimizados para una aplicación específica y la tecnología utilizada para su fabricación está pensada para aplicaciones espaciales (es decir, tolerante a la radiación). Sin embargo, los costes y el tiempo de fabricación son elevados. Por esta razón, las FPGAs están aumentando su presencia en los últimos años como parte de los circuitos de procesamiento de un satélite, debido a su flexibilidad, alto rendimiento computacional y bajo consumo de energía. Una de las principales ventajas de las FPGAs es también su reducido coste en comparación con los ASICs. La flexibilidad inherente a las FPGAs basadas en RAM permite cambiar toda o algunas partes de la funcionalidad de forma dinámica para adaptarla a los nuevos requisitos que puedan aparecer durante la vida de la misión o incluso si se produce una corrupción por efectos de la radiación. Además, las FPGAs favorecen el paralelismo de tareas, ejecutando simultáneamente diferentes operaciones si no se presentan dependencias de datos entre ellas, lo que supone una ventaja frente al comportamiento secuencial de los microprocesadores embebidos.

A.2 Objetivos y metodología de trabajo

Como se ha mencionado, los sensores de próxima generación que se integrarán en las futuras misiones espaciales adquirirán tal cantidad de datos que la compresión a bordo será obligatoria para mantener altas tasas de adquisición de datos y superar las restricciones en términos de almacenamiento disponible y ancho de banda del enlace descendente. La cantidad y los tipos de sensores a bordo de un satélite son diversos y las soluciones para la compresión de datos deben ser fácilmente adaptables o configurables a cada misión.

Además, la limitación de la memoria y los recursos computacionales a bordo exigirá la compresión de datos sobre la marcha (es decir, en tiempo real). Aunque las soluciones propuestas son agnósticas a la tecnología, se demostrarán en FPGAs, ya que son una plataforma interesante para las misiones espaciales actuales y futuras. Esto implicará el desarrollo de arquitecturas hardware que exploten eficientemente las bondades de las FPGAs.

El uso de metodologías de alto nivel acelerará aún más el proceso de diseño y verificación. Aunque el uso de metodologías de alto nivel no está extendido en aplicaciones espaciales, esta Tesis investigará el uso de HLS cuando el rendimiento y el consumo de recursos requerido por la solución se consideren adecuados para la aplicación objetivo.

En la literatura especializada abundan alternativas para la compresión de datos, que pueden clasificarse en función de la naturaleza de los datos tratados, del nivel de pérdidas introducido en el proceso de compresión o de las características de rendimiento que se priorizan, como la baja complejidad o la alta capacidad de cómputo. Además, existen enfoques de compresión específicamente pensados para trabajar a bordo de satélites, teniendo en cuenta las limitaciones de un entorno tan duro como el espacio exterior. Sin embargo, se ha observado que no existe una forma común de desarrollar e implementar soluciones de compresión para misiones espaciales, lo que hace necesarias soluciones de descompresión customizadas en tierra para cada enfoque de compresión específico.

Por esta razón, esta Tesis se centra en los algoritmos de compresión que proporcionan un buen compromiso entre la complejidad y la eficiencia de la compresión, prestando especial atención a etapas como los preprocesadores basados en la predicción y los codificadores de entropía basados en los códigos Golomb-Rice. Más concretamente, se consideran los algoritmos de compresión propuestos por los estándares CCSDS para datos unidimensionales e imágenes 3D (es decir, multi e hiperespectrales). Esto está motivado principalmente por el hecho de que, además del equilibrio que proporcionan entre el rendimiento de la compresión y la complejidad del hardware, la descompresión puede realizarse sobre el terreno utilizando un descompresor estándar, lo que permite la compatibilidad y reutilización entre diferentes aplicaciones.

El objetivo principal de esta Tesis es proporcionar soluciones modulares que puedan ser adaptadas para comprimir datos de diferente naturaleza, incluyendo información genérica o unidimensional, imágenes multi e hiperespectrales y secuencias de vídeo, adquiridos por sensores de alta resolución en misiones espaciales de nueva generación.

La flexibilidad que proporciona este esquema modular sobre FPGA también permite cambiar la funcionalidad de la solución de compresión si aparecen nuevos requisitos durante la vida de la misión. Esto se consigue sustituyendo las etapas de procesamiento adecuadas para reducir la ocupación del hardware y el consumo de energía, para aumentar la relación de compresión o para acelerar el rendimiento.

Los objetivos específicos de esta Tesis se detallan a continuación:

- Analizar las soluciones propuestas por el CCSDS para la compresión de datos unidimensionales, imágenes 2D, imágenes 3D multi e hiperespectrales y secuencias de vídeo. Este análisis determinará las mejores soluciones para ser implementadas en aplicaciones espaciales, teniendo en cuenta su complejidad y el rendimiento esperado.
- Desarrollar una exploración del espacio de diseño de los algoritmos CCSDS objeto de estudio, proporcionando alternativas arquitecturales para su implementación, en función de los requerimientos de la aplicación objetivo.
- Proporcionar soluciones eficientes y modulares para la compresión de datos en FPGAs de grado espacial con un alto rendimiento y una baja utilización de recursos hardware. La atención se centra en los algoritmos CCSDS propuestos para la compresión de datos unidimensionales y de imágenes multiespectrales/hiperespectrales debido a su interés para las futuras misiones espaciales: la norma CCSDS 121.0-B-3 [27] y la norma CCSDS 123, incluyendo la edición 1 (sólo pensada para la compresión sin pérdidas) [30] y la reciente edición 2 (que amplía la funcionalidad para la compresión casi sin pérdidas) [31].
- Proponer el uso de diferentes metodologías de diseño, como RTL y HLS, en función de los requisitos de la misión espacial en términos de tiempo de desarrollo y rendimiento. Se estudiarán los resultados obtenidos para remarcar las debilidades y fortalezas de las metodologías de diseño seguidas.
- Proponer diferentes configuraciones de hardware, dependiendo de los requisitos de compresión y de la naturaleza de los datos a comprimir. Siguiendo este enfoque, se pueden definir múltiples cadenas de compresión reutilizando módulos que realicen las etapas de predicción y codificación entrópica de forma eficiente, teniendo en cuenta la naturaleza de los datos y los requisitos de la aplicación final.

- Demostrar que las soluciones son viables para una serie de escenarios cuando se implementan en FPGAs de grado espacial. Esto se hará realizando una verificación exhaustiva y también validación en hardware.

A.3 Técnicas de compresión modulares a bordo de satélites

Se ha detectado una carencia en el estado del arte en cuanto a la versatilidad de las soluciones de compresión a bordo. Esto significa que no existe una colección de bloques funcionales que puedan ser reutilizados para diferentes propósitos o para diferentes misiones espaciales con diferentes objetivos de rendimiento, necesitando un desarrollo completo cada vez que se requiera una solución de compresión. Esto enlaza con uno de los principales objetivos de esta Tesis, que es proporcionar soluciones de compresión modulares, basadas en los estándares CCSDS, que permitan conformar una cadena de compresión optimizada para una aplicación específica teniendo en cuenta diferentes restricciones, como el rendimiento, la ocupación del hardware o el ratio de compresión deseado, seleccionando los módulos adecuados entre las alternativas disponibles. Dado que las cadenas de compresión diseñadas cumplen con los estándares CCSDS, se garantiza una correcta descompresión y gestión de los datos en tierra.

Por este motivo, en esta Tesis se han desarrollado diferentes IPs de compresión para misiones espaciales. Estos IPs están compuestos por dos bloques funcionales principales: un preprocesador basado en la predicción que actúa como decorrelador de datos y una etapa de codificación de entropía, encargada de reducir el número de bits con el que se representan los residuos de la predicción. Combinando los bloques funcionales desarrollados, tal y como se muestra en la Figura A.1, es posible implementar una solución de compresión adecuada para una aplicación o misión espacial específica. También existe la posibilidad de implementar una solución de compresión versátil capaz de procesar los datos adquiridos por múltiples sensores (es decir, de distinta naturaleza) con un solo núcleo de compresión, reduciendo la ocupación del hardware y el consumo de energía.

La selección de los bloques funcionales adecuados está condicionada no sólo por la naturaleza de los datos de entrada, sino también por las limitaciones de las misiones espaciales, incluyendo los objetivos en términos de rendimiento, ocupación del hardware o consumo de energía. La selección de la metodología de diseño adecuada (RTL o HLS) vendrá

condicionada por dichos objetivos funcionales, además de por el tiempo de desarrollo disponible.

Una ventaja adicional del enfoque modular propuesto es la reutilización. Esto significa que los bloques funcionales independientes que se han desarrollado para cada etapa de compresión (es decir, el preprocesamiento y la codificación de entropía) pueden reutilizarse en futuras misiones espaciales para la compresión a bordo. Se garantiza un comportamiento adecuado a nivel funcional, ya que estos módulos han sido profundamente caracterizados, al tiempo que el programa espacial puede suavizar la programación gracias a que se evita un desarrollo desde cero.

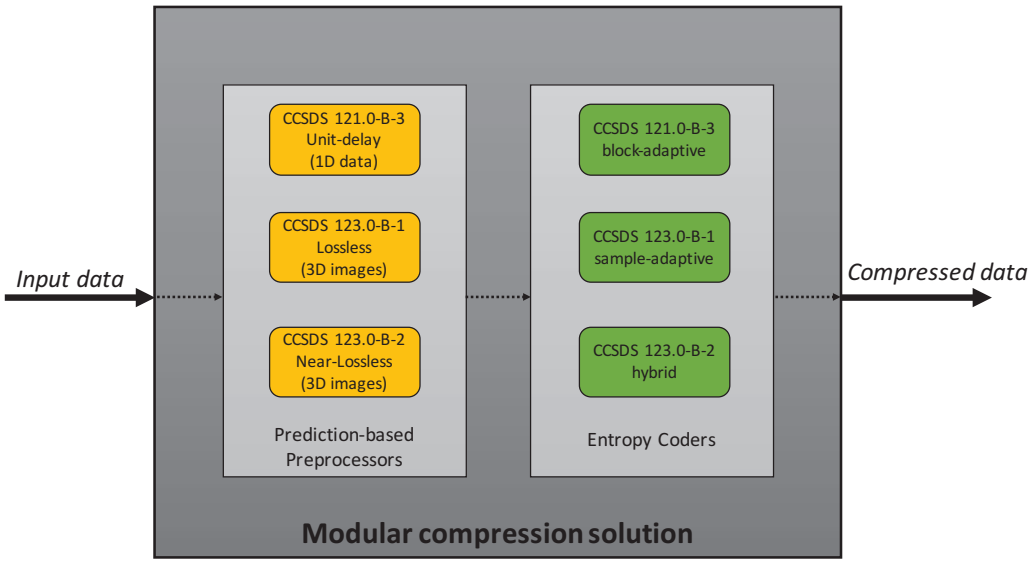


FIGURE A.1: Concepto de solución de compresión modular

A.3.1 Implementación en FPGA de un compresor sin pérdidas de datos génericos basado en el estándar CCSDS 121.0-B-3

Como compresor de datos unidimensional, se propone una implementación hardware compuesta principalmente por el predictor de retardo unitario descrito en el estándar CCSDS 121.0-B-3 y uno de los codificadores de entropía desarrollados. Por lo tanto, se obtienen tres soluciones alternativas combinando el predictor de retardo unitario con los codificadores block-adaptive, sample-adaptive o hybrid. La combinación del predictor de retardo unitario con el codificador block-adaptive cumple con el estándar CCSDS 121.0-B-3.

Como ejemplo de aplicación, se considera esta solución para su integración en la misión Lagrange [183] y también en la misión SUNRISE III [184], en la que se espera lanzar un globo a la estratosfera que actuará como observatorio solar. Estos casos de uso surgen en el contexto de una colaboración entre nuestro Grupo, el Instituto de Astrofísica de Canarias (IAC) y el Instituto de Astrofísica de Andalucía (IAA).

Se han definido diferentes conjuntos de parámetros de configuración para analizar el impacto de los valores de los parámetros del IP tanto en la ocupación del hardware como en la frecuencia de reloj máxima. Los principales parámetros que cambian entre configuraciones son el rango dinámico de las muestras de entrada *D_GEN*; el tamaño del bloque *J_GEN*; y la anchura del buffer de salida, *W_BUFFER_GEN*.

En general, se alcanzan los valores más altos de frecuencia de reloj para los valores más bajos seleccionados para el rango dinámico D y, especialmente, para el tamaño de bloque J. La combinación de valores bajos para ambos parámetros permite la reducción de la longitud de la ruta crítica. En concreto, se obtiene una frecuencia de reloj máxima de 176,3 MHz para la Kintex UltraScale XCKU040 bajo Set7, mientras que se alcanzan hasta 115,9 MHz para la Virtex5QR XQR5VFX130 bajo Set5. En cuanto al throughput, como el IP es capaz de procesar una muestra por ciclo de reloj, el rendimiento máximo es igual a la frecuencia de reloj máxima alcanzada. De este modo, se obtiene un rendimiento máximo de 176,3 y 163,7 MSmuestras/s para la Xilinx XCKU040 cuando se procesan muestras de entrada con 8 y 16 bits de precisión, respectivamente. Los resultados en términos de utilización de recursos lógicos están en consonancia con los obtenidos en términos frecuencia máxima; es decir, mientras mayor es la frecuencia máxima alcanzada, mayor es la utilización de recursos lógicos. Con la configuración más restrictiva se usa el 4,2% de LUTs en la XCKU040. La utilización de memoria es mínima en esta implementación (5 BRAMs). El consumo de DSPs también es mínimo (un máximo del 0,3% en XCKU040). Los resultados para una configuración típica, con $D = 16$ y $J = 32$, se muestran a modo de ejemplo en la Tabla A.1.

A.3.2 Implementación en FPGA de un compresor sin pérdidas de imágenes multi- e hiperespectrales basado en el estándar CCSDS 123.0-B-1

Este escenario de validación proporciona soluciones para comprimir imágenes hiperespectrales sin pérdidas. Para ello, se utiliza el predictor 3D descrito en el estándar CCSDS

TABLE A.1: Compresor de datos unidimensionales sin pérdidas - Síntesis en Xilinx Kintex UltraScale XCKU040 con $D = 16$ y $J = 32$

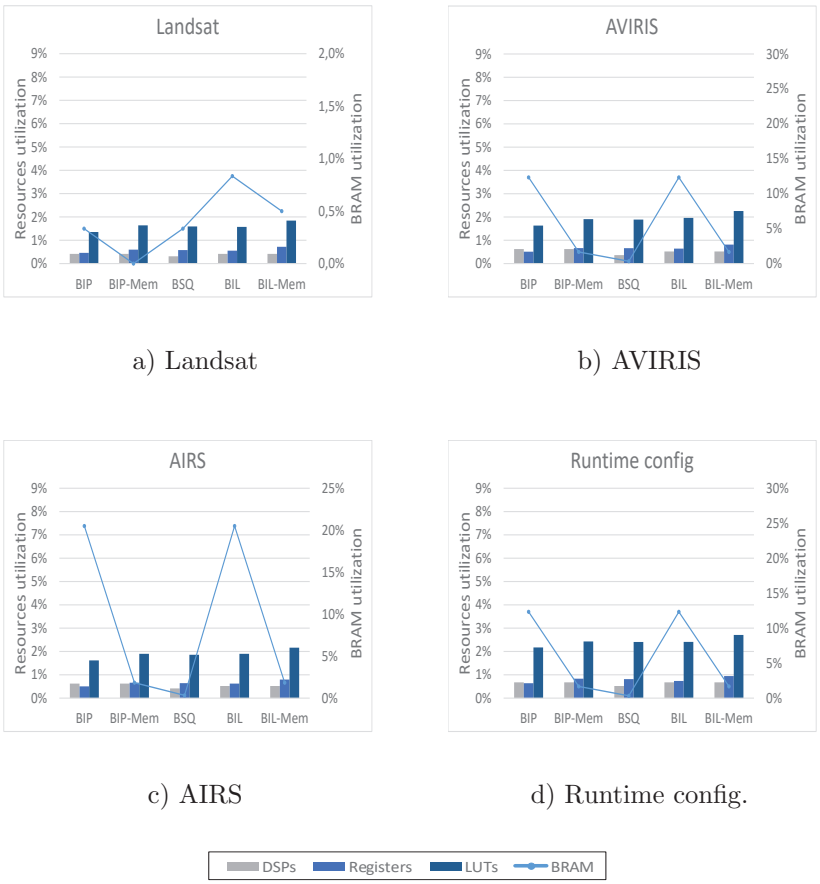
Recursos	**Total**	**Usado**
Bloques RAM	600	1
DSP48	1920	6
Registros	484800	1830
LUTs	242400	5252
Frecuencia máxima (Clk_S) (MHz)		141.1

123.0-B-1. Podrían utilizarse todos los codificadores de entropía diseñados, pero la implementación del codificador híbrido no será una solución estándar. Como ejemplo, se ha diseñado una implementación de hardware totalmente compatible con el estándar CCSDS 123.0-B-1 para la compresión sin pérdidas de imágenes multi e hiperespectrales a bordo de satélites, implementando el codificador sample-adaptive y proporcionando también la alternativa de sustituirlo por el block-adaptive. Como ejemplo de este escenario cabe destacar los IPs de compresión desarrollados bajo el nombre de SHyLoC, que actualmente forman parte del catálogo de IPs ofrecidos por la ESA para futuras misiones espaciales [58, 60] Se ofrecen resultados de varias soluciones para sensores hiperespectrales conocidos (por ejemplo, Landsat, AVIRIS y AIRS). El IP ha sido evaluado en síntesis utilizando hasta 20 conjuntos diferentes de parámetros de síntesis, tratando de cubrir un amplio rango de casos y verificando todas las arquitecturas del predictor.

La frecuencia de reloj máxima alcanzada para Kintex UltraScale XCKU040 es superior a 119 MHz para todas las arquitecturas del predictor desarrolladas en todos los escenarios planteados. No todas las arquitecturas proporcionan el mismo throughput, siendo BIP la que única que puede alcanzar una muestra por ciclo de reloj. Por tanto, el throughput máximo es de 151,6 MSamples/s en XCKU040 en el escenario hiperespectral. Los resultados de la síntesis en términos de utilización de recursos se resumen en la Figura A.2 para la FPGA XCKU040. En general, el IP utiliza pocos recursos lógicos, lo que demuestra la baja complejidad de la cadena de compresión propuesta. El uso de DSPs, LUT y registros es casi constante, con ligeras diferencias dependiendo de la arquitectura del predictor implementado y el tamaño de la imagen de entrada. Las diferencias son más notables para la utilización de la memoria, que viene determinada principalmente por la arquitectura del predictor. En el caso de las arquitecturas que no hacen uso del almacenamiento externo, el tamaño de la imagen tiene un gran impacto en el uso de memoria. Concretamente, el tamaño de una línea espectral ($N_x N_z$) condiciona esa utilización del almacenamiento ya que fija el tamaño de las FIFOs que almacenan las muestras adyacentes para el cálculo de

las sumas locales y de las diferencias locales durante la predicción. El número de bandas previas P utilizadas para la predicción también influye en el consumo de BRAMs, ya que determina el tamaño del vector de pesos y el número de elementos a considerar durante el cálculo de las diferencias locales.

En cuanto al consumo de recursos lógicos (es decir, LUTs y FFs), el rango dinámico D supone la principal restricción, ya que define el ancho de bits de las diferentes operaciones internas. Además, la resolución de pesos Ω también tiene una ligera influencia en el consumo de LUTs, ya que especifica la precisión de cada elemento del vector de pesos. En Kintex UltraScale XCKU040 se utiliza hasta el 20% de los recursos de memoria, cuando se trata de un escenario ultraespectral. En cuanto a la utilización de DSPs y LUTs, todas las arquitecturas y configuraciones proporcionan resultados similares.



a) Landsat

b) AVIRIS

c) AIRS

d) Runtime config.

FIGURE A.2: Compresor de imágenes hiperespectrales sin pérdidas - Utilización de recursos en Xilinx Kintex UltraScale XCKU040

A.3.3 Implementación en FPGA de un compresor casi sin pérdidas de imágenes multi- e hiperespectrales basado en el estándar CCSDS 123.0-B-2

Este escenario de validación proporciona compresión de imágenes hiperespectrales en modo casi sin pérdidas. Para ello, el predictor CCSDS 123.0-B-2 casi sin pérdidas puede combinarse con cualquiera de los codificadores de entropía desarrollados. De esta manera, se proporciona una solución de compresión completa basada en el estándar CCSDS 123.0-B-2, que puede ser utilizada para demostrar la viabilidad de esta solución de compresión para diferentes programas espaciales que embarcan sensores espectroscópicos y necesitan de altos CRs.

Como ejemplo, se presenta el IP de compresión del instrumento CHIME. Como codificador entrópico, se utiliza el block-adaptive, ya que tiene una menor huella de área y un mayor throughput que el híbrido. Además, permite conseguir ratios de compresión por debajo de 1 bpp, razón por la que se ha descartado el codificador sample-adaptive para esta implementación.

Se ha seguido una estrategia basada en pipelining en la etapa de predicción, solapando el procesamiento de etapas consecutivas en ausencia de dependencias de datos y mejorando así el throughput. La selección de la suma local narrow neighbour-oriented junto con el modo de predicción reducido beneficia a este enfoque, ya que se eliminan las dependencias de datos con la muestra situada a la izquierda de la actual en la misma banda $s_{z,y,x-1}$, que es una restricción principal bajo el orden BIL, el elegido para esta implementación. La versión optimizada del predictor es capaz de procesar una muestra de entrada (16 bits de precisión) cada 11 ciclos de reloj a una frecuencia de 100 MHz. Esta característica está todavía lejos del objetivo final de 1 ciclo de reloj por muestra para CHIME, pero se espera que se consiga con una descripción puramente VHDL del predictor CCSDS 123.0-B-2. En cualquier caso, este modelo permite validar el comportamiento de la cadena de compresión propuesta como prototipo en etapas tempranas del flujo de diseño de la misión, y puede considerarse como el peor caso para una futura implementación VHDL.

Los resultados en términos de utilización de recursos se muestran en la Tabla A.2. Como se puede observar, el factor limitante es el uso de memoria del predictor, que está claramente condicionado por el tamaño de la imagen de entrada. Esto se debe a que todas las muestras preprocesadas de la línea espectral anterior ($N_x \cdot N_z$ muestras) deben ser almacenadas para calcular la predicción de la línea espectral actual. Para la implementación de CHIME, se

utilizan un total de 144 BRAMs sólo para almacenar esta memoria. No obstante, aunque el predictor se haya diseñado mediante técnicas HLS, la ocupación del hardware se considera aceptable, ya que el diseño se puede mapear satisfactoriamente en la FPGA objetivo.

TABLE A.2: CHIME IP - Utilización de recursos en Xilinx Kintex UltraScale XCKU040

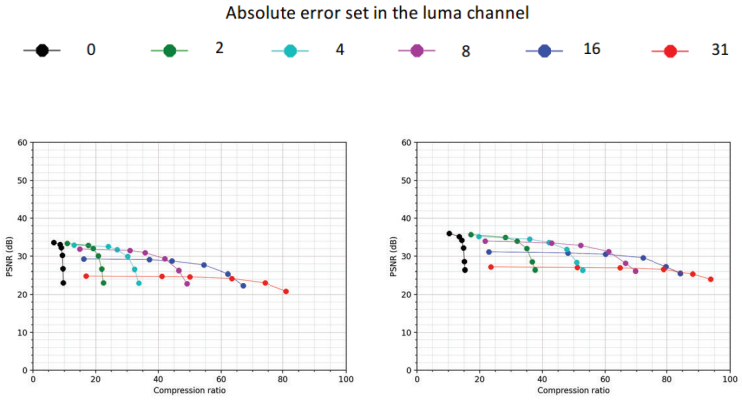
	36Kb BRAM	DSP48E	Registros	LUTs
Predictor (HLS)	167 (27.8%)	33 (1.7%)	7388 (1.5%)	9737 (4.0%)
Block-adaptive Encoder (VHDL)	4 (0.7%)	4 (0.2%)	1725 (0.4%)	4944 (2.0%)
Total	**171 (28.5%)**	**37 (1.9%)**	**9113 (1.9%)**	**14681 (6.0%)**

A.3.4 Aplicación del estándar CCSDS 123.0-B-2 para comprimir secuencias de vídeo monocromáticas y RGB

Finalmente, se ha evaluado el algoritmo de compresión sin pérdidas propuesto en el estándar CCSDS 123.0-B-2 para la compresión de vídeo. Este algoritmo puede adaptarse fácilmente para la compresión de vídeo monocromático, sustituyendo la dimensión espectral z definida en la norma por el dominio temporal t, utilizando para la inter-predicción las muestras vecinas en los P frames anteriores.

Para la compresión de vídeo RGB se ha explotado la correlación entre cada canal espectral en diferentes frames adquiridos consecutivamente en el dominio temporal. Se ha decidido añadir etapas de preprocesamiento adicionales para aumentar el rendimiento global de la compresión. Estas etapas realizan, por un lado, una transformación espectral del espacio de color RGB a YCbCr, concentrando la mayor parte de la información espacial en el canal de luma (Y), que representa el brillo de la escena, mientras que las bandas de croma (Cb y Cr) almacenan la información de color. De este modo, se puede introducir un mayor nivel de pérdidas en las componentes Cb y Cr, ya que almacenan información menos perceptible para el ojo humano, permitiendo así aumentar el ratio de compresión sin degradar la calidad del vídeo reconstruido. Por otro lado, se puede aplicar adicionalmente un submuestreo espacial de las bandas Cb y Cr, aumentando aún más el ratio de compresión conseguido a costa de introducir cierto desenfoque espacial en los canales Cb y Cr, sin introducir distorsiones significativas desde el punto de vista visual. En el enfoque propuesto, el tamaño de los canales Cb y Cr se ha reducido en un factor de 2 en sus dimensiones de anchura y altura, disminuyendo así la cantidad total de datos a procesar posteriormente en un factor de 4.

Fijando los límites de error para cada canal espectral en $A_Y = 4$ y $A_{Cb,Cr} = 8$ y aplicando las etapas de preprocesamiento propuestas, se consiguen CRs máximos de hasta 30 (es decir, alrededor de 0,27 bits por píxel) y 47 (es decir, 0,17 bits por píxel). Con esta configuración, la compresión aún no tiene efecto sobre la calidad del vídeo a partir de una inspección visual. Cuando se aumentan los límites de error a $A_Y = 8$ y $A_{Cb,Cr} = 16$, se consiguen mayores ratios de compresiónm alcanzando valores máximos de 66, mientras que la calidad de vídeo sigue siendo superior a 25 dB en ambos casos. Los resultados alcanzados se pueden observar en la Figura A.3 para las 4 secuencias de vídeo que forman parte del dataset.



a) *Guagua* sequence b) *Top* sequence

c) *DeathCircle* sequence d) *Nexus* sequence

FIGURE A.3: Relación entre el ratio de compresión y la calidad del vídeo decomprimido, medida en términos de PSNR, para la cadena de compresión para vídeo RGB completa

Desde el punto de vista visual, la degradación del vídeo se aprecia claramentepara los ratios de compresión más altos (por ejemplo, para $A_Y \geq 16$), aunque las formas de los objetos se siguen conservando. En la Figura A.4 se muestra un ejemplo para una de las secuencias de test, aplicando en cada caso diferentes niveles de error a los canales de luma y croma. La degradación no se aprecia visualmente hasta el tercer frame, cuando se observa una ligera degradación en la parte superior, aplicando valores de error de $A_Y = 16$ y $A_{Cb,Cr} = 4$. Con esta configuración, se obtiene un CR de 61 (es decir, 0,13 bits por píxel). Los valores de PSNR se sitúan en torno a los 30 dB.

El mayor nivel de degradación se observa en el último fotograma, cuando $A_Y = 31$ y $A_{Cb,Cr} = 8$. En este punto, las pérdidas de brillo se combinan con la saturación del color, obteniendo una secuencia distorsionada en la que los objetos se siguen identificando en la escena. Además, se observan algunos artefactos, relacionados con la forma en que se realiza la predicción por parte del algoritmo CCSDS 123.0-B-2. El nivel de degradación introducido por estos artefactos depende del error absoluto fijado en la predicción. Dado que se fijan errores más altos para los canales de croma, estos artefactos tienden a tener un mayor impacto en el color de la imagen que en las formas de la misma, que son preservadas por el canal de luma. Bajo esta configuración, se obtiene un ratio de compresión de 83 (es decir, aproximadamente 0,1 bits por píxel), con una PSNR alrededor de 25 dB.

a) A_Y=4 - A_C=2 b) A_Y=8 - A_C=2 c) A_Y=16 - A_C=4 d) A_Y=31 - A_C=8

FIGURE A.4: Frames decomprimidos de la secuencia de vídeo*DeathCircle* aplicando diferentes errores a los canales de luma y cromas

A.4 Conclusiones

Una vez resumido el trabajo realizado a lo largo de esta Tesis y los principales resultados alcanzados, se considera que los objetivos planteados al comienzo de su desarrollo se han cumplido satisfactoriamente. Se ha demostrado el concepto de solución de compresión modular mediante su aplicabilidad a cuatro escenarios de validación diferentes: compresión de datos unidimensionales sin pérdidas; compresión de imágenes hiperespectrales sin pérdidas y casi sin pérdidas; y compresión de vídeo casi sin pérdidas. La cadena de compresión propuesta para cada uno de los escenarios de validación se compone principalmente de un preprocesador basado en la predicción y un codificador entrópico, seleccionando la alternativa adecuada de entre los bloques funcionales desarrollados.

Se han desarrollado tres preprocesadores diferentes basados en la predicción, todos ellos conformes con los estándares CCSDS. El predictor de retardo unitario, basado en la norma CCSDS 121.0-B-3, se centra en la decorrelación de datos unidimensionales. Los predictores 3D que cumplen con los estándares CCSDS 123.0-B-1 y 123.0-B-2 proporcionan una decorrelación tanto espacial como espectral al procesar imágenes multi e hiperespectrales bajo compresión sin pérdidas y casi sin pérdidas, respectivamente.

También se han presentado tres enfoques de codificación entrópica que cumplen con las normas CCSDS para la compresión de datos a bordo. El codificador block-adaptive, definido originalmente en el estándar de compresión universal sin pérdidas CCSDS 121, se basa en la codificación Rice. Las alternativas del codificador sample-adaptive y del híbrido se presentan en el estándar de compresión CCSDS 123 para imágenes multi e hiperespectrales. Aunque ambos se basan en códigos Golomb, el codificador híbrido permite obtener mayores CRs al explotar la baja entropía de los datos de entrada.

Se proporcionan resultados de caracterización para escenarios de compresión sin pérdidas (es decir, datos unidimensionales e imágenes hiperespectrales), incluyendo la frecuencia de reloj máxima y la utilización de recursos en una variedad de FPGAs de grado espacial, incluyendo la novedosa familia BRAVE. El escenario de compresión de imágenes hiperespectrales casi sin pérdidas se ha validado con éxito en la FPGA Xilinx Kintex UltraScale XCKU040, ya que sus principales bloques funcionales se han diseñado siguiendo un flujo de diseño HLS y ese dispositivo es el que se empleará en el instrumento CHIME. Por último, se ha verificado el enfoque de compresión de vídeo a nivel algorítmico, demostrando la viabilidad de utilizar el algoritmo CCSDS 123.0-B-2 no sólo para comprimir imágenes multi e hiperespectrales, sino también secuencias de vídeo pancromáticas y RGB. Esto deriva en una solución de compresión versátil capaz de comprimir datos de diferente naturaleza en futuras misiones espaciales que embarquen diferentes tipos de sensores, reduciendo al mismo tiempo el área empleada.

Appendix B

Publications

B.1 Journals

The following publications are closely linked to the research goals of this Thesis:

[1] Y. Barrios, A. J. Sánchez, L. Santos and R. Sarmiento (2020). SHyLoC 2.0: A Versatile Hardware Solution for On-Board Data and Hyperspectral Image Compression on Future Space Missions. *IEEE Access*, vol. 8, pp. 54269-54287.

[2] L. A. Aranda, A. Sánchez, F. Garcia-Herrero, Y. Barrios, R. Sarmiento, and J. A. Maestro (2020). Reliability Analysis of the SHyLoC CCSDS123 IP Core for Lossless Hyperspectral Image Compression Using COTS FPGAs. *Electronics*, vol. 9, no. 10, p. 1681.

[3] Y. Barrios, A. Sánchez, R. Guerra and R. Sarmiento (2021). Hardware Implementation of the CCSDS 123.0-B-2 Near-Lossless Compression Standard Following an HLS Design Methodology. *Remote Sensing*, vol. 13, no. 21, p. 4388.

[4] Y. Barrios, R. Guerra, S. López and R. Sarmiento (2022). Performance Assessment of the CCSDS-123 Standard for Panchromatic Video Compression on Space Missions, in *IEEE Geoscience and Remote Sensing Letters*, vol. 19, Art no. 5507905, pp. 1-5.

[5] Y. Barrios, R. Guerra, S. López and R. Sarmiento (2022). Adaptation of the CCSDS 123.0-B-2 Standard for RGB and Multispectral Video Compression, in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 1656-1669.

In addition, other collaborations are summarised next:

[6] R. Guerra, Y. Barrios, M. Díaz, L. Santos, S. López, and R. Sarmiento (2018). A New Algorithm for the On-Board Compression of Hyperspectral Images. *Remote Sensing*, vol. 10, no. 3, p. 428.

[7] R. Guerra, Y. Barrios, M. Díaz, A. Baez, S. López and R. Sarmiento (2019). A Hardware-Friendly Hyperspectral Lossy Compressor for Next-Generation Space-Grade Field Programmable Gate Arrays. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 12, pp. 4813-4828.

[8] Y. Barrios, A. Rodríguez, A. Sánchez, A. Pérez, S. Lápez, A. Otero, E. de la Torre, and R. Sarmiento (2020). Lossy Hyperspectral Image Compression on a

Reconfigurable and Fault-Tolerant FPGA-Based Adaptive Computing Platform. *Electronics*, vol. 9, no. 10, p. 1576.

B.2 International Conferences

Different publications related to the work developed in this Thesis have been presented in relevant international conferences. These publications are listed below:

[1] A. Sánchez, Y. Barrios, L. Santos and R. Sarmiento (2018). SHyLoC-e: Improving CCSDS Standard Compliant IP Cores for On-Board Lossless Compression of Hyperspectral Images. In *6th International Workshop on On-Board Payload Data Compression (OBPDC)*, pp. 1-8.

[2] Y. Barrios, A. Sánchez, L. Santos, S. López, J. F. López and R. Sarmiento (2018). Hardware Implementation of the CCSDS 123.0-B-1 Lossless Multispectral and Hyperspectral Image Compression Standard by means of High Level Synthesis Tools. In *9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1-5.

[3] A. Sánchez, Y. Barrios, L. Santos and R. Sarmiento (2019). Evaluation of TMR effectiveness for soft error mitigation in SHyLoC compression IP core implemented on Zynq SoC under heavy ion radiation. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1-4.

[4] Y. Barrios, Antonio Sánchez, L. Santos and R. Sarmiento (2019). Implementation feasibility of a lossless data compression IP core compliant with the CCSDS 121.0-B-2 standard on space-qualified FPGAs. In *XXXIV Conference on Design of Circuits and Integrated Systems, DCIS 2019*, pp. 1-6.

[5] Antonio Sánchez, Y. Barrios, L. Santos and R. Sarmiento (2019). SHyLoC 2.0 CCSDS-123 IP: improving CCSDS 123.0-B-1 standard compliant IP core for on-board lossless compression of hyperspectral images. In *XXXIV Conference on Design of Circuits and Integrated Systems, DCIS 2019*, pp. 1-6.

[6] Y. Barrios, P. Rodríguez, A. Sánchez, M.I. González, L. Berrojo and R. Sarmiento (2020). Implementation of Cloud Detection and Processing Algorithms and CCSDS-Compliant Hyperspectral Image Compression for CHIME Mission. In *7th International Workshop on On-Board Payload Data Compression (OBPDC)*, pp. 1-8.

[7] D. Ventura, Y. Barrios, A. Sánchez and R. Sarmiento (2021). EDAC implementation on a data lossless compressor compliant with the CCSDS 121.0-B-3 standard. In *XXXVI Conference on Design of Circuits and Integrated Systems (DCIS) 2021*, pp. 1-6, doi: 10.1109/DCIS53048.2021.9666189.

In addition, other collaborations are summarised next:

[8] Y. Barrios, Antonio Sánchez, Roberto Sarmiento, Alfonso Rodríguez, Andrés Otero, and Eduardo de la Torre (2018). Hyperspectral Image Lossy Compression on a Reconfigurable and Fault-Tolerant Architecture Implemented over a Commercial Off-The-Shelf FPGA-based System-on-Chip. In *6th International Workshop on On-Board Payload Data Compression (OBPDC)*, pp. 1-7.

[9] R. Guerra, M. Díaz, Y. Barrios, S. López and R. Sarmiento (2018). A Hardware-Friendly Algorithm for the On-Board Compression of Hyperspectral Images. In *9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1-5.

[10] R. Guerra, M. Díaz, Y. Barrios, S. López and R. Sarmiento (2018). A hardware-friendly algorithm for compressing hyperspectral images. In *Proc. SPIE 10792, High-Performance Computing in Geoscience and Remote Sensing VIII*, 1079208.

B.3 Book Chapters

[1] L. Armesto Caride, A. Rodríguez, A. Pérez, S. Sáez, J. Valls, Y. Barrios, A.J. Sánchez, D. González Arjona, A.J. P-Herrera, F Veljkovic (2020). Reconfigurable Video Processor for Space. In: A. Leitner, D. Watzenig, J. Ibanez-Guzman (eds) *Validation and Verification of Automated Systems*. Springer, Cham.

References

[1] Bing Lu, Phuong Dao, Jiangui Liu, Yuhong He, and Jiali Shang. Recent advances of hyperspectral imaging technology and applications in agriculture. *Remote Sensing*, 12(16):2659, Aug 2020. ISSN 2072-4292. doi: 10.3390/rs12162659. URL http://dx.doi.org/10.3390/rs12162659.

[2] E. Raymond Hunt Jr. and Craig S. T. Daughtry. What good are unmanned aircraft systems for agricultural remote sensing and precision agriculture? *International Journal of Remote Sensing*, 39(15-16):5345–5376, 2018. doi: 10.1080/01431161.2017.1410300. URL https://doi.org/10.1080/01431161.2017.141030.

[3] Sima Peyghambari and Yun Zhang. Hyperspectral remote sensing in lithological mapping, mineral exploration, and environmental geology: an updated review. *Journal of Applied Remote Sensing*, 15(3):1 – 25, 2021. doi: 10.1117/1.JRS.15.031501. URL https://doi.org/10.1117/1.JRS.15.031501.

[4] Luis Arias, Jose Cifuentes, Milton Marín, Fernando Castillo, and Hugo Garcés. Hyperspectral Imaging Retrieval Using MODIS Satellite Sensors Applied to Volcanic Ash Clouds Monitoring. *Remote Sensing*, 11(11):1393, Jun 2019. ISSN 2072-4292. doi: 10.3390/rs11111393. URL http://dx.doi.org/10.3390/rs11111393.

[5] Stefania Amici and Alessandro Piscini. Exploring PRISMA Scene for Fire Detection: Case Study of 2019 Bushfires in Ben Halls Gap National Park, NSW, Australia. *Remote Sensing*, 13(8):1410, Apr 2021. ISSN 2072-4292. doi: 10.3390/rs13081410. URL http://dx.doi.org/10.3390/rs13081410.

[6] Milad Niroumand-Jadidi, Francesca Bovolo, and Lorenzo Bruzzone. Water Quality Retrieval from PRISMA Hyperspectral Images: First Experience in a Turbid Lake and Comparison with Sentinel-2. *Remote Sensing*, 12(23):3984, Dec 2020. ISSN 2072-4292. doi: 10.3390/rs12233984. URL http://dx.doi.org/10.3390/rs12233984.

[7] Pauliina Salmi, Matti A. Eskelinen, Matti T. Leppänen, and Ilkka Pölönen. Rapid Quantification of Microalgae Growth with Hyperspectral Camera and Vegetation Indices. *Plants*, 10(2):341, Feb 2021. ISSN 2223-7747. doi: 10.3390/plants10020341. URL http://dx.doi.org/10.3390/plants10020341.

[8] Jun Liu, Bin Luo, Sylvain Douté, and Jocelyn Chanussot. Exploration of Planetary Hyperspectral Images with Unsupervised Spectral Unmixing: A Case Study of Planet Mars. *Remote Sensing*, 10(5), 2018. ISSN 2072-4292. doi: 10.3390/rs10050737. URL https://www.mdpi.com/2072-4292/10/5/737.

[9] Zhiping He, Rui Xu, Chunlai Li, Liyin Yuan, Chengyu Liu, Gang Lv, Jian Jin, Jianan Xie, Chuifeng Kong, Feifei Li, Xiaowen Chen, Rong Wang, Sheng Xu, Wei Pan, Jincai Wu, Changkun Li, Wang Tianhong, Haijun Jin, Hourui Chen, and Jianyu Wang. Mars Mineralogical Spectrometer (MMS) on the Tianwen-1 Mission. *Space Science Reviews*, 217, 03 2021. doi: 10.1007/s11214-021-00804-z.

[10] Michael A. Wulder, Thomas R. Loveland, David P. Roy, Christopher J. Crawford, Jeffrey G. Masek, Curtis E. Woodcock, Richard G. Allen, Martha C. Anderson, Alan S. Belward, Warren B. Cohen, John Dwyer, Angela Erb, Feng Gao, Patrick Griffiths, Dennis Helder, Txomin Hermosilla, James D. Hipple, Patrick Hostert, M. Joseph Hughes, Justin Huntington, David M. Johnson, Robert Kennedy, Ayse Kilic, Zhan Li, Leo Lymburner, Joel McCorkel, Nima Pahlevan, Theodore A. Scambos, Crystal Schaaf, John R. Schott, Yongwei Sheng, James Storey, Eric Vermote, James Vogelmann, Joanne C. White, Randolph H. Wynne, and Zhe Zhu. Current status of Landsat program, science, and applications. *Remote Sensing of Environment*, 225: 127–147, 2019. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2019.02.015. URL https://www.sciencedirect.com/science/article/pii/S0034425719300707.

[11] J.S. Pearlman, P.S. Barry, C.C. Segal, J. Shepanski, D. Beiso, and S.L. Carman. Hyperion, a space-based imaging spectrometer. *IEEE Transactions on Geoscience and Remote Sensing*, 41(6):1160–1173, 2003. doi: 10.1109/TGRS.2003.815018.

[12] Christine M. Lee, Morgan L. Cable, Simon J. Hook, Robert O. Green, Susan L. Ustin, Daniel J. Mandl, and Elizabeth M. Middleton. An introduction to the NASA Hyperspectral InfraRed Imager (HyspIRI) mission and preparatory activities. *Remote Sensing of Environment*, 167:6–19, 2015. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2015.06.012. URL https://www.sciencedirect.com/science/

article/pii/S0034425715300419. Special Issue on the Hyperspectral Infrared Imager (HyspIRI).

[13] Wenxue Fu, Jianwen Ma, Pei Chen, and Fang Chen. *Remote Sensing Satellites for Digital Earth*, pages 68–136. Springer Link, 2020. ISBN 978-981-329-915-3. doi: https://doi.org/10.1007/978-981-32-9915-3. URL https://link.springer.com/book/10.1007/978-981-32-9915-3.

[14] J. Nieke and M. Rast. Status: Copernicus Hyperspectral Imaging Mission For The Environment (CHIME). In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 4609–4611, 2019. doi: 10.1109/IGARSS.2019.8899807.

[15] Sylvain Michel, Philippe Gamet, and Marie-José Lefevre-Fonollosa. HYPXIM — A hyperspectral satellite defined for science, security and defence users. In *2011 3rd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4, 2011. doi: 10.1109/WHISPERS.2011.6080864.

[16] Elia Vangi, Giovanni D'Amico, Saverio Francini, Francesca Giannetti, Bruno Lasserre, Marco Marchetti, and Gherardo Chirici. The New Hyperspectral Satellite PRISMA: Imagery for Forest Types Discrimination. *Sensors*, 21(4):1182, Feb 2021. ISSN 1424-8220. doi: 10.3390/s21041182. URL http://dx.doi.org/10.3390/s21041182.

[17] Tal Feingersh and Eyal Ben Dor. *SHALOM – A Commercial Hyperspectral Space Mission*, chapter 11, pages 247–263. John Wiley & Sons, Ltd, 2015. ISBN 9781118945179. doi: https://doi.org/10.1002/9781118945179.ch11. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118945179.ch11.

[18] Manuela Sornig, Sebastian Fischer, Christian Chlebek, Martin Muecke, Hans-Peter Honold, and Brian Heider. The hyperspectral instrument onboard EnMAP: overview and current status. In Zoran Sodnik, Nikos Karafolas, and Bruno Cugny, editors, *International Conference on Space Optics — ICSO 2018*, volume 11180, pages 81 – 90. International Society for Optics and Photonics, SPIE, 2019. URL https://doi.org/10.1117/12.2535926.

[19] Tsuneo Matsunaga, Akira Iwasaki, Satoshi Tsuchida, Jun Tanii, Osamu Kashimura, Ryosuke Nakamura, Hirokazu Yamamoto, Tetsushi Tachikawa, and Shuichi Rokugawa. Current status of Hyperspectral Imager Suite (HISUI). In *2013 IEEE International*

Geoscience and Remote Sensing Symposium - IGARSS, pages 3510–3513, 2013. doi: 10.1109/IGARSS.2013.6723586.

[20] Qingxi Tong, Yongqi Xue, and Lifu Zhang. Progress in Hyperspectral Remote Sensing Science and Technology in China Over the Past Three Decades. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(1): 70–91, 2014. doi: 10.1109/JSTARS.2013.2267204.

[21] Yanfei Zhong, Xinyu Wang, Shaoyu Wang, and Liangpei Zhang. Advances in spaceborne hyperspectral remote sensing in China. *Geo-spatial Information Science*, 24(1):95–120, 2021. doi: 10.1080/10095020.2020.1860653. URL https://doi.org/10.1080/10095020.2020.1860653.

[22] Planet Labs. *SkySat Imagery. Product Specification*, May 2018. https://assets.planet.com/marketing/PDF/SkySat-Ortho-Scene-Product-Spec-Sheet.pdf.

[23] Guo Huadong, Dong Liang, and Liu Guang. Progress of Earth Observation in China. *Chinese Journal of Space Science*, pages 908–919, 10 2020. doi: 10.11728/cjss2020.05.908.

[24] Wentao Li, Fang Gao, Peng Zhang, Yihui Li, Yuan An, Xing Zhong, and Qing Lu. Research on Multiview Stereo Mapping Based on Satellite Video Images. *IEEE Access*, 9:44069–44083, 2021. doi: 10.1109/ACCESS.2021.3059487.

[25] Dong-Hyun Cho, Won Sub Choi, Min-Ki Kim, Jin-Hyung Kim, Eunsup Sim, and Hae-Dong Kim. High-Resolution Image and Video CubeSat (HiREV): Development of Space Technology Test Platform Using a Low-Cost CubeSat Platform. *International Journal of Aerospace Engineering*, 2019:1–17, 05 2019. doi: 10.1155/2019/8916416.

[26] Carole Thiebaut and Roberto Camarero. *CNES Studies for On-Board Compression of High-Resolution Satellite Images*, pages 29–46. Springer New York, New York, NY, 2011. ISBN 978-1-4614-1183-3. doi: 10.1007/978-1-4614-1183-3_2. URL https://doi.org/10.1007/978-1-4614-1183-3_2.

[27] Consultative Committee for Space Data Systems. *Lossless Data Compression, Recommended Standard CCSDS 121.0-B-3*. CCSDS, August 2020. Blue Book.

[28] Consultative Committee for Space Data Systems. *Image Data Compression, Recommended Standard CCSDS 122.0-B-2*. CCSDS, September 2017. Blue Book.

[29] Consultative Committee for Space Data Systems. *Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression, Recommended Standard CCSDS 122.1-B-1.* CCSDS, September 2017. Blue Book.

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

[31] Consultative Committee for Space Data Systems. *Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression, CCSDS 123.0-B-2*, volume 2. CCSDS, blue book edition, February 2019.

[32] Shen-En Qian. *Hyperspectral Satellites and System Design.* CRC Press, 04 2020. ISBN 9780429266201. doi: 10.1201/9780429266201.

[33] José-Antonio Martínez-Heras, David Evans, and Rainer Timm. Housekeeping Telemetry Compression: When, How and Why Bother? In *2009 First International Conference on Advances in Satellite and Space Communications*, pages 35–40, 2009. doi: 10.1109/SPACOMM.2009.30.

[34] David J. Evans and Alessandro Donati. The ESA POCKET+ housekeeping telemetry compression algorithm: Why make spacecraft operations harder than it already is? In *SpaceOps Conference*, pages 1–15, 2018. doi: 10.2514/6.2018-2613. URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-2613.

[35] Stephen M. Trimberger. Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology. *Proceedings of the IEEE*, 103(3):318–331, 2015. doi: 10.1109/JPROC.2015.2392104.

[36] NanoXplore. From eFPGA cores to RHBD System-on-Chip FPGA. In *SEFUW: SpacE FPGA Users Workshop, 4th Edition*, 2018. URL https://indico.esa.int/event/232/contributions/2137/attachments/1820/2121/2018-04_NX-From_eFPGA_cores_to_RHBH_SoC_FPGAs-JLM-v2.pdf.

[37] MarketsandMarkets. FPGA Market with COVID-19 Impact Analysis by Configuration, Technology, Node Size, Vertical and Region - Forecast to 2026. https://www.marketsandmarkets.com/Market-Reports/fpga-market-194123367.html, 2021. Accessed: 2022-01-28.

[38] M. Wirthlin. High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond. *Proceedings of the IEEE*, 103(3):379–389, March 2015. ISSN 0018-9219. doi: 10.1109/JPROC.2015.2404212.

[39] A. D. George and C. M. Wilson. Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites. *Proceedings of the IEEE*, 106(3):458–470, March 2018. ISSN 0018-9219. doi: 10.1109/JPROC.2018.2802438.

[40] European Cooperation for Space Standardization. *ASIC and FPGA development, Space product assurance (ECSS-Q-ST-60-02C)*. ECSS, July 2008.

[41] Pong P. Chu. *Introduction to Digital System Design*, pages 1–22. IEEE Press, 2006. doi: 10.1002/0471786411.ch1.

[42] Thomas Schuster, Rolf Meyer, Rainer Buchty, Luca Fossati, and Mladen Berekovic. SoCRocket - A virtual platform for the European Space Agency's SoC development. In *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–7, 2014. doi: 10.1109/ReCoSoC.2014.6860690.

[43] J. Zhu and N. Dutt. *Electronic Design Automation. Synthesis, Verification, and Test*, chapter Electronic System-Level design and High-Level Synthesis, pages 235–298. Elsevier, 2009.

[44] Philippe Coussy, Daniel D. Gajski, Michael Meredith, and Andres Takach. An Introduction to High-Level Synthesis. *IEEE Design Test of Computers*, 26(4):8–17, 2009. doi: 10.1109/MDT.2009.69.

[45] Benjamin Carrion Schafer and Zi Wang. High-Level Synthesis Design Space Exploration: Past, Present, and Future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2628–2639, 2020. doi: 10.1109/TCAD.2019.2943570.

[46] Jason Cong, Bin Liu, Stephen Neuendorffer, Juanjo Noguera, Kees Vissers, and Zhiru Zhang. High-Level Synthesis for FPGAs: From Prototyping to Deployment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(4):473–491, 2011. doi: 10.1109/TCAD.2011.2110592.

[47] Razvan Nane, Vlad-Mihai Sima, Christian Pilato, Jongsok Choi, Blair Fort, Andrew Canis, Yu Ting Chen, Hsuan Hsiao, Stephen Brown, Fabrizio Ferrandi, Jason Anderson, and Koen Bertels. A Survey and Evaluation of FPGA High-Level Synthesis

Tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(10):1591–1604, 2016. doi: 10.1109/TCAD.2015.2513673.

[48] Sakari Lahti, Panu Sjövall, Jarno Vanne, and Timo D. Hämäläinen. Are We There Yet? A Study on the State of High-Level Synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(5):898–911, 2019. doi: 10.1109/TCAD.2018.2834439.

[49] Alexander Goetz, Gregg Vane, Jerry Solomon, and Barrett Rock. Imaging Spectrometry for Earth Remote Sensing. *Science (New York, N.Y.)*, 228:1147–53, 07 1985. doi: 10.1126/science.228.4704.1147.

[50] Pasquale Maglione. Very High Resolution optical satellites: an overview of the most commonly used. *American Journal of Applied Sciences*, 13:91–99, 01 2016. doi: 10.3844/ajassp.2016.91.99.

[51] María Lucana Santos Falcón. *Hyperspectral image compression onboard next-generation satellites. Implementation solutions on GPU and FPGAs*. PhD thesis, ULPGC, Universidad de Las Palmas de Gran Canaria, 2014. URL https://accedacris.ulpgc.es/handle/10553/13000?mode=full.

[52] Xun Cao, Tao Yue, Xing Lin, Stephen Lin, Xin Yuan, Qionghai Dai, Lawrence Carin, and David J. Brady. Computational Snapshot Multispectral Cameras: Toward dynamic capture of the spectral world. *IEEE Signal Processing Magazine*, 33(5): 95–108, 2016. doi: 10.1109/MSP.2016.2582378.

[53] Nathan A. Hagen and Michael W. Kudenov. Review of snapshot spectral imaging technologies. *Optical Engineering*, 52(9):1 – 23, 2013. doi: 10.1117/1.OE.52.9.090901. URL https://doi.org/10.1117/1.OE.52.9.090901.

[54] Manfred Ehlers, Sascha Klonus, Pär Johan Åstrand, and Pablo Rosso. Multi-sensor image fusion for pansharpening in remote sensing. *International Journal of Image and Data Fusion*, 1(1):25–45, 2010. doi: 10.1080/19479830903561985. URL https://doi.org/10.1080/19479830903561985.

[55] Farzaneh Dadrass Javan, Farhad Samadzadegan, Soroosh Mehravar, Ahmad Toosi, Reza Khatami, and Alfred Stein. A review of image fusion techniques for pan-sharpening of high-resolution satellite imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 171:101–117, 2021. ISSN 0924-2716. doi: https://doi.org/

10.1016/j.isprsjprs.2020.11.001. URL https://www.sciencedirect.com/science/article/pii/S0924271620303002.

[56] Tomasz Bieliński. A Parallax Shift Effect Correction Based on Cloud Height for Geostationary Satellites and Radar Observations. *Remote Sensing*, 12(3):365, Jan 2020. ISSN 2072-4292. doi: 10.3390/rs12030365. URL http://dx.doi.org/10.3390/rs12030365.

[57] E. Christophe. Hyperspectral data compression tradeoff. In S. Prasad, L. M. Bruce, and J. Chanussot, editors, *Optical Remote Sensing, Advances in Signal Processing and Exploitation Techniques*, chapter 2, pages 9–29. Springer, 2011.

[58] ESA. SHyLoC IP core. https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Microelectronics/SHyLoC_IP_Core, 2018. Accessed: 2021-03-19.

[59] L. Santos, A. Gomez, and R. Sarmiento. Implementation of CCSDS standards for lossless multispectral and hyperspectral satellite image compression. *IEEE Transactions on Aerospace and Electronic Systems*, pages 1–1, 2019. ISSN 0018-9251. doi: 10.1109/TAES.2019.2929971.

[60] Y. Barrios, A. J. Sánchez, L. Santos, and R. Sarmiento. SHyLoC 2.0: A Versatile Hardware Solution for On-Board Data and Hyperspectral Image Compression on Future Space Missions. *IEEE Access*, 8:54269–54287, 2020. doi: 10.1109/ACCESS.2020.2980767.

[61] J Nieke and M Rast. Towards the Copernicus Hyperspectral Imaging Mission for the Environment (CHIME). In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 157–159, 07 2018. doi: 10.1109/IGARSS.2018.8518384.

[62] Yubal Barrios, Antonio Sánchez, Raúl Guerra, and Roberto Sarmiento. Hardware Implementation of the CCSDS 123.0-B-2 Near-Lossless Compression Standard Following an HLS Design Methodology. *Remote Sensing*, 13(21):4388, Oct 2021. ISSN 2072-4292. doi: 10.3390/rs13214388. URL http://dx.doi.org/10.3390/rs13214388.

[63] VIDEO Consortium. Video Imaging Demonstrator for Earth Observation. https://video-h2020.eu. Accessed: 2021-12-09.

[64] Aday García del Toro. *Implementaciones hardware usando nuevas técnicas de diseño de sistemas de compresión con pérdidas de imágenes hiperespectrales para futuras*

misiones espaciales. PhD thesis, ULPGC, Universidad de Las Palmas de Gran Canaria, 2017. URL https://accedacris.ulpgc.es/handle/10553/54042.

[65] Raúl Guerra, Yubal Barrios, María Díaz, Lucana Santos, Sebastián López, and Roberto Sarmiento. A new algorithm for the on-board compression of hyperspectral images. *Remote Sensing*, 10(3):428, Mar 2018. ISSN 2072-4292. doi: 10.3390/rs10030428. URL http://dx.doi.org/10.3390/rs10030428.

[66] R. Guerra, Y. Barrios, M. Díaz, A. Baez, S. López, and R. Sarmiento. A hardware-friendly hyperspectral lossy compressor for next-generation space-grade field programmable gate arrays. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(12):4813–4828, 2019. doi: 10.1109/JSTARS.2019.2919791.

[67] Julián Caba, María Díaz, Jesús Barba, Raúl Guerra, Jose A. de la Torre López, and Sebastián. FPGA-Based On-Board Hyperspectral Imaging Compression: Benchmarking Performance and Energy Efficiency against GPU Implementations. *Remote Sensing*, 12(22):3741, Nov 2020. ISSN 2072-4292. doi: 10.3390/rs12223741. URL http://dx.doi.org/10.3390/rs12223741.

[68] Guoxia Yu, Tanya Vladimirova, and Martin N. Sweeting. Image compression systems on board satellites. *Acta Astronautica*, 64(9):988 – 1005, 2009. ISSN 0094-5765. doi: https://doi.org/10.1016/j.actaastro.2008.12.006. URL http://www.sciencedirect.com/science/article/pii/S0094576508004062.

[69] Swetha Vura, Premjyoti Patil, and Shantakumar B. Patil. A study of different compression algorithms for multispectral images. *Materials Today: Proceedings*, 2021. ISSN 2214-7853. doi: https://doi.org/10.1016/j.matpr.2021.06.175. URL https://www.sciencedirect.com/science/article/pii/S2214785321045417.

[70] N. Beser. Space data compression standards. *Johns Hopkins APL Tech. Dig.*, 15: 206–223, September 1994.

[71] N. Moayeri. A low-complexity, fixed-rate compression scheme for color images and documents. 1998.

[72] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. doi: 10.1109/JRPROC.1952.273898.

[73] Robert F Rice. Some practical universal noiseless coding techniques. *JPL Publication*, 1979.

[74] J. Capon. A probabilistic model for run-length coding of pictures. *IRE Transactions on Information Theory*, 5(4):157–163, 1959. doi: 10.1109/TIT.1959.1057512.

[75] Paul G. Howard and Jeffrey Scott Vitter. *Arithmetic Coding for Data Compression*, pages 65–68. Springer US, Boston, MA, 2008. ISBN 978-0-387-30162-4. doi: 10.1007/978-0-387-30162-4_34. URL https://doi.org/10.1007/978-0-387-30162-4_34.

[76] A.J. Hussain, Ali Al-Fayadh, and Naeem Radi. Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, 300:44 – 69, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.02.094. URL http://www.sciencedirect.com/science/article/pii/S0925231218302935.

[77] J. Portell, R. Iudica, E. García-Berro, A. G. Villafranca, and G. Artigues. FAPEC, a versatile and efficient data compressor for space missions. *International Journal of Remote Sensing*, 39(7):2022–2042, 2018. doi: 10.1080/01431161.2017.1399478. URL https://doi.org/10.1080/01431161.2017.1399478.

[78] X. Wu and N. Memon. Context-based, adaptive, lossless image coding. *IEEE Transactions on Communications*, 45(4):437–444, 1997. doi: 10.1109/26.585919.

[79] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, June 1984. ISSN 0018-9162. doi: 10.1109/MC.1984.1659158. URL https://doi.org/10.1109/MC.1984.1659158.

[80] X. Ma, C. Xu, P. Zhang, and C. Hu. The Application of the Improved LZW Algorithm in the Data Processing of GNSS Simulation. In *2012 Fourth International Conference on Computational and Information Sciences*, pages 160–163, 2012. doi: 10.1109/ICCIS.2012.319.

[81] J. T. Rucker, J. E. Fowler, and N. H. Younan. JPEG2000 coding strategies for hyperspectral data. In *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05.*, volume 1, pages 4 pp.–, 2005. doi: 10.1109/IGARSS.2005.1526121.

[82] M. J. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, 9(8):1309–1324, 2000. doi: 10.1109/83.855427.

[83] Xiaolin Wu and N. Memon. Context-based lossless interband compression-extending CALIC. *IEEE Transactions on Image Processing*, 9(6):994–1001, 2000. doi: 10.1109/83.846242.

[84] F. Chen, S. Sahni, and B. C. Vemuri. Efficient Algorithms for Lossless Compression of 2D/3D Images. In Dionysius P. Huijsmans and Arnold W. M. Smeulders, editors, *Visual Information and Information Systems*, pages 683–690, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48762-3.

[85] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, 1996. doi: 10.1109/76.499834.

[86] Jean-Michel Gaucel, Carole Thiebaut, Romain Hugues, and Roberto Camarero. On-board compression of hyperspectral satellite data using band-reordering. In Bormin Huang, Antonio J. Plaza, and Carole Thiebaut, editors, *Satellite Data Compression, Communications, and Processing VII*, volume 8157, pages 213 – 224. International Society for Optics and Photonics, SPIE, 2011. doi: 10.1117/12.893881. URL https://doi.org/10.1117/12.893881.

[87] M. I. Afjal, M. A. Mamun, and M. P. Uddin. Weighted-Correlation based Band Reordering Heuristics for Lossless Compression of Remote Sensing Hyperspectral Sounder Data. In *2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*, pages 1–4, 2018. doi: 10.1109/ICAEEE.2018.8642975.

[88] M. J. Ryan and J. F. Arnold. The lossless compression of AVIRIS images by vector quantization. *IEEE Transactions on Geoscience and Remote Sensing*, 35(3):546–550, 1997. doi: 10.1109/36.581964.

[89] Shen-En Qian, A. B. Hollinger, D. Williams, and D. Manak. Vector quantization using spectral index-based multiple subcodebooks for hyperspectral data compression. *IEEE Transactions on Geoscience and Remote Sensing*, 38(3):1183–1190, 2000. doi: 10.1109/36.843010.

[90] M. R. Pickering and M. J. Ryan. Efficient spatial-spectral compression of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 39(7):1536–1539, 2001. doi: 10.1109/36.934084.

[91] Daniel Báscones, Carlos González, and Daniel Mozos. Hyperspectral Image Compression Using Vector Quantization, PCA and JPEG2000. *Remote Sensing*, 10(6):907, Jun 2018. ISSN 2072-4292. doi: 10.3390/rs10060907. URL http://dx.doi.org/10.3390/rs10060907.

[92] Filipe Magalhaes, Francisco M. Araújo, Miguel Correia, Mehrdad Abolbashari, and Faramarz Farahi. High-resolution hyperspectral single-pixel imaging system based on compressive sensing. *Optical Engineering*, 51(7):1 – 7, 2012. doi: 10.1117/1.OE.51.7.071406. URL https://doi.org/10.1117/1.OE.51.7.071406.

[93] K. S. Gunasheela and H. S. Prasantha. Compressive Sensing Approach to Satellite Hyperspectral Image Compression. In Suresh Chandra Satapathy and Amit Joshi, editors, *Information and Communication Technology for Intelligent Systems*, pages 495–503, Singapore, 2019. Springer Singapore. ISBN 978-981-13-1742-2.

[94] Saurabh Kumar, Subhasis Chaudhuri, Biplab Banerjee, and Feroz Ali. Onboard Hyperspectral Image Compression Using Compressed Sensing and Deep Learning. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 30–42, Cham, 2019. Springer International Publishing. ISBN 978-3-030-11012-3.

[95] G. Coluccia, C. Lastri, D. Guzzi, E. Magli, V. Nardino, L. Palombi, I. Pippi, V. Raimondi, C. Ravazzi, F. Garoi, D. Coltuc, R. Vitulli, and A. Z. Marchi. Optical compressive imaging technologies for space big data. *IEEE Transactions on Big Data*, 6(3):430–442, 2020. doi: 10.1109/TBDATA.2019.2907135.

[96] A. Karami, S. Beheshti, and M. Yazdi. Hyperspectral image compression using 3D discrete cosine transform and support vector machine learning. In *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pages 809–812, 2012. doi: 10.1109/ISSPA.2012.6310664.

[97] Jin Li and Zilong Liu. Multispectral Transforms Using Convolution Neural Networks for Remote Sensing Multispectral Image Compression. *Remote Sensing*, 11(7):759, Mar 2019. ISSN 2072-4292. doi: 10.3390/rs11070759. URL http://dx.doi.org/10.3390/rs11070759.

[98] D. Valsesia and E. Magli. High-Throughput Onboard Hyperspectral Image Compression With Ground-Based CNN Reconstruction. *IEEE Transactions on Geoscience and Remote Sensing*, 57(12):9544–9553, 2019. doi: 10.1109/TGRS.2019.2927434.

[99] Diego Wildenstein and Alan D. George. Towards Intelligent Compression of Hyperspectral Imagery. In *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 01–06, 2021. doi: 10.1109/CONECCT52877.2021.9622585.

[100] Miguel Hernández-Cabronero, Jordi Portell, Ian Blanes, and Joan Serra-Sagristà. High-Performance Lossless Compression of Hyperspectral Remote Sensing Scenes Based on Spectral Decorrelation. *Remote Sensing*, 12(18):2955, Sep 2020. ISSN 2072-4292. doi: 10.3390/rs12182955. URL http://dx.doi.org/10.3390/rs12182955.

[101] Andrea Abrardo, Mauro Barni, Andrea Bertoli, Raoul Grimoldi, Enrico Magli, and Raffaele Vitulli. *Low-Complexity Approaches for Lossless and Near-Lossless Hyperspectral Image Compression*, pages 47–65. Springer New York, New York, NY, 2011. ISBN 978-1-4614-1183-3. doi: 10.1007/978-1-4614-1183-3_3. URL https://doi.org/10.1007/978-1-4614-1183-3_3.

[102] M. Conoscenti, R. Coppola, and E. Magli. Constant SNR, Rate Control, and Entropy Coding for Predictive Lossy Hyperspectral Image Compression. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7431–7441, 2016. doi: 10.1109/TGRS.2016.2603998.

[103] D. Valsesia and E. Magli. Fast and Lightweight Rate Control for Onboard Predictive Coding of Hyperspectral Images. *IEEE Geoscience and Remote Sensing Letters*, 14 (3):394–398, 2017. doi: 10.1109/LGRS.2016.2644726.

[104] M. Klimesh. Low-Complexity Lossless Compression of Hyperspectral Imagery Via Adaptive Filtering. *Interplanetary Network Progress Report*, 01 2005.

[105] A. Gersho. Adaptive filtering with binary reinforcement. *IEEE Transactions on Information Theory*, 30(2):191–199, 1984. doi: 10.1109/TIT.1984.1056890.

[106] D. Keymeulen, D. Dolman, S. Shin, J. Riddley, M. Klimesh, A. Kiely, D.R. Thompson, M. Cheng, S. Dolinar, E. Liggett, P. Sullivan, M. Bernas, K. Roth, C. Holyoake, K. Crocker, and A. Smith. High Performance Space Data Acquisition, Clouds Screening and Data Compression with modified COTS Embedded System-on-Chip Instrument Avionics for Space-based Next Generation Imaging Spectrometers (NGIS). In *6th International Workshop on On-Board Payload Data Compression (OBPDC)*, pages 1–25, September 2018.

[107] Xiaolin Wu and N. Memon. Context-based lossless interband compression-extending CALIC. *IEEE Transactions on Image Processing*, 9(6):994–1001, 2000. doi: 10.1109/83.846242.

[108] E. Magli, G. Olmo, and E. Quacchio. Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC. *IEEE Geoscience and Remote Sensing Letters*, 1(1):21–25, 2004. doi: 10.1109/LGRS.2003.822312.

[109] J. Mielikainen. Lossless compression of hyperspectral images using lookup tables. *IEEE Signal Processing Letters*, 13(3):157–160, 2006. doi: 10.1109/LSP.2005.862604.

[110] Jarno Mielikainen and Pekka Toivanen. Lossless Compression of Hyperspectral Images Using a Quantized Index to Lookup Tables. *IEEE Geoscience and Remote Sensing Letters*, 5(3):474–478, 2008. doi: 10.1109/LGRS.2008.917598.

[111] Jarno Mielikainen. *Lookup-Table Based Hyperspectral Data Compression*, pages 169–184. Springer New York, New York, NY, 2011. ISBN 978-1-4614-1183-3. doi: 10.1007/978-1-4614-1183-3_8. URL https://doi.org/10.1007/978-1-4614-1183-3_8.

[112] J. Mielikainen and B. Huang. Lossless Compression of Hyperspectral Images Using Clustered Linear Prediction With Adaptive Prediction Length. *IEEE Geoscience and Remote Sensing Letters*, 9(6):1118–1121, 2012. doi: 10.1109/LGRS.2012.2191531.

[113] Hongda Shen, Zhuocheng Jiang, and W. Pan. Efficient Lossless Compression of Multitemporal Hyperspectral Image Data. *Journal of Imaging*, 4(12):142, Dec 2018. ISSN 2313-433X. doi: 10.3390/jimaging4120142. URL http://dx.doi.org/10.3390/jimaging4120142.

[114] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of JPEG-2000. In *Proceedings DCC 2000. Data Compression Conference*, pages 523–541, March 2000. doi: 10.1109/DCC.2000.838192.

[115] International Telecommunication Union. *Advanced video coding for generic audiovisual services, Recommendation ITU-T H.264*. ITU-T, June 2019. Series H: Audiovisual and Multimedia Systems. Infrastructure of audiovisual services, coding of moving video.

[116] Zixiang Xiong, K. Ramchandran, M. T. Orchard, and Ya-Qin Zhang. A comparative study of DCT- and wavelet-based image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(5):692–695, 1999. doi: 10.1109/76.780358.

[117] David S. Taubman and Michael W. Marcellin. *Introduction to JPEG2000*, pages 399–415. Springer US, Boston, MA, 2002. ISBN 978-1-4615-0799-4. doi: 10.1007/978-1-4615-0799-4_9. URL https://doi.org/10.1007/978-1-4615-0799-4_9.

[118] I. Blanes and J. Serra-Sagrista. Cost and scalability improvements to the Karhunen–Loêve Transform for remote-sensing image coding. *IEEE Transactions on Geoscience and Remote Sensing*, 48(7):2854–2863, July 2010. ISSN 0196-2892. doi: 10.1109/TGRS.2010.2042063.

[119] B. Penna, T. Tillo, E. Magli, and G. Olmo. Transform Coding Techniques for Lossy Hyperspectral Data Compression. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1408–1421, 2007.

[120] I. Blanes and J. Serra-Sagrista. Pairwise Orthogonal Transform for spectral image coding. *IEEE Transactions on Geoscience and Remote Sensing*, 49(3):961–972, March 2011. ISSN 0196-2892. doi: 10.1109/TGRS.2010.2071880.

[121] Q. Du and J. E. Fowler. Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis. *IEEE Geoscience and Remote Sensing Letters*, 4(2):201–205, 2007. doi: 10.1109/LGRS.2006.888109.

[122] C. Lee, S. Youn, T. Jeong, E. Lee, and J. Serra-Sagristà. Hybrid Compression of Hyperspectral Images Based on PCA With Pre-Encoding Discriminant Information. *IEEE Geoscience and Remote Sensing Letters*, 12(7):1491–1495, 2015. doi: 10.1109/LGRS.2015.2409897.

[123] P. Luigi Dragotti, G. Poggi, and A. R. P. Ragozini. Compression of multispectral images by three-dimensional SPIHT algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 38(1):416–428, 2000. doi: 10.1109/36.823937.

[124] X. Tang and W. A. Pearlman. Lossy-To-Lossless Block-Based Compression of Hyperspectral Volumetric Data. In *2006 International Conference on Image Processing*, pages 1133–1136, 2006. doi: 10.1109/ICIP.2006.312756.

[125] F. Khelifi, F. Kurugollu, and A. Bouridane. SPECK-Based Lossless Multispectral Image Coding. *IEEE Signal Processing Letters*, 15:69–72, 2008. doi: 10.1109/LSP.2007.911156.

[126] Consultative Committee for Space Data Systems. *Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression, Informational Report CCSDS 120.3-G-1.* CCSDS, March 2019. Green Book.

[127] Timothy S. Wilkinson and Val D. Vaughn. Application of video-based coding to hyperspectral imagery. In Sylvia S. Shen, editor, *Hyperspectral Remote Sensing and Applications*, volume 2821, pages 44 – 54. International Society for Optics and Photonics, SPIE, 1996. doi: 10.1117/12.257183. URL https://doi.org/10.1117/12.257183.

[128] L. Santos, S. Lopez, G. M. Callico, J. F. Lopez, and R. Sarmiento. Performance Evaluation of the H.264/AVC Video Coding Standard for Lossy Hyperspectral Image Compression. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):451–461, 2012. doi: 10.1109/JSTARS.2011.2173906.

[129] Miloš Radosavljević, Branko Brkljač, Predrag Lugonja, Vladimir Crnojević, Željen Trpovski, Zixiang Xiong, and Dejan Vukobratović. Lossy Compression of Multispectral Satellite Images with Application to Crop Thematic Mapping: A HEVC Comparative Study. *Remote Sensing*, 12(10):1590, May 2020. ISSN 2072-4292. doi: 10.3390/rs12101590. URL http://dx.doi.org/10.3390/rs12101590.

[130] International Telecommunication Union. *High Efficiency Video Coding, Recommendation ITU-T H.265*. ITU-T, November 2019. Series H: Audiovisual and Multimedia Systems. Infrastructure of audiovisual services, coding of moving video.

[131] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003. doi: 10.1109/TCSVT.2003.815165.

[132] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-complexity transform and quantization in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):598–603, 2003. doi: 10.1109/TCSVT.2003.814964.

[133] D. Marpe, H. Schwarz, and T. Wiegand. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, 2003. doi: 10.1109/TCSVT.2003.815173.

[134] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. doi: 10.1109/TCSVT.2012.2221191.

[135] International Organization for Standardization. *ISO/IEC 15444:32002*. ISO, September 2002. Information technology - JPEG 2000 image coding system, Part 3: Motion JPEG 2000.

[136] Consultative Committee for Space Data Systems. *Digital Motion Imagery, Recommended Standard CCSDS 766.1-B-2*. CCSDS, August 2016. Blue Book.

[137] G. Salazar and G. Steele. Commercial Off-The-Shelf (COTS) Graphics Processing Board (GPB) Radiation Test Evaluation Report. *National Aeronautics and Space Administration (NASA), Johnson Space Center*, pages 1–15, December 2013.

[138] European Cooperation for Space Standardization. *Space Engineering, Software (ECSS-E-ST-40C)*. ECSS, March 2009.

[139] ESA. Data compression tools. http://www.esa.int/Enabling_Support/Space_Engineering_Technology/Onboard_Data_Processing/Data_Compression_Tools. Accessed: 2020-12-14.

[140] Universitat Autonoma de Barcelona GICI Group. Emporda software. http://gici.uab.cat/GiciWebPage/downloads.php#emporda, 2011. Accessed: 2021-10-12.

[141] CNES. CCSDS 123.0-B-2 & 121.0-B-3. https://logiciels.cnes.fr/en/license/173/728, 2021. Accessed: 2021-10-26.

[142] J. Andersson, M. Hjorth, F. Johansson, and S. Habinc. LEON Processor Devices for Space Missions: first 20 years of LEON in Space. In *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, pages 136–141, 2017. doi: 10.1109/SMC-IT.2017.31.

[143] Cobham Gaisler. GR740 Quad-Core LEON4 SPARC V8 Processor. https://www.gaisler.com/index.php/products/components/gr740, . Accessed: 2020-12-17.

[144] Cobham Gaisler. GR712RC Dual-Core LEON3FT SPARC V8 Processor. https://www.gaisler.com/index.php/products/components/gr712rc, . Accessed: 2020-12-17.

[145] M. Olaru, A. Bulumac and V. Picos. Parallel lossless Compression for Multispectral and Hyperspectral Images on Multicore architectures. In *6th International Workshop on On-Board Payload Data Compression (OBPDC), Matera, Italy*, pages 1–5. European Space Agency, 2018.

[146] Stefano Di Mascio, Alessandra Menicucci, Gianluca Furano, Claudio Monteleone, and Marco Ottavi. The Case for RISC-V in Space. In Sergio Saponara and Alessandro De Gloria, editors, *Applications in Electronics Pervading Industry, Environment and Society*, pages 319–325, Cham, 2019. Springer International Publishing. ISBN 978-3-030-11973-7.

[147] W. Powell. High-Performance Spaceflight Computing (HPSC) Project Overview. In *Radiation Hardened Electronics Technology (RHET) Conference*, pages 1–29, 2018.

[148] NanoXplore. NG-LARGE NX1H140TSP Preliminary Datasheet. https://www.nanoxplore.com/uploads/NanoXplore_NG-LARGE_Datasheet_v1.0.pdf, 2018. Accessed: 2020-12-17.

[149] VORAGO Technologies. VA10800/VA10820 ARM Cortex-M0 based Processor, Programmers Guide. https://static1.squarespace.com/static/5d920c8760259e0ec548338d/t/5de5a1d5cc029b3bcd7110ea/1575330273161/VA10800_VA10820_PG_v19.pdf, 2018. Accessed: 2020-12-18.

[150] Texas Instruments Inc. *Rad-Tolerant Class-V Floating-Point Digital Signal Processor SMJ320C6701-SP*, Sep 2013. https://www.ti.com/lit/ds/symlink/smj320c6701-sp.pdf?ts=1608469709754.

[151] R. Ginosar, P. Aviely, T. Israeli, and H. Meirov. RC64: High performance rad-hard manycore. In *2016 IEEE Aerospace Conference*, pages 1–9, 2016. doi: 10.1109/AERO.2016.7500697.

[152] Daw-Tung Lin and Chung-Yu Yang. H.264/AVC Video Encoder Realization and Acceleration on TI DM642 DSP. In Toshikazu Wada, Fay Huang, and Stephen Lin, editors, *Advances in Image and Video Technology*, pages 910–920, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-540-92957-4.

[153] Jiming Fan, Jiankang Zhou, Xinhua Chen, and Weimin Shen. Hyperspectral image data compression based on DSP. In Toru Yoshizawa, Ping Wei, Jesse Zheng, and Tsutomu Shimura, editors, *Optoelectronic Imaging and Multimedia Technology*, volume 7850, pages 92 – 101. International Society for Optics and Photonics, SPIE, 2010. doi: 10.1117/12.870307. URL https://doi.org/10.1117/12.870307.

[154] L. Qianwen and H. Bingliang. The Hyper-Spectral Image Compression System Based on DSP. In *2008 International Workshop on Education Technology and Training*

and 2008 International Workshop on Geoscience and Remote Sensing, volume 2, pages 171–174, 2008. doi: 10.1109/ETTandGRS.2008.270.

[155] Nektarios Kranitis, Ioannis Sideris, Antonios Tsigkanos, Georgios Theodorou, Antonios Paschalis, and Raffaele Vitulli. Efficient field-programmable gate array implementation of CCSDS 121.0-B-2 lossless data compression algorithm for image compression. *Journal of Applied Remote Sensing*, 9(1):1 – 15, 2015. doi: 10.1117/1.JRS.9.097499. URL https://doi.org/10.1117/1.JRS.9.097499.

[156] Alberto G. Villafranca, Shan Mignot, Jordi Portell, and Enrique García-Berro. FAPEC in an FPGA: a simple low-power solution for data compression in space. In Bormin Huang, Antonio J. Plaza, and Carole Thiebaut, editors, *Satellite Data Compression, Communications, and Processing VII*, volume 8157, pages 137 – 145. International Society for Optics and Photonics, SPIE, 2011. doi: 10.1117/12.895138. URL https://doi.org/10.1117/12.895138.

[157] 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 Transactions on Emerging Topics in Computing*, pages 1–1, 2019. ISSN 2168-6750. doi: 10.1109/TETC.2018.2854412.

[158] 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 *6th International Workshop on On-Board Payload Data Compression (OBPDC)*, pages 1–8, September 2018.

[159] J. Fjeldtvedt, M. Orlandic, and T. A. Johansen. An efficient real-time FPGA implementation of the CCSDS-123 compression standard for hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(10):3841–3852, Oct 2018. ISSN 1939-1404. doi: 10.1109/JSTARS.2018.2869697.

[160] D. Bascones, C. Gonzalez, and D. Mozos. Parallel implementation of the CCSDS 1.2.3 standard for hyperspectral lossless compression. *Remote Sensing*, 9(10), 2017. ISSN 2072-4292. doi: 10.3390/rs9100973. URL https://www.mdpi.com/2072-4292/9/10/973.

[161] M. Orlandic, J. Fjeldtvedt, and T. A. Johansen. A parallel FPGA implementation of the CCSDS-123 compression algorithm. *Remote Sensing*, 11(6), 2019. ISSN 2072-4292. doi: 10.3390/rs11060673. URL https://www.mdpi.com/2072-4292/11/6/673.

[162] D. Keymeulen, D. Dolman, S. Shin, J. Riddley, M. Klimesh, A. Kiely, D. R. Thompson, M. Cheng, S. Dolinar, E. Liggett, P. Sullivan, M. Bernas, K. Roth, C. Holyoake, K. Crocker and A. Smith. High Performance Space Data Acquisition, Clouds Screening and Data Compression with modified COTS Embedded System-on-Chip Instrument Avionics for Space-based Next Generation Imaging Spectrometers (NGIS). In *6th International Workshop on On-Board Payload Data Compression (OBPDC), Matera, Italy*, pages 1–25. European Space Agency, 2018.

[163] Li Li, Gang Zhou, Bjorn Fiethe, Harald Michalik, and Bjorn Osterloh. Efficient implementation of the CCSDS 122.0-B-1 compression standard on a space-qualified field programmable gate array. *Journal of Applied Remote Sensing*, 7(1):1 – 13, 2013. doi: 10.1117/1.JRS.7.074595. URL https://doi.org/10.1117/1.JRS.7.074595.

[164] Luis Guanter, Hermann Kaufmann, Karl Segl, Saskia Foerster, Christian Rogass, Sabine Chabrillat, Theres Kuester, André Hollstein, Godela Rossner, Christian Chlebek, and et al. The EnMAP Spaceborne Imaging Spectroscopy Mission for Earth Observation. *Remote Sensing*, 7(7):8830–8857, Jul 2015. ISSN 2072-4292. doi: 10.3390/rs70708830. URL http://dx.doi.org/10.3390/rs70708830.

[165] Satish S. Bhairannawar, Sayantam Sarkar, and K. B. Raja. FPGA Implementation of Optimized Karhunen–Loeve Transform for Image Processing Applications. *Journal of Real-Time Image Processing*, 17:357–370, 2018. doi: 10.1007/s11554-018-0776-x. URL https://link.springer.com/article/10.1007/s11554-018-0776-x.

[166] Lucana Santos, Ian Blanes, Aday García, Joan Serra-Sagristà, José López, and Roberto Sarmiento. On the hardware implementation of the arithmetic elements of the pairwise orthogonal transform. *Journal of Applied Remote Sensing*, 9(1):1 – 12, 2015. doi: 10.1117/1.JRS.9.097496. URL https://doi.org/10.1117/1.JRS.9.097496.

[167] Daniel Báscones, Carlos González, and Daniel Mozos. An FPGA Accelerator for Real-Time Lossy Compression of Hyperspectral Images. *Remote Sensing*, 12 (16):2563, Aug 2020. ISSN 2072-4292. doi: 10.3390/rs12162563. URL http://dx.doi.org/10.3390/rs12162563.

[168] A. García, L. Santos, S. López, G. Marrero, J. F. López, and R. Sarmiento. High level modular implementation of a lossy hyperspectral image compression algorithm on a FPGA. In *2013 5th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4, 2013. doi: 10.1109/WHISPERS.2013.8080624.

[169] D. Báscones, C. González, and D. Mozos. An Extremely Pipelined FPGA Implementation of a Lossy Hyperspectral Image Compression Algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–13, 2020.

[170] Teng Wang, Chih-Kuang Chen, Qi-Hua Yang, and Xin-An Wang. FPGA Implementation and Verification System of H.264/AVC Encoder for HDTV Applications. In David Jin and Sally Lin, editors, *Advances in Computer Science and Information Engineering*, pages 345–352, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-30223-7.

[171] G. Pastuszak. Architecture Design of the H.264/AVC Encoder Based on Rate-Distortion Optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(11):1844–1856, 2015. doi: 10.1109/TCSVT.2015.2402911.

[172] C. C. Thiele, B. B. Vizzotto, A. L. M. Martins, V. S. da Rosa, and S. Bampi. A low-cost and high efficiency entropy encoder architecture for H.264/AVC. In *2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, pages 117–122, 2012. doi: 10.1109/VLSI-SoC.2012.7332087.

[173] G. Pastuszak. A High-Performance Architecture of the Double-Mode Binary Coder for H.264.AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(7):949–960, 2008. doi: 10.1109/TCSVT.2008.920743.

[174] G. Pastuszak and A. Abramowski. Algorithm and Architecture Design of the H.265/HEVC Intra Encoder. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):210–222, 2016. doi: 10.1109/TCSVT.2015.2428571.

[175] S. Atapattu, N. Liyanage, N. Menuka, I. Perera, and A. Pasqual. Real time all intra HEVC HD encoder on FPGA. In *2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 191–195, 2016. doi: 10.1109/ASAP.2016.7760792.

[176] Jean-Luc Poupat and Raffaele Vitulli. CWICOM: A Highly Integrated & Innovative CCSDS Image Compression ASIC. In L. Ouwehand, editor, *DASIA 2013 - DAta Systems In Aerospace*, volume 720 of *ESA Special Publication*, page 62, August 2013.

[177] eoPortal. GAIA (Global Astrometric Interferometer for Astrophysics) Mission. https://directory.eoportal.org/web/eoportal/satellite-missions/g/gaia. Accessed: 2019-07-22.

[178] R. Vitulli. PRDC: an ASIC device for lossless data compression implementing the Rice algorithm. In *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, volume 1, page 320, 2004. doi: 10.1109/IGARSS.2004.1369025.

[179] Stanislau Augé, José Enrique Sánchez, Aaron Kiely, Ian Blanes Garcia, and Joan Serra Sagrisà. Performance impact of parameter tuning on the CCSDS-123 lossless multi- and hyperspectral image compression standard. *Journal of applied remote sensing*, 7(1), Aug 2013. doi: 10.1117/1.JRS.7.074594. URL https://ddd.uab.cat/record/129769.

[180] M. Hernandez-Cabronero, A. B. Kiely, M. Klimesh, I. Blanes, J. Ligo, E. Magli, and J. Serra-Sagrista. The CCSDS 123.0-B-2 Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression Standard: A comprehensive review. *IEEE Geoscience and Remote Sensing Magazine*, pages 0–0, 2021. doi: 10.1109/MGRS.2020.3048443.

[181] Ian Blanes, Aaron Kiely, Miguel Hernández-Cabronero, and Joan Serra-Sagristà. Performance Impact of Parameter Tuning on the CCSDS-123.0-B-2 Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression Standard. *Remote Sensing*, 11(11):1390, Jun 2019. ISSN 2072-4292. doi: 10.3390/rs11111390. URL http://dx.doi.org/10.3390/rs11111390.

[182] Qizhi Fang, Yuxuan Liu, and Lili Zhang. Design and Implementation of a Lossless Compression System for Hyperspectral images. *IIETA Traitement du Signal*, 37(5): 745–752, 2020.

[183] IEEE Spectrum. European Space Agency Targets Orbital Debris, Solar Storms. https://spectrum.ieee.org/tech-talk/aerospace/satellites/european-space-agency-esa-mission-news-orbital-debris-solar-storms. Accessed: 2021-05-21.

[184] Max Planck Institute for Solar System Research (MPS). SUNRISE III: a balloon-borne Solar Observatory. https://www.mps.mpg.de/solar-physics/sunrise. Accessed: 2021-05-26.

[185] ESA. SpaceWire-b (SpW-b). https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Microelectronics/SpWb, 2009. Accessed: 2022-02-23.

[186] S.M. Parkes and P. Armbruster. Spacewire: a spacecraft onboard network for real-time communications. In *14th IEEE-NPSS Real Time Conference, 2005.*, pages 6–10, 2005. doi: 10.1109/RTC.2005.1547397.

[187] Stuart Mills, Pete Scott, and Steve Parkes. High speed test and development with the SpaceWire Brick Mk3. In *2014 International SpaceWire Conference (SpaceWire)*, pages 1–5, 2014. doi: 10.1109/SpaceWire.2014.6936273.

[188] 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 Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(2):757–770, Feb 2016. ISSN 1939-1404. doi: 10.1109/JSTARS.2015.2497163.

[189] 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 *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 278–284, July 2014. doi: 10.1109/AHS.2014.6880188.

[190] D. Bascones, C. Gonzalez, and D. Mozos. FPGA implementation of the CCSDS 1.2.3 standard for real-time hyperspectral lossless compression. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(4):1158–1165, April 2018. ISSN 1939-1404. doi: 10.1109/JSTARS.2017.2767680.

[191] 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 *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2019. doi: 10.1109/ISCAS.2019.8702428.

[192] Yubal Barrios, Alfonso Rodríguez, Antonio Sánchez, Arturo Pérez, Sebastián López, Andrés Otero, Eduardo de la Torre, and Roberto Sarmiento. Lossy Hyperspectral Image Compression on a Reconfigurable and Fault-Tolerant FPGA-Based Adaptive Computing Platform. *Electronics*, 9(10):1576, Sep 2020. ISSN 2079-9292. doi: 10.3390/electronics9101576. URL http://dx.doi.org/10.3390/electronics9101576.

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

[194] Y. Barrios, P. Rodríguez, A. Sánchez, M.I. González, L. Berrojo, and R. Sarmiento. Implementation of cloud detection and processing algorithms and CCSDS-compliant hyperspectral image compression for CHIME mission. In *7th International Workshop on On-Board Payload Data Compression (OBPDC)*, pages 1–8, September 2020.

[195] Xilinx Inc. KCU105 Board, February 2019. User Guide, v1.10.

[196] Imaging Development Systems. *uEye Cameras, User Manual*. IDS, 3.32 edition, Mar 2009. URL https://www.1stvision.com/cameras/IDS/dataman/uEye_User_Manual.pdf.

[197] Consultative Committee for Space Data Systems. *Lossless Multispectral and Hyperspectral Image Compression, Informational Report CCSDS 120.2-G-1*. CCSDS, December 2015. Green Book.

[198] International Telecommunication Union. *Parameter values for ultra-high definition television systems for production and international programme exchange*, volume 1. UIT-R, Recommendation ITU-R BT.2020-2 edition, 2015.

[199] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 549–565, Cham, 2016. Springer International Publishing.

[200] Zhou Wang and A.C. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9(3):81–84, 2002. doi: 10.1109/97.995823.