ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΠΤΥΣΗ ΤΟΥ ΑΡΓΟΡΙΘΜΟΥ CCSDS123 ΣΕ ΑΡΧΙΤΕΚΤΟΝΙΚΗ XPP-III

Διπλωματική Εργασία

Τηλέμαχος Τσιάπρας

Επιβλέπων: Γεώργιος Σταμούλης

Βόλος 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DEVELOPEMENT OF THE CCSDS 123 ALGORITHM ON XPP-III ARCHITECTURE

Diploma Thesis

Tilemachos Tsiapras

Supervisor: Georgios Stamoulis

Volos 2021

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Γεώργιος Σταμούλης** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανι- κών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Νέστωρ Ευμορφόπουλος** Αναπληρωτής Καθηγητής Τμήμα Ηλεκτρολόγων Μηχανι- κών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Αντώνιος Δαδαλιάρης** Επίκουρος Καθηγητής, Τμήμα Πληροφορικής και Τηλέπικοινωνιών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 05-03-2021

Supervising committee:

Supervisor	**Georgios Stamoulis** Professor, Department of electrical and computer engineering, University of Thessaly
Co-supervisor	**Nestoras Eumorfopoulos** Associate Professor, Department of electrical and computer engineering, University of Thessaly
Co-supervisor	**Antotions Dadaliaris** Assistant professor, Department of Computer Science and Telecommunication, University of Thessaly

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

Τηλέμαχος Τσιάπρας
05/03/2021

ΠΕΡΙΛΗΨΗ

Κατά τη διάρκεια των τελευταίων δεκαετιών πολλοί τομείς της άμυνας, έρευνας και υπηρεσιών εξαρτώνται από την τηλεπισκόπιση. Η Υπερφασματική και Πολυφασματική απεικόνιση αποτελεί σημαντικό εργαλείο για τη συλλογή πληροφοριών για την επιφάνεια της γης. Η αυξανόμενη χωρική και φασματική ανάλυση των αισθητήρων ωθεί προς την υιοθέτηση πιο αποδοτικών αλγορίθμων για επεξεργασία, συμπίεση και μετάδοση του μεγάλου όγκου δεδομένων που παράγετε. Σε απάντηση στις αυξανόμενες ανάγκες, η CCSDS δημοσίευσε μια σειρά από στάνταρντ για τη μετάδοση, προ-επεξεργασία και συμπίεσή δεδομένων.

Ο στόχος μας για αυτή τη διπλωματική εργασία είναι η υλοποίηση του στάνταρ συμπίεσης δίχως-απώλειες για Υπερφασματικά και Πολυφασματικά δεδομένα. Η κατεύθυνση που ακολουθήθηκε ήταν η υλοποίηση του στάνταρ στο καινούργιάο space-suitable RC σύστημα HPDH. Κατά τη διάρκεια της υλοποίησης προσπαθήσαμε να χρησιμοποιήσουμε τα ειδικά χαρακτηριστικά που προσέφερε η επιλεγμένη αρχιτεκτονική ώστε να μπορέσουμε να πλησιάσουμε το θεωρητικό μέγιστο όριο επίδοσης.

ABSTRACT

For the past decades many aspects of our defence, research and services have come to depend on remote sensing technologies. Hyperspectral and Multispectral imaging are an important asset for collecting earth surface data. The Increasing spatial and spectral resolution of the sensors though is constantly pushing for ever more efficient algorithms for processing, compressing, and transmitting the large amount of data generated. In response to the increasing requirements, CCSDS have released a series of standards concerning the aspects of transition, pre-processing, and compression.

Or goal with this thesis is the implementation of the lossless-compression standard for Hyperspectral/multispectral data released by the CCSDS. The approach we follow is the implementation of the standard on the new space-suitable RC device HPDP. During the implementation we try to utilize the features provided by the chosen architecture in order to come close to the theoretical maximum performance that we can achieve.

Table of Contents

List of figures

Abbreviations

This section defines the acronyms used in this document.

Acronym	Description
ALU	Arithmetic Logic Unit
ASIC	Application specific Intergraded Circuit
CCSDS	Consultative Committee for Space Data Systems
FPGA	Field programmable Array
CLB	Configurable Logic Block
XPP	eXtreme Processing Platform
HPDP	High Performance Data Processing
DMA	Direct Memory Access
FL	Fast Lossless
ESA	European Space Agency
JPEG-LS	Joint Photographic Experts Group – LosslesS
ALU	Arithmetic Logic Unit
PLD	Programmable Logic Device
PAE	Processing Array Element
(S)CM	(Supervising) Configuration Manager
HS	Hyper-Spectral
MS	Multispectral

Mathematical Notation

The largest integer n such $n \leq x : \ n = \lfloor x \rfloor$

The largest integer n such $n \geq x : \ n = \lceil x \rceil$

The modulus of an integer M with respect to a positive integer divisor n:

$$M \ mod \ n = M - n\lfloor M/n \rfloor$$

For any integer x and positive integer R, the function $mod_R^*[x]$ is defined as:

$$mod_R^*[x] = \left((x + 2^{R-1}) mod \ 2^R\right) - 2^{R-1}$$

The notation $clip(x, \{x_{min}, x_{max}\})$clip denotes the clipping of an integer number x to the range $[x_{min}, x_{max}]$:

$$clip(x, \{x_{min}, x_{max}\}) = \begin{cases} x_{min}, & x < x_{min} \\ x, & x_{min} \leq x \leq x_{max} \\ x_{max}, & x > x_{max} \end{cases}$$

The notation $sgn(x)$ is defined as:

$$sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

Finally, the notation $sgn^+(x)$ is defined as:

$$sgn^+(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

CHAPTER 1: Introduction

1.1 Problem Description

In today's world more and more aspects of our infrastructure, agriculture, defense, and environmental research begin to rely on earth observing technologies. The main source of data for this case is remote sensing devices i.e. orbiting or geostationary satellites. One of the ways that satellites provide information is by carrying specialized hyper-spectral imaging sensors that can provide us with much “richer” data about the surface of the earth, than a simple picture in the visible spectrum. This innovation though brings a challenge for the on-board computational units (OBCUs) and the downlinks to earth stations, as the huge amount of data is generated by HS sensors. The solution is the inclusion of an inline compression step in the on-board processing pipeline. Compression algorithms are though computationally intensive, and the computational units limited by power consumption restrictions do not offer efficient solutions.

1.2 Thesis contribution

Conventional satellite payloads using ASICs for greater performance/power consumption ratios have very little flexibility when it comes to adapting to changing standards and application evolution [1]. One possible solution comes in the form of reconfigurable technologies. The contribution of this thesis is the implementation of state-of-the-art compression standards of hyper spectral imagery used for scientific research on a Unique reconfigurable architecture integrated in a space suitable system.

1.3 Thesis structure

The main body of the thesis is composed of 4 chapters.

- **Chapter 2** includes an introduction to the XPP(RC) architecture and then dives deeper in the specifics of the architecture.
- **Chapter 3** includes an introduction in the technology behind Hyperspectral/Multispectral imaging and after that, it defines in detail the compression standard.

- **Chapter 4** describes the implementation of the standard on the HPDP architecture as well as difficulties encountered during the development phase.
- **Chapter 5** includes results for the tests performed, conclusion of the thesis and possible improvements that can be implemented in future work.

CHAPTER 2: Introduction to XPP(RC) architecture

2.1 Introduction to Reconfigurable Computing

During the mid 1980's a new technology called field programmable gate array (FPGA) was introduced. These devices when introduced were smaller and slower than the existing Mask programmable gate arrays (MPGAs) and larger and more expensive than PLDs. The advantage of the FPGAs over MPGAs was the configuration process. MPGAs were designed to handle larger logic circuits and consist of an array of prefabricated transistors that could be customized to implement a specific logic. The customization of the interconnection among the rows of transistors took place during fabrication making the setup cost for an MPGA far larger than the user programmable equivalent FPGA [2]. Many FPGAs were initially configured using static random-access memory (SRAM) cells in the array. This configuration medium was the key for many applications as it allows for the programming of an FPGA by a completely electrical process. That meant that the programming or configuration of the FPGA could change and configured to suit multiple applications [3]. So, a setup using an FPGA and a read-only memory that stores multiple configurations could function as a multimode hardware able to change depending on the current demands of a specific application. Another advantage is the ability for systems that include FPGAs to adapt to new data processing standards or communication protocols expanding the abilities of older already deployed systems. Apart from superior adaptability and application specific performance gains over general purpose computational units, one more application that showed great potential was logic emulation or chip verification. This process is becoming more and more complex and computationally intensive when software tools are used. The solution offered by FPGAs is the ability to directly map the desired circuit on a system of FPGAs. This way the tested circuit can run in real-time and minimize the time between test cycles [3].

2.1.1 FPGA Technology

This section briefly presents the technology of the SRAM based FPGAs which is the most common programming technology. Most of the descriptions of architectural abstractions

are applicable two all other Technologies [4]. In general, all FPGA architectures consist of configurable logic blocks, configurable I/O blocks and programmable interconnect. Additional circuitry is also present for controlling clock signals as well as special purpose blocks like ALUs, Floating point arithmetic blocks, memory and for certain applications, digital signal processing (DSP) blocks (DSPs usually come in the form of an embedded core). These blocks are organized in a matrix of configurable logic blocks (CLBs) interconnected by configurable interconnection circuitry. A generic illustration is given in figure (1).

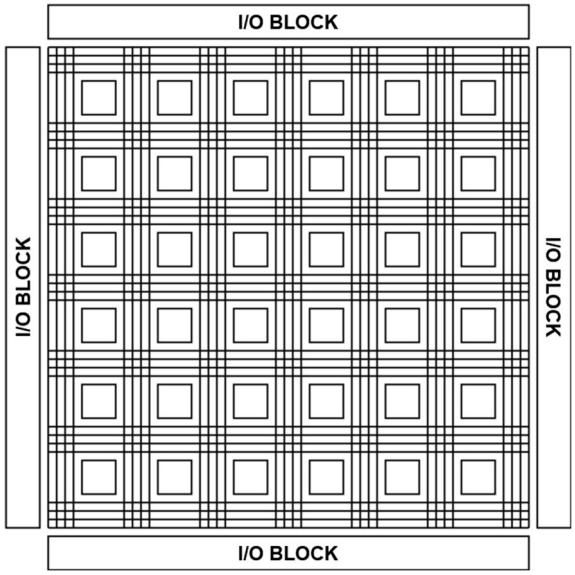

Figure 1: Generic FPGA architecture.

CLBs are the blocks that implement the logic of the FPGA, they are the basic FPGA cell. They implement macros and other design functions. Each CLB consists of several look up tables (LUTs) whose outputs are multiplexed (reprogrammable routing control).

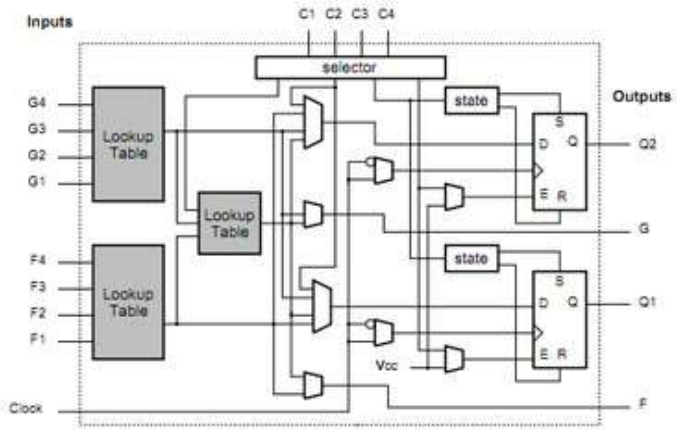

Figure 2: Generic CLB [5]

The CLBs are connected between them using the programmable interconnect. The interconnect resources can be used either as connection for far CLBs or can be used as internal bus. The connections are turned on or off using transistors. For CLBs that are physically close together, shorter lines exist. Those lines are a source of delay in a FPGA design as connection between CLBs may have to go through multiple interconnect transistors.

2.2 XPP architecture Overview

This next section offers a basic introduction to the eXtreme Processing Platform (XPP) and it is based on the introduction of the XPP architecture by PACT in 2003 [6].

The limitations of conventional processors and the rising importance of stream-based applications like digital signal processing and multimedia, increase the need for a faster and more efficient alternative. This alternative can come in the form of reconfigurable architectures as they combine the performance benefits of Application Specific Intergraded Circuits (ASICs) and the applications flexibility of processors [6]. The XPP architecture created by the French company PACT provides all the advantages of RC architecture providing additional functionality by including the ability of run-time reconfiguration and/or self-reconfiguration. This feature combined with the coarse grain, adaptive computing elements and the packet-oriented communication makes this architecture well suited for DSP applications, graphics and other stream based applications as different types of parallelism like, pipelining, instruction level, data flow, task level parallelism are supported [6].

The basic elements of the XPP architecture consists of reconfigurable ALUs which implement one of many possible basic machine operations like ADDITION, SHIFT and AND. The ALUs communicate via a packet-oriented communication network which feature automatic packet synchronization. This feature offers a level of abstraction giving greater freedom for programmers coming from higher level languages to get familiarized and implement applications on this architecture. In abstract, the description of the operations to be configured in each ALU as well as the interconnection scheme is described by a

configuration file. All this information in derived and can be translated by the data-flow graph of the implemented algorithm. This means that a great deal of focus is given to the dataflow generation of each algorithm. The figure (3) below shows the dataflow for the simple polynomial expressions $5x^2 + 6x + 1$ applied to a stream of data imported trough I/O elements.

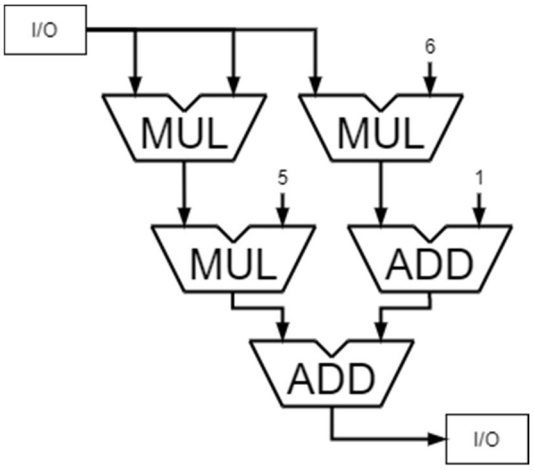

Figure 3: A configuration for polynomial expression calculation

As mentioned above one of the unique features of the XPP architecture is the ability for run-time reconfiguration. This allows for multiple configurations to be executed sequentially [6]. By breaking down the algorithm into smaller, inherently parallelizable segments, each processing a stream of data*, a greater data throughput can be achieved while spreading the overhead of multiple reconfigurations. According to reference [6]this programming paradigm can be described as a configuration flow, opposite to instruction flow embodied in the classical Von-Neuman architecture ** . The overhead added due to multiple reconfigurations I minimized because of caching next configurations making them available instantly. Figure (4) illustrates the difference between instruction and configuration flow.

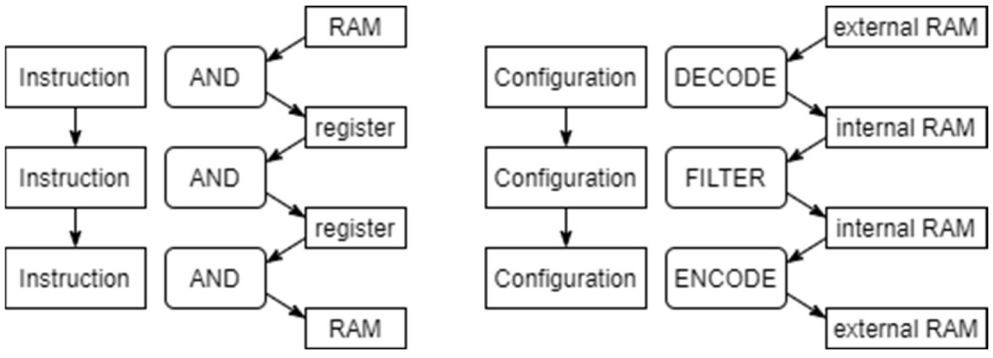

Figure 4: Difference between instruction and configuration flow.

**This concept is particularly important for the comprehension of the direction selected for the development of the CCSDS123 standard using the XPP architecture.*

*** The term von-Neuman architecture has evolved to mean any stored-program computer in which an instruction fetch, and a data operation cannot occur at the same time because they share a common bus. In this case we use the term "instruction flow" to describe the typical execution of a process using multiple instruction cycles.*

2.2.1 Architecture comparison

Field programmable gate arrays:

- Root of all reconfigurable computing devices
- Use fine grained cells and operate in date level
- They only allow complete configuration and cannot hold internal data during reconfiguration
- The resulting performance was only acceptable for algorithms well suited to the FPGA architecture

Partially reconfigurable FPGAs:

- Only required resources need to be configured.

Multi-context PLDs:

- They typically use fine grained architecture.
- They contain multiple planes of context memory.
- A configuration can be changes on the fly by switching planes

Microcontrollers and FPGAs combination:

- First step toward a complete programmable reconfigurable system.
- The microcontroller manages and executes the configuration and reconfigurations of the state of the FPGA.
- This approach cannot be a real solution for the reconfiguration and synchronization issues.

Reconfigurable processors:

- Most advanced class of reconfigurable architecture.
- Use coarse grained architecture and work at the top level.
- Size of configuration files is smaller, so the reconfigurations time is shorter.

XPP architecture:

- Belongs into the “Reconfigurable processors” class
- Main difference is the automatic packet-handling mechanism and its sophisticated hierarchical configuration protocols for full or partial runtime and self-reconfiguration.

2.2.2 Array structure

The reconfigurable array is based on a hierarchical array of coarse grain, adaptive computing elements called processing array elements (PAEs). The PAEs are typical grouped into blocks called processing array clusters (PACs). An XPP device is made up of multiple PACs. The configuration control is carried out by a hierarchical network of configuration managers (CMs) which are embedded in the array. So, each PAC is connected to a CM which is responsible for the loading of configuration data into the PAC. In multi-PAC devices additional CMs are added for the concurrent configuration data handling, each PAC is also connected to neighboring PACs. The root CM is called supervising configuration manager (SCM) and is directly connected to external memory containing configuration data. Figure (5) shown below depicts an abstract representation of a multi-PAC device.

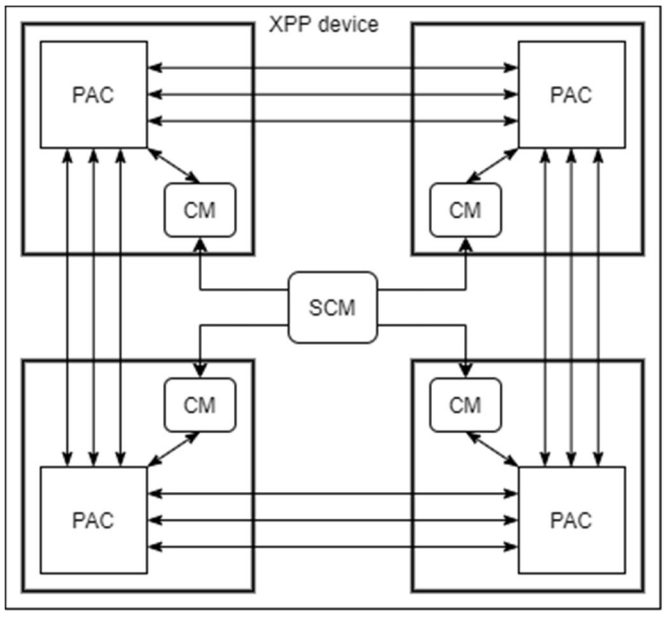

Figure 5: Basic structure of an XPP device

Every PAE in an XPP device is made of multiple configurable elements. The typical case of PAE contain two vertical routing objects, one back register(BREG) and one forward

register(FREG) as well as on ALU object for performing the actual calculations which include standard fixed point arithmetic, logical operations and some special opcodes like counters. Depending on the opcode implemented by each ALU and the result produced, event signals are generated, marking states like a classical microprocessor. The next case of PAE replaces the ALU processing element with a PAE memory object. This object can be used either as FIFO or as an internal RAM for look-up tables and temporary storage in an application.

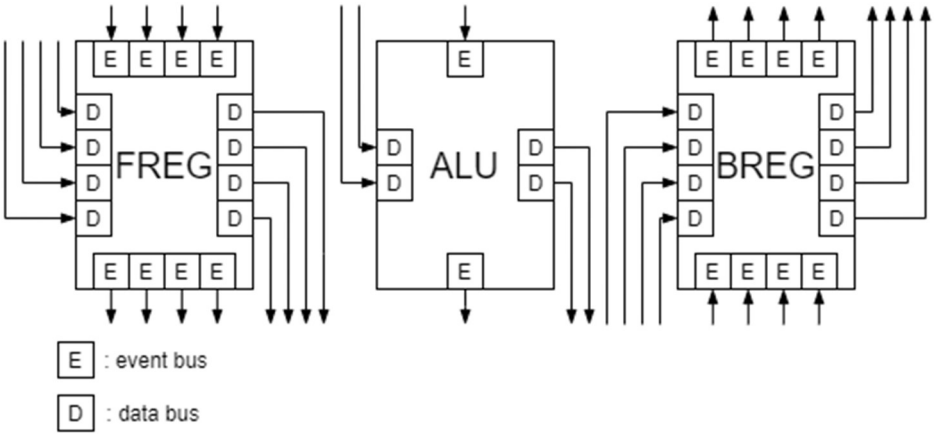

Figure 6: PAE structure

2.2.3 PAE communication

The structures described in the above section communicate via a packet communication network which carries two types packets, data packets and event packets. The packets carrying data have bit-width specific to the device while the event packets a just one bit wide. The event signals/packets add extra flexibility for data steam control. This gives the ability for merging/ demultiplexing and in general controlling the flow of data throughout the array, giving the choice of conditional computations depending on results of previous operations. This flexibility of the computations and control over the stream of data becomes possible due to the self-synchronization feature added to all PAE objects. This feature “holds” values until all required inputs for an operation are filled and only then an output value is produced and passed on. This minimizes the need for explicit scheduling of operations throughout the dataflow thus simplifying the development process for applications.

2.2.4 Configuration Method

A critical aspect of a reconfigurable systems affecting the performance, is the configurations and reconfigurations methods used. Especially devices that belong to the “reconfigurable processors” class cannot have slow conjurations methods as this would limit their use to very specific applications with the minimum computation to configuration ratio. The optimal case is devices which support concurrent computation and configuration. One way that the XPP optimizes the configuration process is by getting every PAE that has been configured to start computations while the rest of the array is still getting configured. Hardware protocols ensure packet integrity for the partially configured applications. More about the configuration methods can be found in [6].

2.2.5 Application mapping and programming tools

As with every reconfigurable architecture, the developer has to create a dataflow of the applications according to the regulations and limitations posed by the architecture, and then in turn, map the dataflow to the available fundamental elements. For this purpose, PACT has developed the Native Mapping Language (NML) which gives direct access to all hardware features to the programmer. A compiler for higher level of abstraction (C compiler) is also available, useful for simpler applications. A configuration consists of modules which contain PAE objects. The objects are explicitly allocated, optionally placed and their connections specified. Each configuration may contain more than one module, a sequence of initially configured modules and pre-fetched requests. The configurations handling is an explicit part of the application program.

2.2.6 Algorithm to diagram to mapping

The process of mapping an application to the device starts with the algorithm diagram. This diagram’ s nodes must be the available opcodes implemented to the arrays PAE objects. Once this step is complete then the diagram can be broken to segments that can be refined for parallel computations. Once the segments are connected, each gets “translated” to a configuration ready to be mapped to the device. A simple example is presented below.

For this example, I implement a simple digital filter, a FIR filter. The FIR filter can be described by the equation:

$$[y[n] = x[n] + x[n-1] + x[n-2]]$$

From this equation I can derive the dataflow and subsequently the configurations to be loaded to the device.

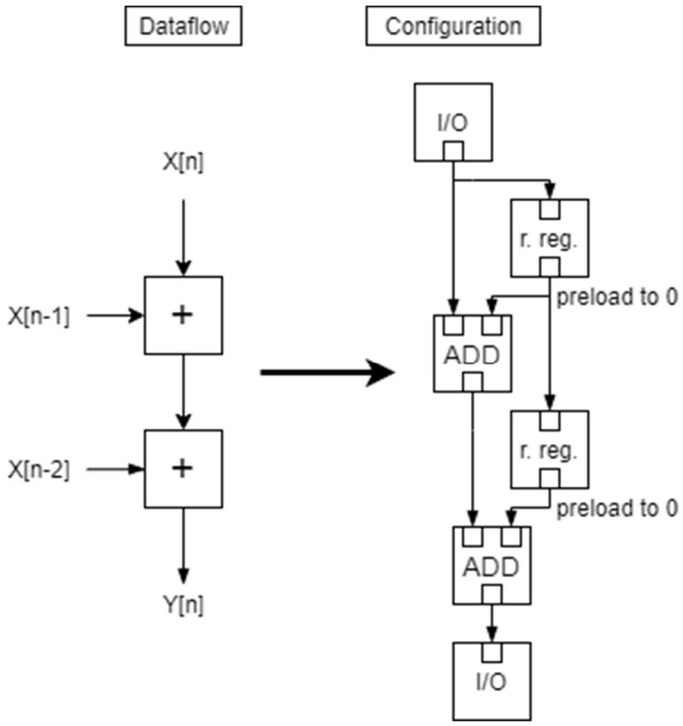

Figure 7: Dataflow to Configuration example.

2.2.7 Integration of the XPP core IP

The XPP architecture introduced in this chapter describes a RC core. The integration of the core is an important design aspect which allows us to take advantage of the features of the IP. HPDP is an array-based processor developed by Airbus Defense and space GmbH in Munich and ISD, SA in Greece. The development of HPDP has been initiated by the European Space Agency(ESA) and DLR to address the need for a flexible and re-programmable high performance data processor [1]. This architecture integrates an XPP RC processing core IP as well as memory interfaces and space suitable peripherals [7]. The XPP IP included in the design of the HPDP consists of 40 ALU elements and 16 RAM elements. For carrying out flow-control tasks, two Harvard type VLIW 16-bit processor cores (FNC-PAE) are included in the system. In general, the features of the HPDP include:

- Array Processor Based on the XPP III from Pact XPP Technologies (40 ALUs Processing elements 16-bit), 2 FNC-PAEs and 256 Kbit high speed on-chip RAM (with error

correction)

- 4x 1.6Gbit/s Streaming Ports
- 3 SpaceWire interfaces operating @ 100Mbps
- Fully reprogrammable platform

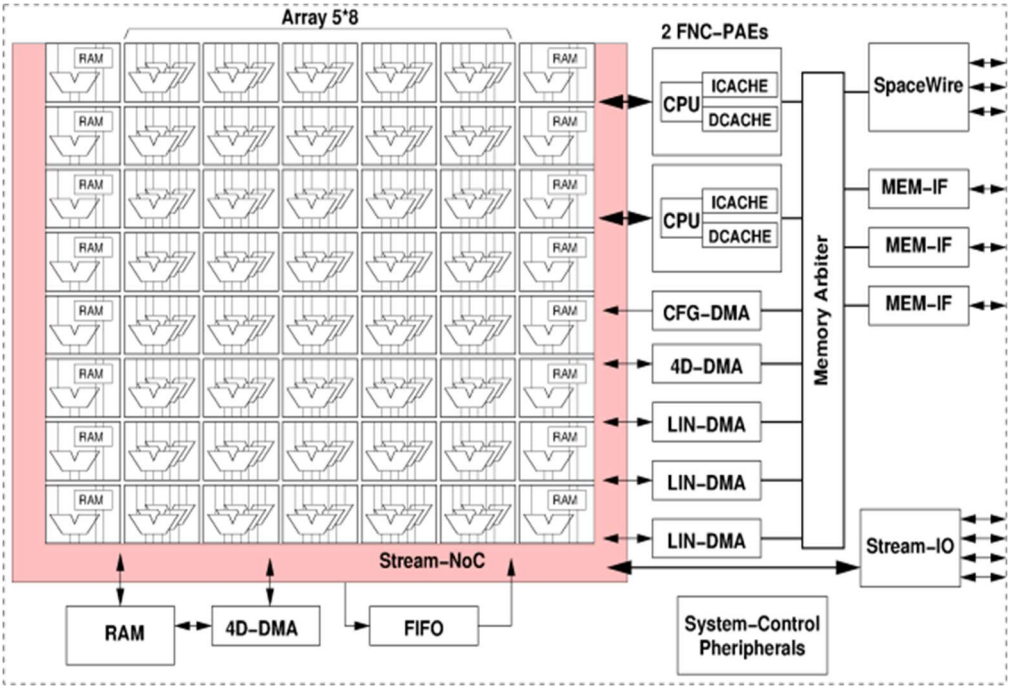

Figure 8: Overview of HPDP architecture [7]

2.2.7 XPP application mapping

For the implementation of the Compression algorithm the XPP SDK. The kit includes C compiler for the code running in the FNC-PAEs, a mapper that maps configuration files on the devices and most important of all, it includes a graphic simulator making possible to ensure validation of the implementation in a cycle by cycle level. Figure (9) offers a snapshot in the array simulator. In chapter 4, diagrams are used for describing the implementation. In these diagrams, as mentioned in this chapter, each distinct Block implements a calculation, a comparison, data routing and more. Each of these blocks are directly mapped to the objects shown above. The functionality of each block is included in the diagram when is non intuitive operations are performed i.e. counters, data routing, event generation.

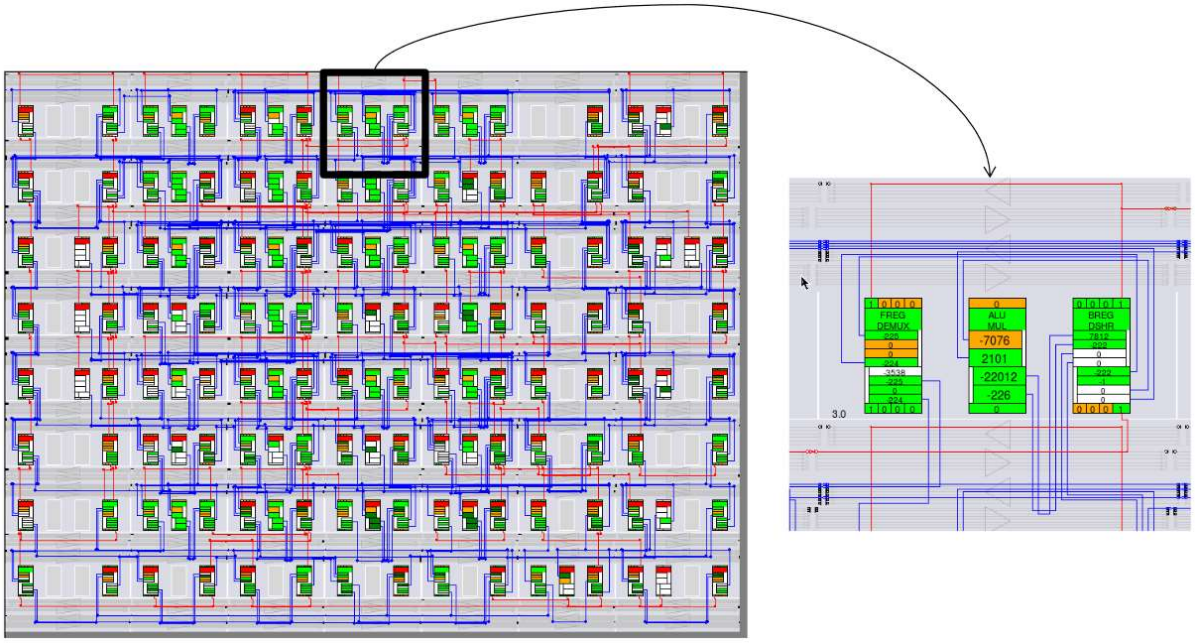

Figure 9: Snapshot from XPP array simulation [8].

CHAPTER 3: Introduction to the CCSDS123 standard

3.1 Introduction to Multispectral/Hyperspectral imaging

The term multispectral or spectral imaging describes the process of selecting information across a wide range of the electromagnetic spectrum. The early development of Spectral imaging begun nearly 50 years ago and until the end of the 1980s the use of this technology was mainly restricted to military use as well as use from astrophysics for enhancing remote sensing capabilities. The main difference with conventional imaging is the size of the electromagnetic spectrum covered by the sensing devices. While common imaging sensors can only "sense" information in the visible spectrum, specifically three narrow bands i.e. Red, Green and Blue, a multispectral sensor can perceive a much wider range of electromagnetic frequencies in the form of multiple bands [9] . Multispectral imaging measures discrete spectral bands and hyperspectral imaging measures "continues" spectral band. [10]

Below a series of images of the city of Volos are included, captured by the Sentinel-2 mission* at 03/01/2021. The first image is the RGB true colour image and the following three combinations of spectral bands information. Each combination of bands shows the reflectivity of the earth's surface and atmosphere at a specific spectrum range. This is turn can enable us to analyse changes on the environment or atmosphere. (the colormaps used in the three images do not convey information other than the magnitude of the index value)

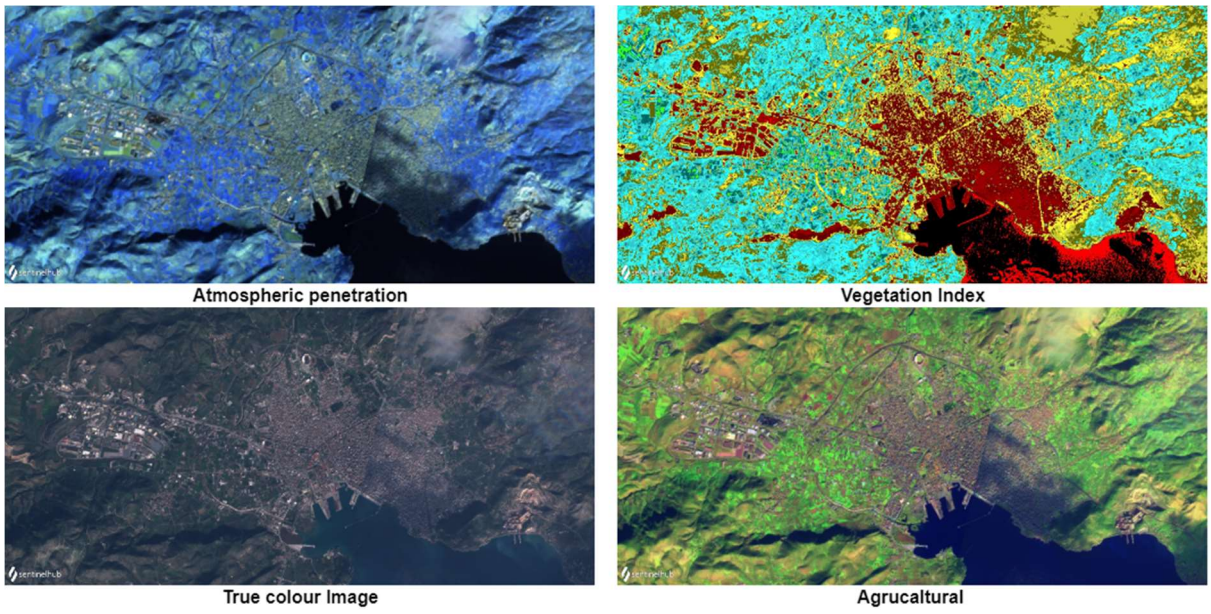

Figure 10: Same location in different spectral bands

* “The Copernicus Sentinel-2 mission comprises a constellation of two polar-orbiting satellites placed in the same sun-synchronous orbit, phased at 180° to each other. It aims at monitoring variability in land surface conditions, and its wide swath width (290 km) and high revisit time (10 days at the equator with one satellite, and 5 days with 2 satellites under cloud-free conditions which results in 2-3 days at mid-latitudes) will support monitoring of Earth's surface changes.” [11]

3.2 CCSDS

The Consultative Committee for Space Data Systems (CCSDS) is a “multi-national forum aimed at the development of communication and data systems standards for space flight”. It was found at 1982 and at this moment is comprised of “eleven member agencies, twenty-eight observer agencies, and over 140 industrial associates " [11] . The CCSDS has developed and formalize a variety of standards covering areas like data compression, data transmission, data collection and management. The CCSDS123 standard is one of the recommended standards developed of lossless data compression. The current members of the committee are the:

- United Kingdom Space Agency (UKSA)
- State Space Corporation (ROSCOSMOS)
- National Aeronautics and Space Administration (NASA)
- Japan Aerospace Exploration Agency (JAXA)
- Instituto Nacional de Pesquisas Espaciais (INPE)
- European Space Agency (ESA)
- Deutsche Zentrum für Luft- und Raumfahrt (DLR)
- China National Space Administration (CNSA)
- Centre National d'Etudes Spatiales (CNES)
- Canadian Space Agency (CSA)
- Agenzia Spaziale Italiana (ASI)

3.3 Need for CCSDS 123 standard

In this section we describe the CCSDS123, a standard designed for implementation onboard satellite computational systems. The importance of compression algorithms lies upon the restriction that these systems pose, as on-board memory is limited, and the downlink resources are spread thin across multiple on-board applications sending simultaneously data to station, usually during limited time windows. One important aspect of spectral imaging is that it generates a large amount of data, which may be difficult to handle [13]. The size of a spectral image "cube" can easily reach several tens of megabytes [2]. The standard proposed is a good compromise as it enables us to reduce the memory needs, minimizing the contact time need for communication with the station as well as reducing the data archival volume.

In addition, researchers using data generated by HS imaging devices need lossless compression methods for experiments. The standard proposed by the CCSDS achieve effective lossless compression using low-complexity methods. This entails the preservation of data accuracy while reducing data redundancy.

2.2 Definition of CCSDS123 standard

The building blocks of a typical image compression system consist of **decorrelation**, **quantization** and **entropy encoding**. The **decorrelation** stage may be a transformation like Discrete Wavelength, Discrete Cosine (used in JPEG) or Karhunen-Loeve transform [12]. The CCSDS-123 is using a predictive scheme based on Fast-lossless(FL) algorithm (NASA) [13] and for the entropy encoder stage, a sample adaptive encoder and a block-adaptive encoder (CCSDS 121.0-B) [14]. Sections 2.2.1-2.2.5 present an overview of the CCSDS-123 and do not attempt to explain the theory underlying the compression algorithm.

2.2.1 General Overview

The CCSDS123 standard defines a payload data compressor. The input of the compressor consists of a three-dimensional "cube" of integer data points called samples. The output of the compressor consists of a stream of bits containing the compressed samples as well as header containing information about the parameter used during compression, needed for

the retrieval of the original data. The compressor is tuneable, meaning that the user can vary the parameters to meet fidelity constrains [12] or to adjust to device limitations. The variation of these parameters will change the length of the final compressed bitstream. For our case, the limitation set by the device used (XPP), restricted the number of parameters we could tune and still ensure an overflow free compression cycle.

The compressor defined in the standard uses a three step process to compress the input image. These steps include:

- an adaptive linear predictor
- a quantizer and
- an entropy encoder.

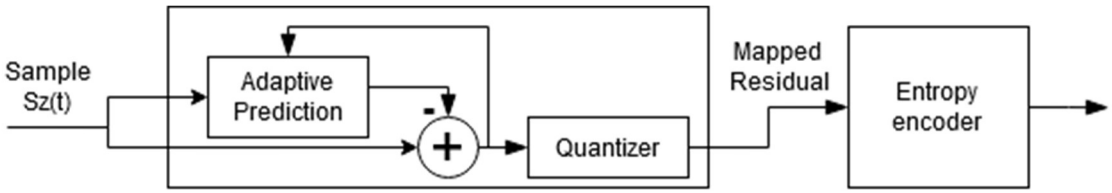

Figure 11: Schematic of the compressor

2.2.2 Parameters

An important aspect of the CCSDS123 standard is the tuneable parameters. These parameters allow us to control the performance and the compressor behaviour. It is important for any hardware implementations, to know which parameters have a greater impact on the performance and the compress ratio so we can adjust the design processes accordingly. Using data from [15] default parameters to be hard-coded into the implementation have been chosen. The table below includes definition and values chosen for every tuneable parameter.

Parameter	Value Range	Default for implementation	Conclusion
Number of bands for prediction	[0,15]	3	For P>3 no major gains found
Prediction mode	full, reduced	full	-
Local Sum Mode	Neighbour, column	column	-

Weight resolution(Ω)	[4,19]	4	Larger value yields more compression but the 16 bit architecture don't allow for larger values
Weight update scaling exponent Vmin and Vmax	[-6, - 9]	3 (both)	Do not have a significant impact
Initial count exponent (γ0)	[1,8]	1	Sets initial counter value. No major impact
Accumulator initialization constant (K)	[4-9]	4	Sets initial accumulator value. No major impact
Rescaling counter size (γ*)	[4-9]	XXX	-
Unary length limit (Umax)	[8-32]	8	Input samples are 8-bit wide, so no larger value is needed

2.2.3 Input specifications

The input to the processor as mentioned, is an HSI cube which extends to three dimensions with the size described by Nx, Ny, Nz where the Nz is the number of bands retrieved for each sample (as shown int the figure (12)). The indexing used to describe the pixels spatial information is the following, for the sample S(x,y,z) the z indicates the spectral band and the (x,y) the spatial coordinates, the pair x, y is also combined to one index t where $t = y * N_x + x$.

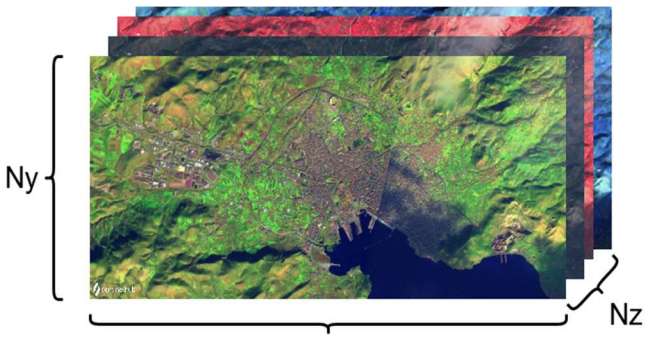

Figure 12: Abstract representation of MS image

The input data can also vary in the order that they are stored in memory, with the two most common ordering schemes being the BSQ and the BIP. BSQ stands for Band SeQuential and this method stores all the data for a band in continuous memory spaces meaning that in order to go from $S(x_i, y_i, z_i)$ to $S(x_i, y_i, z_i + 1)$, then we have to skip $Nx * Ny$ samples.

Files using the BIP (Band Interleaved Pixel) method store the all the band samples for a particular spatial point in continuous positions.

Figure 13: Input sample ordering

2.2.4 Predictor

First stage of the compressor is the sample prediction. The predictor is similar in form to previous algorithms like Lossless JPEG. In general, this prediction model generates a prediction for the value of one pixel/sample based on a set number of neighbouring samples. In this case, prediction a sample $S_{zyx}(t)$depends on the values of nearby samples in the same spactral band and P preceding spectral bands (P is user specified parameter). The first step is the calculation of the local sum $\sigma_{zyx}(t)$ which is a weighed sum of previous sample values in the band z . The three possible methods for calculating local sum are the Wide neighbour-oriented, Narrow neighbour-oriented and column oriented. Each method adds a different level of complexity and requirements during the implementation phase.

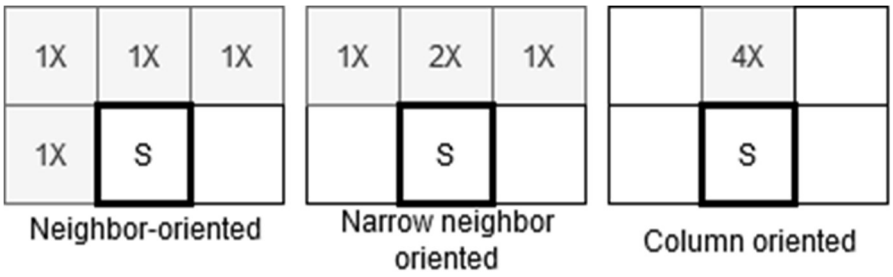

Figure 14: The 3 local sum modes.

When using neighbour-oriented local sum:

$$\sigma_{z,y,x}=\begin{cases} s''_{z,y,x-1}+s''_{z,y-1,x-1}+s''_{z,y-1,x}+s''_{z,y-1,x+1}, & y>0,\ 0<x<N_{\mathrm{X}}-1 \\ 4s''_{z,y,x-1}, & y=0,\ x>0 \\ 2\left(s''_{z,y-1,x}+s''_{z,y-1,x+1}\right), & y>0,\ x=0 \\ s''_{z,y,x-1}+s''_{z,y-1,x-1}+2s''_{z,y-1,x}, & y>0,\ x=N_{\mathrm{X}}-1 \end{cases}$$

Equation 1:Neighbour-oriented local sub

When using Narrow neighbour-oriented local sum:

$$\sigma_{z,y,x}=\begin{cases} s''_{z,y-1,x-1}+2s''_{z,y-1,x}+s''_{z,y-1,x+1}, & y>0,\ 0<x<N_{\mathrm{X}}-1 \\ 4s''_{z-1,y,x-1}, & y=0,x>0,\ z>0 \\ 2\left(s''_{z,y-1,x}+s''_{z,y-1,x+1}\right), & y>0,\ x=0 \\ 2\left(s''_{z,y-1,x-1}+s''_{z,y-1,x}\right), & y>0,\ x=N_{\mathrm{X}}-1 \\ 4s_{\mathrm{mid}}, & y=0,x>0,z=0 \end{cases}$$

Equation 2: Narrow neighbour local sum

When using Column oriented local sum:

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

Equation 3: Column oriented local sum

Using the calculated local sum, I can derive the local and directional differences. I take the directional differences from the band of the current sample and the central differences form the previous P bands and I combine them to create the difference vector. Directional difference is the difference of the local sum to the neighbouring samples of the current sample. The labels N, NW, W are used to define the positions of neighbouring samples. So, for a sample $S_{zyx}(t)$the central and directional differences are the following.

Directional differences:

$$d^{\mathrm{N}}_{z,y,x} = \begin{cases} 4s''_{z,y-1,x} - \sigma_{z,y,x}, & y > 0 \\ 0, & y = 0 \end{cases},$$

$$d^{\mathrm{W}}_{z,y,x} = \begin{cases} 4s''_{z,y,x-1} - \sigma_{z,y,x}, & x > 0,\ y > 0 \\ 4s''_{z,y-1,x} - \sigma_{z,y,x}, & x = 0,\ y > 0 \\ 0, & y = 0 \end{cases}, \text{ and}$$

$$d^{\mathrm{NW}}_{z,y,x} = \begin{cases} 4s''_{z,y-1,x-1} - \sigma_{z,y,x}, & x > 0,\ y > 0 \\ 4s''_{z,y-1,x} - \sigma_{z,y,x}, & x = 0,\ y > 0 \\ 0, & y = 0 \end{cases}.$$

Equation 4: Directional differences

Central difference:

$$d_{z,y,x} = 4\, s''_{z,y,x} - \sigma_{z,y,x}$$

Equation 5: Central difference

The local difference vector $U_z(t)$ is used to store the computed differences. Under reduced mode the $U_z(t)$ contains only the P central differences computed by the P previous bands. Under full mode the local differences from the current band are included. For $t > 0$,we can now calculate the central local difference $\widehat{d_z}(t)$which is equal to the inner product of the weight vector $W_z(t)$with the $U_z(t)$. The scaled predicted value $\breve{S}_z(t)$can now be calculated given that:

$$\breve{s}_z(t) = \mathrm{clip}\left(\mathrm{mod}^*_R\left[\hat{d}_z(t) + 2^{\Omega}\left(\sigma_z(t) - 4s_{\mathrm{mid}}\right)\right] + 2^{\Omega+2}s_{\mathrm{mid}} + 2^{\Omega+1}, \left\{2^{\Omega+2}s_{\min}, 2^{\Omega+2}s_{\max} + 2^{\Omega+1}\right\}\right)$$

Equation 6: Scaled prediction

The double-resolution resolution predicted sample value is:

$$\tilde{s}_z(t) = \begin{cases} \left\lfloor \dfrac{\breve{s}_z(t)}{2^{\Omega+1}} \right\rfloor, & t > 0 \\ 2s_{z-1}(t), & t = 0, P > 0, z > 0 \\ 2s_{\mathrm{mid}}, & t = 0 \text{ and } (P = 0 \text{ or } z = 0) \end{cases}.$$

Equation 7: Double resolution prediction

and finally, the predicted sample values $\widehat{S_z}(t)$is equal to $\frac{\widetilde{S_z}(t)}{2}$.

2.2.5 Weight Initialization

For the initialization of the weight vector $W_z(t)$ we can use either the default weight initialization or the Custom Weight Initialization. When the default weight initialization is used, for each spectral band Z, initial vector components values must be assigned:
For weights corresponding to central differences:

$$\omega_z^{(1)}(1) = \frac{7}{8}2^{\Omega}, \quad \omega_z^{(i)}(1) = \left\lfloor \frac{1}{8}\omega_z^{(i-1)}(1) \right\rfloor, i = 2,3,\ldots,P_z^*$$

Equation 8: Central default Weights

For weights corresponding to directional differences:

$$\omega_z^{\mathrm{N}}(1) = \omega_z^{\mathrm{W}}(1) = \omega_z^{\mathrm{NW}}(1) = 0.$$

Equation 9: Directional default weights

2.2.6 Weight Update

Every prediction cycle for each sample includes the dynamic update of the weight vector using the double-resolution prediction error of the previous prediction $e_z(t)$where $e_z(t) = 2\acute{S}_z(t) - \widetilde{S_z}(t)$.
The updated value of each weight component is corrected by:

$$\Delta W_z(t) = \left\lfloor \frac{1}{2}\left(\mathrm{sgn}^{+}[e_z(t)] \cdot 2^{-\rho(t)} \cdot U_z(t) + 1\right) \right\rfloor$$

Equation 10: Weight update

Where $\rho(t)$ is the update scaling exponent and controls convergence speed and is given by:

$$\rho(t) = \mathrm{clip}\left(\nu_{min} + \left\lfloor \frac{t - N_x}{t_{inc}} \right\rfloor, \{\nu_{min}, \nu_{max}\}\right) + D + \Omega$$

Equation 11: Scaling exponent

2.2.7 Residual Mapping

The prediction residual $\Delta_z(t)$ is the difference between the predicted and the actual sample values.

$$\Delta_z(t) = s_z(t) - \hat{s}_z(t) .$$

The method for fidelity control will not be presented as we are only interested in the lossless aspect of the compressor, so the next step is the mapping of the residuals to a D-bit unsigned integer producing a mapped residual $\delta_z(t)$.

$$\delta_z(t) = \begin{cases} |\Delta_z(t)| + \theta_z(t), & |\Delta_z(t)| > \theta_z(t), \\ 2|\Delta_z(t)|, & 0 \le (-1)^{\tilde{s}_z(t)}\Delta_z(t) \le \theta_z(t), \\ 2|\Delta_z(t)| - 1, & \text{otherwise}, \end{cases}$$

Equation 12: Mapped Residual

where

$$\theta_z(t) = \begin{cases} \min\{\hat{s}_z(0) - s_{\min}, s_{\max} - \hat{s}_z(0)\} & t = 0 \\ \min\left\{ \left\lfloor \frac{\hat{s}_z(t) - s_{\min} + m_z(t)}{2m_z(t)+1} \right\rfloor, \left\lfloor \frac{s_{\max} - \hat{s}_z(t) + m_z(t)}{2m_z(t)+1} \right\rfloor \right\}, & t > 0 \end{cases}$$

2.2.8 Entropy encoder

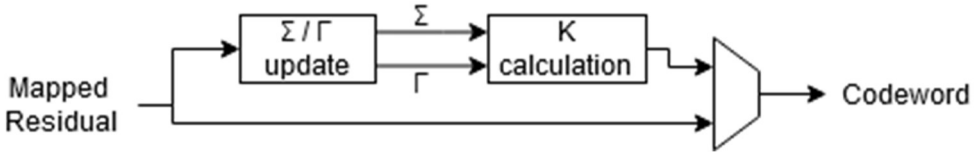

Figure 15: Schematic of Entropy encoder

After prediction, the mapped residuals defined above are then passed to the entropy encoder. In the CCSDS123 standard two methods are defined. First method is the block-adaptive entropy encoder. This method breaks down the $N_z * N_y * N_x$ residuals into blocks of a set size, and after calculating the encoded size of the block for several encoding methods, it uses the most efficient. It also includes special encoding for all-zero blocks. These characteristics make the block-adaptive the superior method for many applications,

especially applications with low entropy inputs, but the implementation in this thesis make use of the second encoding method as the block-adaptive method is not suited for devices with restricted memory and computing elements. The extra computation time needed for the multiple encoding scheme efficiency tests makes this method unsuitable for high throughput applications. The second entropy encoding method which we focus on is the adaptive entropy encoder based on RICE code (figure (15)). Rice coding is a subset of Golomb codes. The difference between the two is that while the Golomb code has a parameter that can be any possible positive integer, the Rice code is using parameters only divisible by two. This makes this method very "convenient" for use with binary arithmetic.

2.2.8.1 Rice codes overview

This section given an overview of the rice codes in the fashion used in the adaptive entropy encoder. Rice coding by limiting the tuneable parameter k to multiplicands of 2, offers the advantage of substituting the divisions needed with shift operations.
Given an input parameter N and the tuneable parameter M (multiple of 2):

$$q = floor(N \div M) \;\; r = NmodM$$

The final codeword has two parts the q encoded in unary code (q 0s followed by 1) and then the r using an K bit representation. Bellow an example is illustrated.

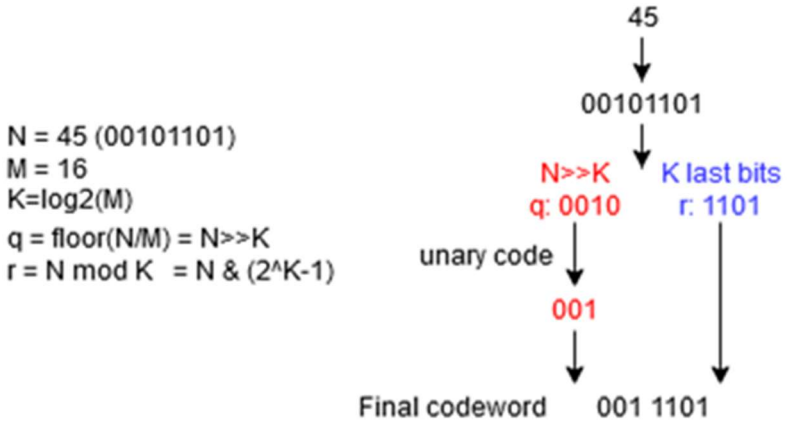

Figure 16: RICE coding example

The coding performer to the samples is length-limited, the length limitation is introduced by the parameter U_{max}. The length of the unary part of the codeword defined above as q is capped to U_{max}. When $q \geq U_{max}$ the codeword will consist of U_{max} zeros followed by the original D-bit representation of the mapped residual.

2.2.8.2 Sample-adaptive coding procedure

The Sample adaptive part of the encoder defined in the standard is referred to the dynamic adjustment of the tuneable parameter K (defined above). The choice of the parameter K is carried out by the first stage of the entropy encoder which takes the average sample value of the residuals in each band. The average is calculated by the accumulation of sample values $\Sigma_z(t)$ and dividing the result by the number of the processed samples $\Gamma(t)$.

The counter $\Gamma(t)$ in incremented for every sample by one for $\Gamma(t) < 2^{\gamma\chi} - 1$ where the rescaling counter size γ^{χ}determines the maximum value of the counter. The accumulator then is defined as:

$$\Sigma_z(t) = \begin{cases} \Sigma_z(t-1) + \delta_{\tilde{F}}(t-1), & \Gamma(t-1) < 2^{\gamma^*} - 1 \\ \left\lfloor \dfrac{\Sigma_z(t-1) + \delta_z(t-1) + 1}{2} \right\rfloor, & \Gamma(t-1) = 2^{\gamma^*} - 1 \end{cases}$$

Equation 13: Sample adaptive accumulator

and the counter is defined as:

$$\Gamma(t) = \begin{cases} \Gamma(t-1) + 1, & \Gamma(t-1) < 2^{\gamma^*} - 1 \\ \left\lfloor \dfrac{\Gamma(t-1) + 1}{2} \right\rfloor, & \Gamma(t-1) = 2^{\gamma^*} - 1 \end{cases}$$

Equation 14: Sample adaptive counter

The initial value for the counter and accumulator is given as:

$$\Gamma(1) = 2^{\gamma_0},$$

$$\Sigma_z(1) = \left\lfloor \frac{1}{2^7} \left(3 \cdot 2^{K+6} - 49\right) \Gamma(1) \right\rfloor$$

Equation 15: Counter and Accumulator initialisation

After defining the $\Sigma_z(t)$and $\Gamma(t)$metrics the parameter $k_z(t)$is defined as:

$$k_z(t) = 0 \text{ if } 2\Gamma(t) > \Sigma_z(t) + \left\lfloor \frac{49}{2^7} \Gamma(t) \right\rfloor$$

Equation 16: K parameter calculation

otherwise $k_z(t)$is the largest positive integer $k_z(t) \leq D - 2$.such that:

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

Equation 17: K parameter calculation

CHAPTER 4: Proposed Implementation

4.1 Executive Summary

During the development of the proposed implementation, many versions were tested, each offering advantages in memory requirements, array coverage and data throughput.
The one presented in this chapter achieves a good compromise between memory requirements and coverage while offering a competitive throughput for low energy applications.

The first configuration includes the prediction stage and receives samples in BIP order and outputs mapped residual values. The BIP order of the input data stream is an important aspect of the implementation as the dependency of the prediction computation on P previous bands (central differences) would create memory deficit in case BSQ was to be used. For example, for Nx=1000 Ny=1000 and P=2, 2M samples should be stored before the calculation of the first prediction. This I not viable as the HDPD provide 256Kbit on-chip RAM [7] The residuals produced by the first configuration are stored to external memory via DMA. After the full stream have passed through the predictor and all the residual values are stored, the configuration is removed from the XPP array and the loading of the second configuration begins. For the majority of reconfigurable architectures this step would add a major overhead. On the contrary, reconfiguration overhead for the XPP architecture, due to the optimizations mentions in chapter 2, is for most cases less than 0.5 % of the total computation time giving us the flexibility of multiple configurations for different processes in the same dataset in the same device. The second configuration receives mapped residuals from external memory via DMA and caries out the adaptive entropy encoding stage.

As mentioned in chapter 2 the HPDP architecture provides the XPP III array processor as well as 2 Harvard VLIW 16-bit processor cores (FNC-PAEs). Both implementations described in this chapter do not utilize the FNC-PAEs for heavy computations, their main purpose is the DMA initialization and configuration/reconfiguration of the XPP III array.

Bellow I include a brief description for the high-level modules that comprise the final implementation (Figure ()).

Configuration 1 – Prediction:

- **ISDM** – Input Store Delay Module

This module is in charge of storing samples in FIFOs so we can compute expressions that require a neighboring sample from the current one. The use of FIFOs allows us to decrease the memory accesses.

- **EGM** – Event Generation Module

The event generation module keeps track of the position of each sample and generates the appropriate events for all conditional multi-branch equations (Local Sum).

- **LSM** - Local Sum Module

Implements the column oriented Local Sum for simplicity Eq. (3).

- **DVM** - Difference Vector Module

This module constructs the difference vector using previous samples and current Local Sum. After the vector is constructed then the dot product between the difference and weight vectors.

- **WUM** - Weight Update Module

This Module receives the Difference vector and the error Sign and apply correction to the weight vector. This Module creates a feedback loop which present the main source of delay factor of this configuration.

- **RMM** – Residual Mapping Module

The final part of the configuration is the residual mapping which receives the scaled prediction error and outputs a mapped residual that gets stored to external memory.

Configuration 2 – Entropy encoder:

- **KM** - K module

This Module uses the Counter and Accumulator to calculate the K parameter for the entropy encoder. After the calculation of the parameter K the Counter and Accumulator get updated.

- **RCM** – RICE Code Module

This module receives the mapped residuals the a parameter K and calculates the RICE code for the specific inputs, taking into account limitations set by the standard.

- **BPM** – Byte Packing Module

The last Module of the configuration is the most complex part of the implementation due to

the difficult task of packing codewords of non-static length in a streaming fashion using a coarse grain architecture. The obvious obstacle in the process is the data dependency that a dynamic code length creates. The main delay in the Packing module is the update of the available bits is every new byte that gets packed. For the proposed implementation the packets created are 16-bit wide for efficiency. The implementation calculate the updated value using (S(t) : Size, A(t): Available bits) :

$$A(t+1) = \begin{cases} |S(t) - A(t)| & , S(t) < A(t) \\ 16 - |S(t) - A(t)| & , S(t) \geq A(t) \end{cases}$$

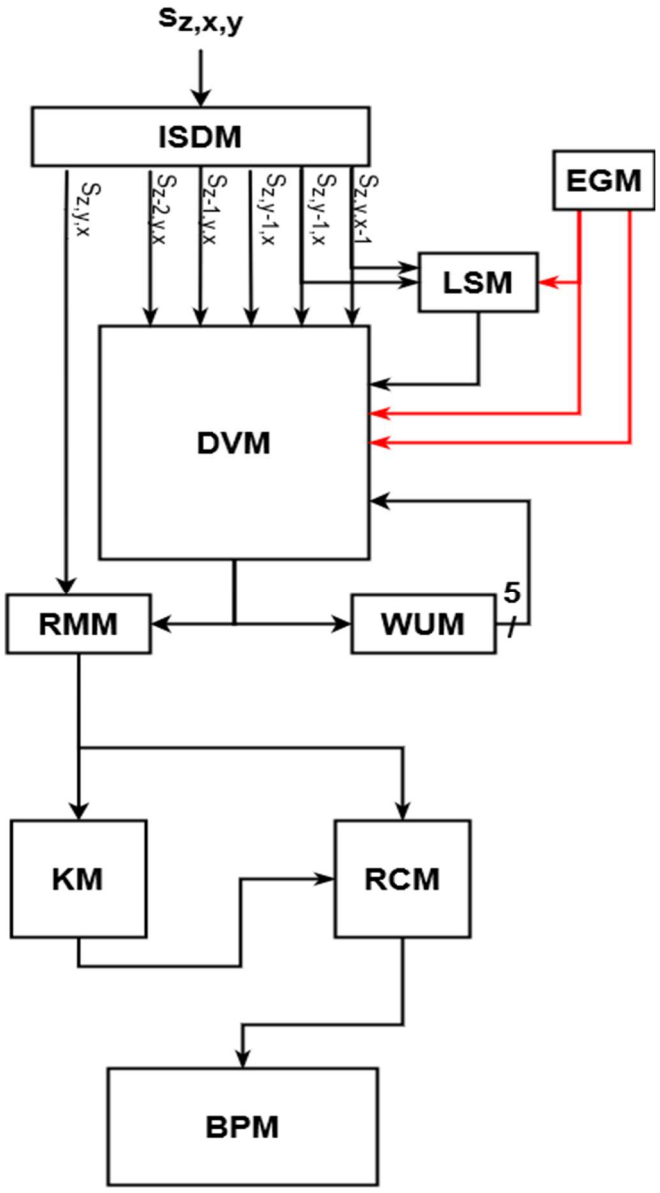

Figure 17: High level Implementation schematic

CHAPTER 5: Results, Conclusion, and future work

5.1 Results

For the test results, the implemented standard was tested against a compressor written in c using the same parameters as the hardwired parameters chosen for the proposed implementation. The compressor was run on Linux VM (intel i7-8550U @ 1.80GHz, 16GB memory). All input files had 8-bit dynamic range. The table below includes the performances results for test images of different dimensions. Compress ratio is not included as the test images were constructed by us. For tests carried out with real datasets, the compress rate is in the range [3.9 – 5.3 bits/sample]. Compress rate is mainly depended on the entropy of the input data and on the fact that we use 8-bit samples and sample-adaptive entropy encoding.

Test	Nx	Ny	Nz	PC runs (time)	HPDH runs (time)	PC runs (bits/s)	HPDP runs (bits/s)
0	100	1000	3	122.60 ms	19.20 ms	19.57 Mb/s	124.83 Mb/s
1	100	1000	9	367.70 ms	19.22 ms	19.58 Mb/s	374.49 Mb/s
2	100	1000	18	809.57 ms	22.42ms	17.78 Mb/s	642.11 Mb/s
3	100	1000	24	1.025 s	24.82ms	18.73 Mb/s	773.38 Mb/s
4	100	1000	36	1.572 s	30.02ms	18.32 Mb/s	959.16 Mb/s
5	100	1000	45	1.831 s	37.22 ms	19.65 Mb/s	967.06 Mb/s
6	100	1000	72	2.831 s	58.82 ms	20.34 Mb/s	979.15 Mb/s
7	512	2048	45	18.46 s	386.30 ms	20.44 Mb/s	977.18 Mb/s

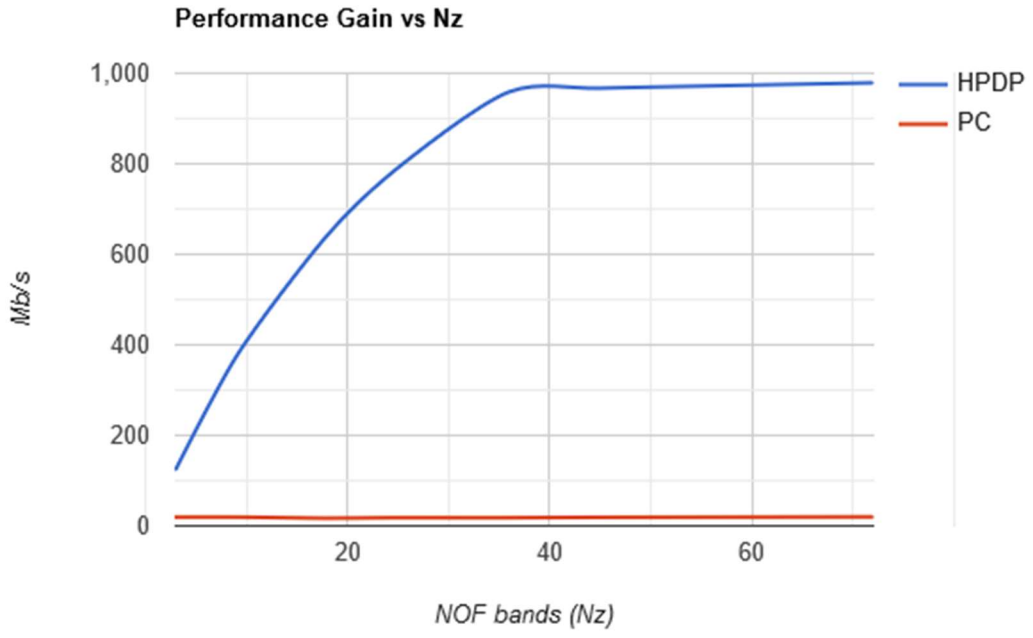

Figure 18: Comparison of performance

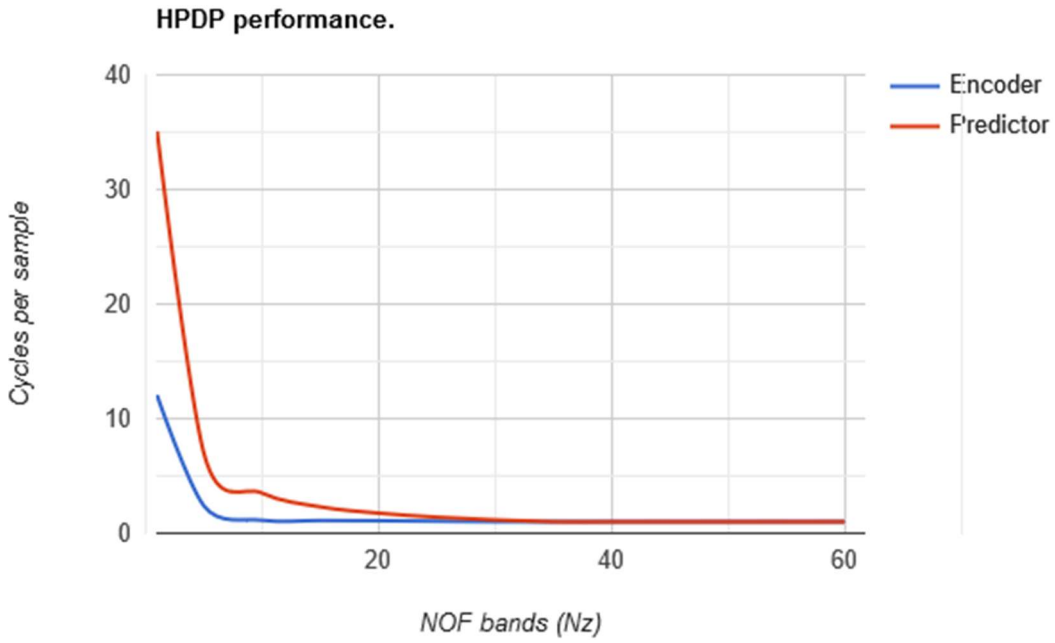

Figure 19: HPDP performance vs Nz

From the test performed is clear the performance advantage of the HPDP over the Simple PC setup. The conclusion that we extract from these results is the clear relationship between the Nz and the performance when it comes to the proposed implementation. It is shown in both Figure (36) and Figure (37) that after the point of 35 Nz we can achieve complete saturation of the device meaning that we need around one cycle per sample for prediction and one cycle for encoding. The differences in the throughput seen for inputs with Nz greater than 35 can be attributed to overhead delays and the delay of reconfiguring. The length of the delays is fixed and is not depended on the input's dimensions. This means that for larger inputs the delays are a smaller percentage of the run time contributing to greater throughput. The theoretical limit for the proposed implementation is around **1Gb/s**.

5.2 Conclusion

In conclusion, given the importance of Remote sensing technologies and especially Hyperspectral/Multispectral imagery, there has been a need for the development of new algorithms, technics, and standards so we can process, store and analyze the huge amount of data generated. Older system unable to adapt to these new standards create the need for a more versatile and efficient solution. This thesis reports our efforts to create a solution that tackles both problems by providing an implementation of a state-of-the-art low-complexity

lossless compression standard, ideal for scientific research, adapted to the features provided by a new space-suitable reconfigurable device. This combination allowed us to offer a competitive solution achieving high throughput at low power usage.

5.3 Future word

During the implementation phase of the compression standard, restrictions posed by the device's memory and available computing elements restricted us from implementing the coding configuration that would yield better compression rates. The block-adaptive encoding scheme would improve the compression rate considerably especially for low entropy inputs and large block sizes. The smallest encoded sample is caused mainly by 'zero' mapped residuals. For the current implementation the smallest possible code for a residual is 1 bit as even for mapped residual = 0 the RICE code produced for K=0 is '1'. For a low entropy input that produces many 'zero' residuals, this creates redundancy. The block - adaptive encoder deals with this issue by recognising all zero blocks and series of all zero blocks, encoding them with special codewords. That mean that depending on the chosen block size, the minimum number of bits per sample can be closer to 0.1 bit per residual.

The second extension of the current implementation is the supporting of 16-bit images. The current implementation is limited to 8-bit samples by the 16-bit arithmetic units. The limitation to 8-bit samples comes from the dot product of the weight and difference vector which will overflow for Weigh resolution larger than 4 or samples larger than 8-bit. In order to support 16-image the input should be split into the high and low bytes and then the two streams can be compressed individually in parallel, in two devices. This approach would create two streams with different level of entropy each. This happens because low bytes of 16-bit sample contains "higher frequency" information or "higher entropy". On the other hand, high bits usually show lower variability. Bellow part of a 16-bit hyperspectral image is presented as an example. It obvious that the high bytes (**bold**) exhibit very low variation.

```
00000000  00 bc 00 c3 00 c2 00 be  00 bf 00 bc 00 ba 00 ba
00000010  00 bc 00 b7 00 bf 00 bc  00 c1 00 bf 00 bc 00 c0
00000020  00 b9 00 bf 00 c0 00 b8  00 be 00 c1 00 c0 00 c1
00000030  00 be 00 bf 00 c1 00 c1  00 be 00 c4 00 c4 00 be
00000040  00 bc 00 c3 00 c2 00 bf  00 bc 00 bb 00 c0 00 bf
00000050  00 c0 00 bd 00 c0 00 be  00 c1 00 c1 00 bb 00 bf
```

This approach can possibly allow us to achieve good compression rates without sacrificing any of the performance.

Bibliography

[1] T. Helfers, G. Vines and C. Papadas, “HPDP-40 High Performance Data Processor-A New Generation Space Processor in Demonstration,” in *OBPD, ESTEC*, 2019.

[2] “FPGA Central,” 16 February 2008. [Online]. Available: http://www.fpgacentral.com/pld-types/mpga-mask-programmable-gate-array. [Accessed 20 December 2020].

[3] S. Hauck, “The roles of FPGAs in reprogrammable systems,” *Proceedings of the IEEE,* vol. 86, no. 4, pp. 615-638, April 1998.

[4] B. Zeidman, “EE Times,” 22 3 2006. [Online]. Available: https://www.eetimes.com/all-about-fpgas/. [Accessed 20 12 2020].

[5] “General Technoligies,” [Online]. Available: https://www.generatecnologias.es/en/fpga_architecture.html.

[6] V. Baumgarte, G. Ehlers, F. May and e. al., “PACT XPP—A Self-Reconfigurable Data Processing Architecture.,” *The Journal of Supercomputing,* vol. 26, p. 167–184, 2003.

[7] G. V. Vallduriola, T. Helfers, D. Bretz, M. Syed, D. Witsh, C. Papadas, V. Perel and S. Bartels, *High performance data processor (HPDP)-Image processing applications of a new generation space processor.*

[8] D. Suárez, T. Helfers, D. J. Weidendorfer, D. Bretz and D. J. Utzmann, “Space Debris Detection on the HPDP, a Coarse-Grained Reconfigurable Array Architecture for Space,” in *DSP Day 2016*, Gothenburg, 2016.

[9] C. Fischer and I. Kakoulli, “Multispectral and hyperspectral imaging technologies in conservation: current research and potential applications, Studies in Conservation,” tandfonline, 13 Dec 2006. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1179/sic.2006.51.Supplement-1.3.

[10] N. A. Hagen and M. W. Kudenov, “Review of snapshot spectral imaging technologies,” *Optical Engineering,* vol. 52, no. 9, 2013.

[11] “CCSDS,” [Online]. Available: https://public.ccsds.org. [Accessed 10 10 2020].

[12] J. Fjeldtvedt, M. Orlandić 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,* vol. 11, pp. 3841-3852, 2018.

[13] N. Aranki, A. Bakhshi, D. Keymeulen and M. Klimesh, “Fast and Adaptive Lossless On-board Hyperspectral Data Compression System for Space Applications”.

[14] CCSDS, *Lossless Data Compression CCSDS 121.0-B-3,* Washington: CCSDS, 2020.

[15] A. García, L. Santos, S. López, G. Marrero, J. F. López and R. Sarmiento, “High level modular implementation of a lossy hyperspectral image compression,” in *5th Workshop on*

Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Gainesville, 2013.

[16] CCSDS, *Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression.,* Washington: CCSDS, 2019.

[17] P. Mouroulis, B. V. Gorp, R. O. Green, H. Dierssen, D. W. Wilson, M. Eastwood, J. Boardman, B.-C. Gao, D. Cohen, B. Franklin, F. Loya, S. Lundeen, A. Mazer, I. McCubbin, D. Randall and B. Richardson, "Portable Remote Imaging Spectrometer coastal ocean sensor: design, characteristics, and first flight results," *Appl. Opt,* vol. 53, pp. 1363-1380, 2014.

[18] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 26, no. 2, pp. 203-215, 2007.