PARALLEL IMPLEMENTATIONS OF THE DISCRETE WAVELET TRANSFORM AND HYPERSPECTRAL DATA COMPRESSION ON RECONFIGURABLE PLATFORMS

APPROACH, METHODOLOGY AND PRACTICAL CONSIDERATIONS

by

Nazeeh Aranki

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2007

Dedication

To My Mother

Acknowledgements

I would like to express my deep gratitude and thanks to my advisor, Dr. Antonio Ortega, for his guidance, support and patience throughout the years I have been in the Ph.D. program at the University of Southern California.

I would like to extend my gratitude to Dr. Sandeep Gupta and Dr. Aiichiro Nakano for serving on my dissertation committee and for their guidance. I also would like to thank Dr. Richard Leahy and Dr. Alexander Sawchuk for serving on my qualifying exam committee.

I would like to thank my family and friends for their continued support. My special thanks to my colleagues and friends at JPL especially Raphael Some for his insight and inspiration, Jeff Namkung and my colleagues in the data compression group at JPL, Hua Xie, Aaron Kiely and Matt Klimesh, for their insight and enjoyable collaboration. I would like to thank my colleagues and former students in the group at USC, especially Wenqing Jiang.

This research was funded in part by the NASA-JPL Interplanetary Directorate (IND), the Remote Exploration and Experimentation Program (REE), and the Center for Integrated Space Microsystems (CISM).

Table of Contents

List of Tables

List of Figures

Abstract

This work was motivated by the need to dramatically reduce communication data rates for space based hyperspectral imagers. Key issues are compression effectiveness, suitability for scientific processing of retrieved data, and efficiency in terms of throughput, power and mass. We address the problem in three stages: first, development of a Field Programmable Gate Array (FPGA) hardware implementation of the parallel Discrete Wavelet Transform (DWT); second, development of a hyperspectral compression algorithm based on the wavelet transform and suitable for spacecraft on-board implementation; and third, development of an FPGA-based hyperspectral data compression “system on a chip” (SoC).

In developing our hardware implementation of the parallel DWT, our contributions are: a structured methodology for moving the 2D DWT, and similar algorithms, into reconfigurable hardware such as an FPGA; a specific representation for the DWT that provides an architecture suitable for efficient hardware implementation; and a data transfer method that provides seamless handling of boundary and transitional states associated with parallel implementations. The resultant new

implementation produced significantly improved performance over previous methods.

In developing our hyperspectral data compression algorithm, our contributions are: a DWT based algorithm, capable of both lossy and lossless compression, that can be tailored to accommodate any scientific instrument, and that is suitable for on-board hardware implementation; algorithm components that are efficiently designed for three dimensional data, for implementation in hardware, and that achieve results comparable to or exceeding previous optimized algorithms at a lower computational cost; the discovery of, and development of mitigation techniques for, a new artifact-producing phenomenon encountered when using the 3D DWT for compression; and a new technique for region-of-interest compression of hyperspectral data that uses “virtual scaling” which satisfies low memory requirements and provides better compression effectiveness.

In developing our FPGA-based SoC, our contributions are: development of a scalable embedded implementation for the 3D DWT hyperspectral data compression; a novel priority-based data formatting and localization technique for bit-plane encoding that provides substantial improvements in throughput efficiency compared to standard techniques; and extension of the wavelet

transform methodology developed in the first part to hybrid Hardware/Software SoC implementations.

Chapter 1

Introduction

1.1 Motivation and Background

Our motivation to investigate compression techniques for hyperspectral data derives from two primary factors: first, the limited bandwidth available to spacecraft communication channels, combined with the extremely high data rates of modern hyperspectral imagers, has become a severe limitation to current and planned space missions; and second, the severe limitations in power and mass budgets for spacecraft subsystems, combined with the high data rates of hyperspectral instruments, requires extremely low power, highly efficient implementations of any required functions. The current state-of-the-practice in NASA space missions is either limited lossless compression, or no compression, of hyperspectral imagery. For example, the deep space Cassini mission [28] uses the lossless Rice chip (Universal Source Encoder for Space (USES)) [107], while the earth orbiting EO1[41] and Atmospheric Infrared Sounder (AIRS) [19] missions have no on-board data compression. Upcoming NASA hyperspectral instruments, whether deep space or Earth orbiting, such as FLORA [17] and the Plant Physiology and Functional Types (PPFT) [45], will be capable of returning an unprecedented amount of science data, but will require effective compression techniques to realize their full potential.

The need to deploy an efficient algorithm that that can be progressive, lossless and/or lossy, and that meets a broad range of application needs and bandwidth requirements, motivated our decision to select the discrete wavelet transform as the basis for our algorithm. In addition, the Consultative Committee for Space Data Systems (CCSDS) recently recommended the use of DWT based compression in future space missions [52]. Furthermore, hyperspectral data exhibits high correlations in the spectral domain as well as in the spatial domain, which makes three dimensional DWT based coding a suitable candidate for data decorrelation. Speed, power, mass and real-time processing constraints prevent many missions from performing compression in software. The computationally intensive nature of a 3D DWT based algorithm strongly implies that any practical solution requires a parallel hardware approach. As discussed in the next paragraph, the cost of development and fabrication, ability to adapt operational parameters and to tune the system for any specific instrument and mission require an FPGA based implementation of any such hyperspectral compressor.

The need for a fast hardware DWT that allows flexibility in customizing the wavelet transform with regard to the filters being used and the structure of the wavelet decomposition, motivated us to target Field Programmable Gate Arrays (FPGAs). FPGAs offer a suitable platform (cost effective and highly flexible) for such an

implementation; they provide reconfigurable, evolvable, remotely repairable and upgradeable system elements. FPGA-based systems represent a new paradigm in the industry – a shift away from individual custom ASIC solutions for each application, to a single hardware assembly (FPGA) that can be reconfigured to accommodate multiple applications and multiple modes of operation. This also provides many advantages over ASIC designs in terms of flexibility of field upgrades, reliability and fault tolerance via reconfiguration to repair and work around in-field failures. Future commercial and space applications can benefit from this flexibility to enable remote repairability and upgradeability.

We first start with the hardware implementation of the 2D DWT. As a result of extensive research in recent years, DWT-based transform coding techniques are central to many modern image and video encoding algorithms. Examples include; JPEG2000 image codec, CREW image compression, AWARE's MotionWavelets and motion-compensated 3D DWT for video encoding [57][24][83][91]. Consequently, efficient software and hardware based transform coding system designs and implementations are a high priority objective at academic, commercial and government research centers. However, while the wavelet transform offers a wide variety of useful features, it is computation intensive. Our aim in this part of the research was to develop efficient 2D DWT FPGA implementations, and a general methodology for parallel implementations of the wavelet transform that can be

extended to other DWT filters and to similar DSP problems. Then we developed a hyperspectral data compression algorithm based on the 3D wavelet transform and tailored for efficient FPGA hardware implementations. Lastly, we extended our FPGA implementation methodology to a hybrid hardware/software "system on a chip" (SoC) FPGA-based implementation for the hyperspectral data compression system.

This thesis investigates the following three topics related to practical system designs based on the discrete wavelet transform as a tool for signal processing and data compression.

1.2 FPGA Parallel Implementation of the 2D Discrete Wavelet Transform

We investigated an FPGA block-based parallel implementation which utilizes various overlapping technique, and developed a methodology for such implementations.

1.2.1 Parallel DWT Architectures

The 1D DWT decomposition can be implemented as a pyramidal recursive filtering operation, also known as the Mallat decomposition [78]. The process for the 2D DWT decomposition for each level is implemented with a cascaded combination of two 1-D wavelet transforms where the data is row transformed first and then column

transformed. To save computations, lifting factorization was introduced to implement the DWT [94]. The lifting algorithm models the DWT as a finite state machine (FSM), which takes advantage of filterbank factorizations such as lattice factorizations [104][105] and uses Daubechies and Sweldens “lifting" scheme [36]. The finite state machine updates (or transforms) each raw input sample (initial state) progressively, via intermediate states, into a wavelet coefficient (final state). It has been shown that lifting-factorization based DWT algorithm can be twice as fast when compared to the pyramidal algorithm [36], and that motivated us to base our parallel implementations on the lifting algorithm.

Since DWT is not a block based transform, problems of speed and artifacts arise near the block boundaries. Overlapping techniques can handle the block boundaries with minimum storage requirements and inter-block communication overhead. Our goal was to achieve efficient performance under practical considerations and real world constraints in: on-chip storage, external memory bandwidth, target hardware platform, power and mass. In our implementations, we consider three different architectures: i) *overlapping* – which overlaps boundaries of image blocks and transforms each block independently, ii) the *overlap-save* technique– which saves the boundary information and exchanges them between blocks at the transform level, and iii) the *Overlap-State* technique [53][54][55]- which saves the states of the partially transformed boundaries for a one time exchange between blocks at the end

of the transform operation (fully detailed in Chapter 3). For the overlap-state implementation, a new data transfer method that utilized an efficient DMA was designed and implemented, and a new architecture for the parallel DWT, in which transitional boundary data (i.e. states) are stored on-chip and passed to the DWT kernels in a seamless manner that minimizes the overhead usually associated with parallel implementations. Figure 1.1 shows a high level block diagram of our FPGA parallel DWT system.

Our final implementations demonstrated improved performance by a factor of 1.4 to 3 over previously reported methods.

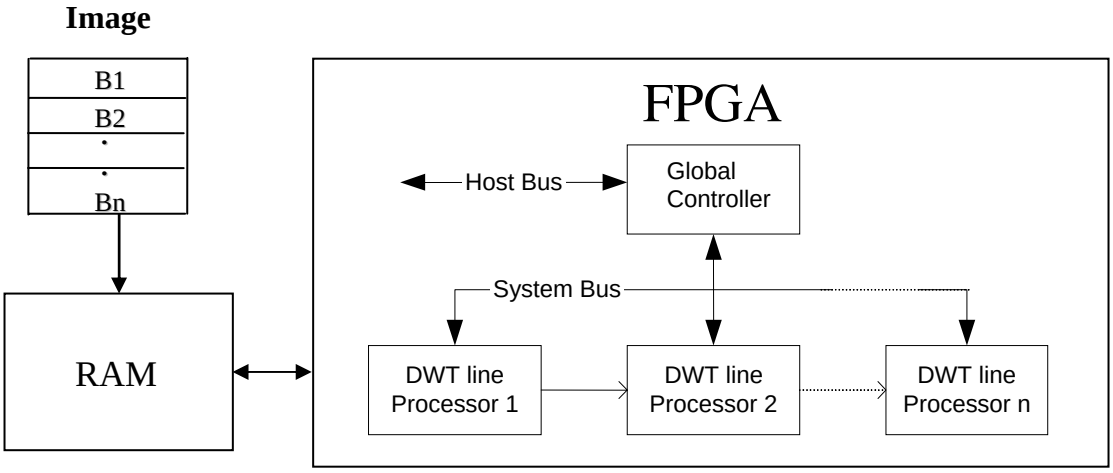


Figure 1.1: Scalable parallel based system for the 2D discrete wavelet transform

1.2.2 Implementation Methodology for FPGA-Based Design of DWT Parallel System

Our goal was to design an efficient scalable FPGA implementation for the 2D DWT that utilizes parallel architectures with no blocking artifacts and that is based on lifting factorization to ensure lower computational cost. Our methodology identifies the design steps, the problems, the limitations and the design issues associated with each step of the implementation. It then leads the designer through a series of analyses and decisions that produce an efficient design. Practical considerations appear in the form of constraints that may force, or heavily weigh, a specific design choice.

Our methodology starts by identifying a specific lifting factorization for the DWT that provides architectures that are efficient for hardware implementations by maximizing the number of shift and addition operations, while minimizing the number of multiplications. It then proceeds to identify an architecture for the DWT filter kernel design that maximizes the 2D processing performance. Available on-chip memory storage and external memory bandwidth determine the range of parallel architecture choices and the degree of design scalability. Finally, consideration of throughput, resources utilization and power determine the final design. Our methodology was used to develop the system implementation described in Chapter 3.

1.3 Hyperspectral Data Compression

1.3.1 Overview

Hyperspectral images are sequences of spatial images, generated by imaging spectrometers or sounders, which record the spectral intensities of the light reflected by the observed area. Because of their three dimensional nature, (two spatial and one spectral), hyperspectral images are often referred to as data cubes. They have a wide variety of Earth remote sensing uses in the study of oceans, atmosphere, and land [41][106]. Hyperspectral imagers are also used in deep-space missions to map the surface minerals and chemical features of planets and their moons [29]. Hyperspectral imaging spectrometers and sounders have a large number of channels ranging from several hundred as in JPL's Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) [106], which contains 224 spectral bands, to thousands as in the case of AIRS [19], which has 2378 infrared channels. These instruments can generate high data rates that cannot be sustained with existing communication links. Efficient on-board compression is needed to reduce bandwidth and storage requirements. In this thesis, we investigate a low-complexity wavelet based approach for three dimensional coding of hyperspectral data cubes suitable for on-board processing and for hardware implementation on reconfigurable platforms. While many hyperspectral applications can tolerate low bit rate lossy data compression, applications such as studies of the atmosphere and of planet surfaces require high accuracy, for retrieval of temperature and humidity profiles, identifications of

atmospheric anomalies and planet surface details (for example, in search of water or life). A lossless or virtually-lossless approach is required to address such applications. Our overall goal is to meet the requirements of various instruments and their applications with a highly efficient compression system that can be both lossless and lossy.

Our compression strategy utilizes a general-purpose methodology for hyperspectral data compression that maximizes compression by fully exploiting spectral (inter-band) correlations as well as spatial correlations, subject to practical considerations and constraints. The algorithm must be: applicable to different spectral resolutions; capable of lossless and lossy compression; low-complexity and suitable for on-board hardware implementation; progressive and suitable for push-broom type instruments and their data formats (hyperspectral data is most often collected in the form of a band interleaved by pixel (BIP) or sometimes in a band interleaved by line (BIL) manner).

1.3.2 3D Coding for Hyperspectral Images

Conventional image compression techniques aim to remove spatial redundancies. Hyperspectral data cubes have large spectral (inter-band) correlations as well as spatial correlations. An efficient compression strategy for hyperspectral data would seek to maximally exploit both spectral and spatial correlations. For lossless compression, effective prediction approaches were proposed such as DPCM [4].

Hybrid schemes were proposed to solve the lossy problem. Here, spectral redundancies are removed through spectral transformation techniques such as principal component analysis (PCA) by selecting the dominant spectral bands and then applying 2D compression such as DCT or DWT [89][25]. At the system level a single scalable algorithm that provides lossless and lossy compression is often preferable to the implementation of multiple algorithms. Our goal was to develop such an algorithm.

Due to its inherent efficiency in data compression, we decided to investigate the discrete wavelet transform as the bases of our 3D coding scheme. Figure 1.2 shows a block diagram of the 3D coding method. Our investigation focused on utilizing and extending compression algorithms that have proven their effectiveness in compressing 2D images, and in addition, can be easily implemented in FPGA hardware. The three dimensional DWT-based algorithm we developed was adapted from the JPL's 2D image compression, ICER [62], hence the name ICER-3D [63]. ICER-3D is a progressive compression algorithm which: (i) is lossy and lossless; (ii) is line-based, operating in scan-mode to minimize storage requirements; and (iii) accommodates pushbroom imaging sensors making it readily portable to on-board hardware implementations. 3D DWT based coding has recently been proposed by research efforts such as 3DSPIHT [40], 3D SPECK [95], and the proposed 3D JPEG2000 3D [58]. Our compression efficiency, as will be shown in Chapter 4, is

comparable to the efficiency of these algorithms, but our approach is distinguished by its lower complexity post 3D DWT encoding, which makes it suitable for spacecraft on-board deployment and for efficient hardware implementation.

Following the wavelet decomposition, the means of the low frequency subband planes are subtracted in preparation for the post transform coding. A context modeling scheme and an entropy coder were ported from the ICER image compressor and extended to accommodate three dimensional data sets. Each DWT coefficient is converted to sign-magnitude form. Bit planes of subbands are compressed one at a time and compressed bit planes of different subband cubes are interleaved, with the goal of having earlier bit planes yield larger improvements in reconstructed image quality per compressed bit. Subband bit planes are compressed in order of decreasing priority value according to the simple priority assignment scheme described in Chapter 4. At each step of the algorithm design, hardware implementation issues were addressed and designs were generated that reduced the complexity of the hardware implementation and increased throughput while maintaining high compression efficiency. The specific hardware oriented implementation of the algorithm will be referred to as ICER-3D-HW.

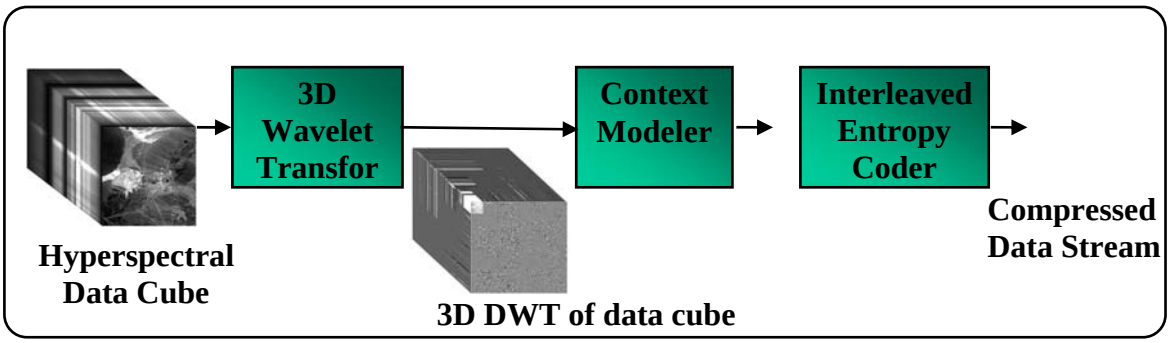


Figure 1.2: 3D Hyperspectral Compressor Block Diagram

Issues arising from extending a 2D DWT compression approach to 3D DWT were encountered and addressed. For example, when a 3D DWT is used for decorrelation, "ringing" artifacts in the spectral dimension can cause the spatially low-pass filtered subbands to have large biases in the individual spatial planes. Specifically, spatial planes of spatially low-pass subbands contain significant biases that vary from plane to plane. This problem is unique to multi and hyperspectral data; an analogous artifact does not generally arise in 2D images. This phenomenon hurts the rate-distortion performance at moderate to low bit rates (~1 bit/pixel/band and below), and occasionally introduces disturbing artifacts into the reconstructed images. We analyzed the *"spectral ringing"* problem and proposed mitigation schemes. The simplest scheme, which was adopted for the hardware implementation, subtracts the mean values from spatial planes of spatially low-pass filtered subbands prior to encoding, thus compensating for the fact that such spatial planes often have mean values that are far from zero. The resulting data are better suited for compression by

methods that are effective for subbands of 2D images such as the ones we are using in the main algorithm. Compression effectiveness was improved by about 10% after applying the mean subtraction scheme.

Region-of-Interest (ROI) for hyperspectral data was also addressed and a scheme for "*virtual scaling*" of high priority data was designed which provides better compression performance and has low memory requirements (suitable for future hardware implementation).

Lossless compression performance of ICER-3D-HW benchmarked on AVIRIS 1997 data sets [20] shows excellent results when compared to all 2D approaches. Improved (or comparable) results are obtained when compared to other 3D approaches. ICER-3D-HW, however is outperformed by the JPL developed, "fast lossless" compressor [71], which was designed and optimized for performing only lossless compression. AVIRIS data sets were losslessly compressed, on average, from 16 bpp down to 5.63 with ICER-3D-HW and down to 5.10 with "fast lossless". Lossy compression comparisons show significant gains in compression efficiency over other state-of-the-art 2D techniques, and comparable results to other optimized 3D algorithms. We show gains of at least 40% compression efficiency over the ICER 2D image compressor.

1.4 Scalable Embedded Hyperspectral Data Compression on Reconfigurable Platforms

1.4.1 System Design and Practical Considerations

The 3D DWT compression algorithm, ICER-3D-HW, described in the previous section, is computationally intensive, causing real-time processing difficulties (data rates for some instruments can go to 100s of Gbits/sec). High speed processors are power hungry and do not fit in NASA's vision of next generation spacecraft. Dedicated hardware solutions are highly desirable - offloading the main processor, while providing a power efficient solution at the same time. We investigated an efficient scalable parallel implementation on an FPGA platform. We designed cascaded line-based wavelet transform modules, which allow the wavelet transform in the 3D DWT case to be computed as the lines of the image data cube arrive rather than waiting for an entire frame of data, thus efficiently accommodating pushbroom sensors. The other key modules of the hardware compressor, the context modeler and the interleaved entropy coder, were designed for efficient throughput performance, utilizing a priority–based data formatting and localization technique that transposes bit-planes after the 3D DWT decompositions and stores them in memory locations that are readily available for the priority indexed bit-plane encoding discussed in Chapter 4. This formatting scheme accelerates encoding by a factor of more than 10:1 over standard techniques.

The implementation targets state-of-art FPGAs, such as Xilinx Virtex II pro. These FPGAs provide on-chip, hard-wired PPC 405 processor cores [113], allowing them to be used to form an embedded platform for SoC implementations. Such platforms allow efficient partitioning of the algorithm into software and hardware modules so as to take full advantage of the available hardware resources. In our system, the on-chip processor was used to implement the system global controller to manage the overall operation of the compression system and internal and external data transfers. Figure 1.3 illustrates the final hyperspectral compressor. The parallel hardware compression system is capable of acceleration of up to two orders of magnitude vs. a software implementation running on current state-of-the-art processors.

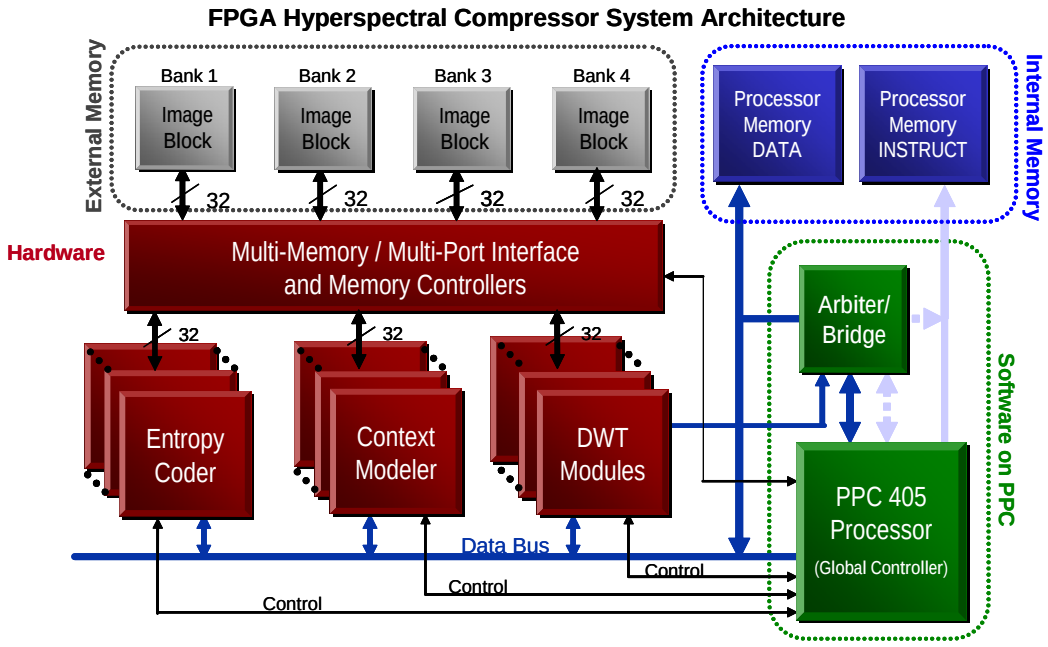


Figure 1.3: System on Chip FPGA Hyperspectral Data Compressor

1.4.2 Implementation Methodology

We extended the 2D DWT methodology developed in the first part of this thesis to hybrid Hardware/Software SoC FPGA implementations

In addition to the steps that address the practical considerations, constraints, limitations and design choices listed for the 2D DWT methodology, our SoC methodology addresses the issue of SW/HW partitioning of algorithm modules. Partitioning is done after performing software profiling to identify the most appropriate candidates for hardware acceleration. Dynamic range expansion studies for the DWT are also performed in the context of 3D processing and long pixel depth data. Finally, scalability issues were addressed, and trade-off studies for speed, hardware resources utilization and power were performed to complete the design.

1.5 Outline and Contributions of this Thesis

The main contributions of this research are:

- *A methodology for FPGA based parallel architectures and implementations of the Discrete Wavelet Transform*. We investigated and analyzed parallel and efficient hardware implementations targeting state-of-the-art FPGAs. We addressed practical considerations and various design choices and

decisions at all design stages to achieve efficient DWT implementations subject to a given set of constraints and limitations.

- *A specific lifting representation for the* DWT that provides architectures suitable for efficient hardware implementation.
- *A novel data transfer method that provides seamless handling of boundary and transitional states* associated with parallel implementations.

- *A Low-complexity algorithm for hyperspectral data compression suitable for hardware implementation*. We investigated the problem of on-board hyperspectral data compression for space based systems and adapted a 2D DWT based algorithm to 3D data sets to provide both lossy and lossless compression. We addressed issues arising from extension of 2D DWT to 3D DWT as well as hardware implementation considerations.
- *Discovery of and development of mitigation techniques for, the spectral ringing artifacts phenomenon* encountered when using the 3D DWT for compression.
- *Region-of-interest compression for hyperspectral data using "virtual scaling"* that results in low memory requirements and improved compression effectiveness.

- *A methodology for scalable embedded FPGA based implementation of complex 3D compression systems.* We extended the wavelet transform methodology developed in the first part to hybrid Hardware/Software SoC FPGA implementations We addressed issues of SW/HW partitioning of algorithm modules, dynamic range expansion for the DWT, scalability of design, and trade-offs to meet practical considerations and constraints.
- *A scalable embedded implementation* of 3D DWT based hyperspectral data compression – a single chip solution to hyperspectral data compression.
- *A novel priority-based data formatting and localization technique for bit-plane encoding* providing more than 10x in throughput efficiency compared to standard techniques.

This thesis is organized as follows:

Chapter 2 provides an overview of hyperspectral data and the hyperspectral data compression system and its characteristics.

Chapter 3 addresses in detail the parallel implementation methodology of the 2D parallel DWT. We provide detailed descriptions of the methodology, practical constraints, design parameters selection, analysis and trade-offs among three overlapping FPGA parallel implementations. Analysis, simulations and

implementations for the (9,7) DWT are also presented along with comparisons to other DWT architectures and implementations.

Chapter 4 addresses the problem of hyperspectral data compression. Compression approach and strategy are presented. Detailed description of a three dimensional DWT coding algorithm suitable for hardware implementation, ICER-3D-HW, is presented. Description of individual compression modules is provided. Compression results and comparisons to current state-of-the-art approaches are also detailed.

Chapter 5 addresses the implementation of the hyperspectral compressor on a reconfigurable platform. A detailed design approach and methodology are presented. A detailed FPGA implementation of the ICER-3D-HW is described along with performance analysis and description of the hardware resources and their utilization.

Chapter 6 completes the thesis with summary, conclusions and potential future work.

Chapter 2

Hyperspectral Data Compression: System Overview

2.1 Introduction

Hyperspectral images are three dimensional data sets (two spatial and one spectral) that consist of hundreds of narrowly spaced spectral bands. These bands are generated by hyperspectral imaging spectrometers or sounders and comprise the reflectance, at different wavelengths, of the region being viewed by the scanning instrument. They are powerful tools for many applications, such as detection and identification of land surface and atmosphere constituents, studies of soil and monitoring agriculture, surveillance, studies of the environment and the ozone, and weather prediction [106][41][19]. They are also used in deep-space missions to map surface minerals and chemical features of planets and their moons. For instance, Cassini's *Visible and Infrared Mapping Spectrometer Subsystem (VIMS)* gathers hyperspectral images of Saturn, its rings and its moons [29]. FLORA and PPFT are proposed Earth orbiting instruments that will provide global, high spatial resolution measurements of ecosystem disturbances, vegetation composition, and productivity, including interactive responses to climate variability and land-use change [18][27][80].

Recent breakthroughs in hyperspectral imaging and sounding technologies (such as IR detector arrays and cryocooler technology), have resulted in far more spectral channels and higher spectral resolution than earlier generations. For example, *High Resolution Imaging Spectrometer (HIRIS)* [46] has 192 channels, AVIRIS [106] has 224 spectral bands and AIRS [19] has 2378 IR channels and is often referred to as *ultraspectral.*

Hyperspectral imaging instruments are capable of producing enormous volumes of data that quickly overwhelm the space communication downlinks and require massive on-board storage capabilities. Effective on-board techniques for compressing such data sets are essential to overcome downlink limitation and make efficient use of on-board storage.

In this chapter we present an overview of a wavelet based compression system for three dimensional coding of hyperspectral data cubes suitable for on-board processing and its hardware implementation on reconfigurable platforms. Many hyperspectral applications such as classification and content retrieval may tolerate low bit rate lossy compression, but studies of the atmosphere and planet surfaces require high accuracy for retrieval of temperature and humidity profiles and planet surface details. To address the need of both types of applications, our algorithm was

designed to be progressive and capable of lossy and lossless compression to meet data rates requirements and address scientific needs.

This chapter is organized as follows. Section 2.2 provides an overview of hyperspectral data. Section 2.3 gives an overview of the 3D compression algorithm and its compression results. Section 2.4 describes a scalable embedded hardware implementation of the compression system, and finally section 2.5 concludes this chapter with a summary and conclusions.

2.2 Hyperspectral Data: An Overview

Determining the composition of, and inferring the processes active on, the Earth and other planetary surfaces, by counting photons (energy) at the top of the atmosphere (or from space), is a challenging problem. Spectroscopy provides a framework based in physics to achieve this remote measurement objective in the context of the interaction of photons with matter. The physics, chemistry and biology of spectroscopy are validated through more than 100 years of laboratory, astronomical and other observational research and applications [106][27]. Spectral sensor instruments acquire data at different wavelengths. Imaging spectrometers provide the spectral data for each pixel in an image, quantified into discrete levels of brightness. An example of this type of instrument is hyperspectral imaging sensors. These sensors acquire data in a vast number of narrow and contiguous spectral bands, thus

the use of the term hyperspectral. Examples of hyperspectral data utilized in this research are gathered mostly by AVIRIS, but a few examples use data sets from Hyperion and AIRS instruments.

AVIRIS is an airborne hyperspectral imaging instrument that has been providing valuable information about Earth since the late eighties. The AVIRIS concept [106] is illustrated in Figure 2.1. It consists of 224 spectral bands, with each pixel having 12 bits of precision. It has a spectral range of 3 nm to 2.5 μm, covering the visible and near infrared regions. The pixel size and swath width of the AVIRIS data depend on the altitude from which the data is collected. When collected by the ER-2 (20km above the ground) each pixel produced by the instrument covers an area approximately 20 meters diameter on the ground. When collected by the Twin Otter (4km above the ground), each ground pixel is 4m square. The images obtained from AVIRIS can have a size of up to several Gbytes. They have spatial lines of 614 pixels extended over the region of interest. The sets analyzed in this chapter were divided into images of approximately 130Mbytes for ease of handing (512 spatial lines of 614 pixels each, across 224 spectral channels). An example of an image cube generated from AVIRIS is shown in Figure 2.2. The Hyperion spectrometer is space-borne as part of the Earth Orbiting 1 mission (EO1) [41]. It consists of 242 spectral bands ranging from 0.4 - 2.5 μm, with each pixel having 12 bits of precision. After elimination of unusable bands (due to either noise or poor sensor pointing), we

analyzed images of 220 spectral bands. Hyperion complements AVIRIS and addresses a broad range of issues and world-wide sites, from agriculture in Australia, to glaciers in Antarctica, to grasslands, minerals and forests in the Americas [23]. The AIRS instrument [19] is aboard NASA's Aqua spacecraft that was launched in 2002. It employs a 49.5 degree cross-track scanning with a 1.1 degree instantaneous field of view (see Figure 2.3) to provide twice daily coverage of essentially the entire globe. The AIRS data consists of 2378 infrared channels in the 3.74 to 15.4 μm region of the spectrum divided into three contiguous sets of bands. Data is gathered as scanlines containing 90 cross-track footprints per scan line, where a footprint consists of 2378 pixels of infrared data covering the same surface region. The objective of AIRS is to provide improved global temperature and humidity profiles to meet NASA's global change research objectives and the National Oceanic and Atmospheric Administration's (NOAA) operational weather prediction requirements.

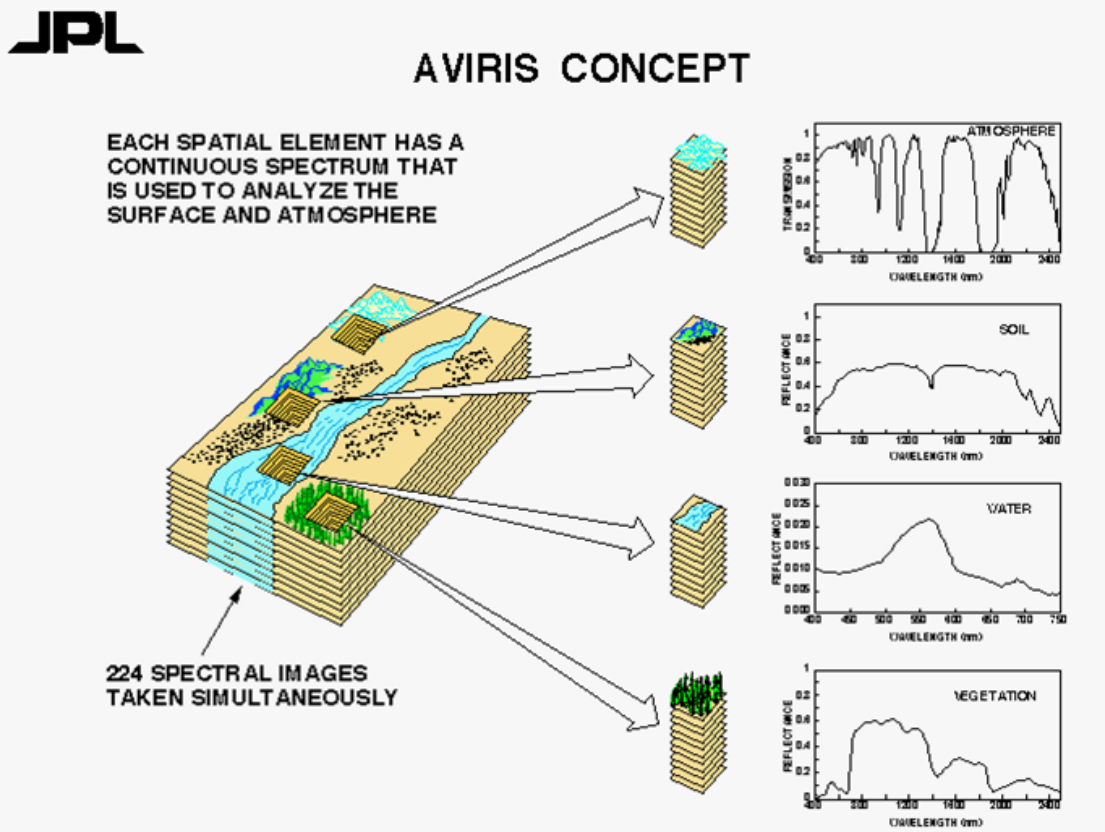
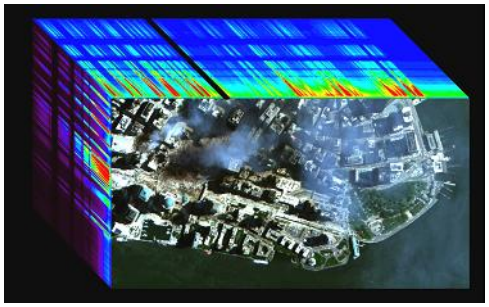


Figure 2.1: AVIRIS Concept and Data



(a) (b)

Figure 2.2: AVIRIS hyperspectral data "cubes"

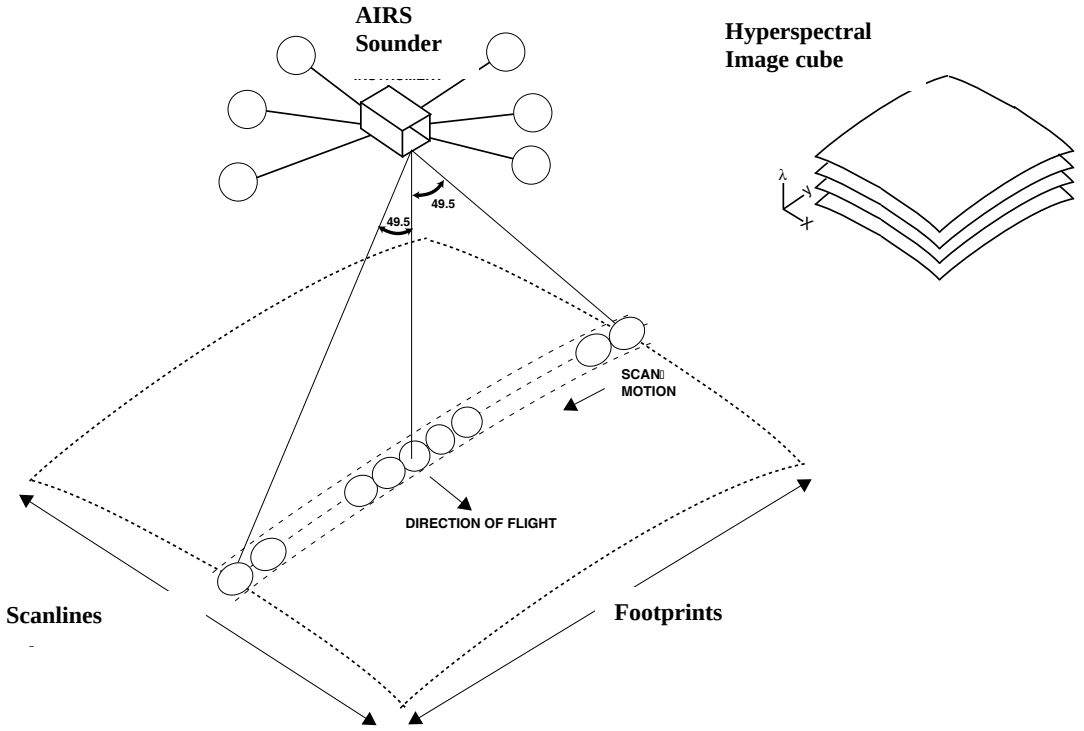


Figure 2.3: Hyperspectral Scan Geometry for AIRS

Data sets were analyzed for spatial and spectral correlations to guide the compression algorithm development. It was observed that the data sets do indeed exhibit high spectral redundancies/correlations in addition to spatial correlations. To analyze these spectral redundancies, normalized correlations within a single footprint of data across a vector of all spectral bands were calculated. Figure 2.4 shows plots of these correlations for footprints in two different scan lines of AIRS simulated data. Table 2.1 lists different characteristics of the analyzed data sets, as well as their average correlations.

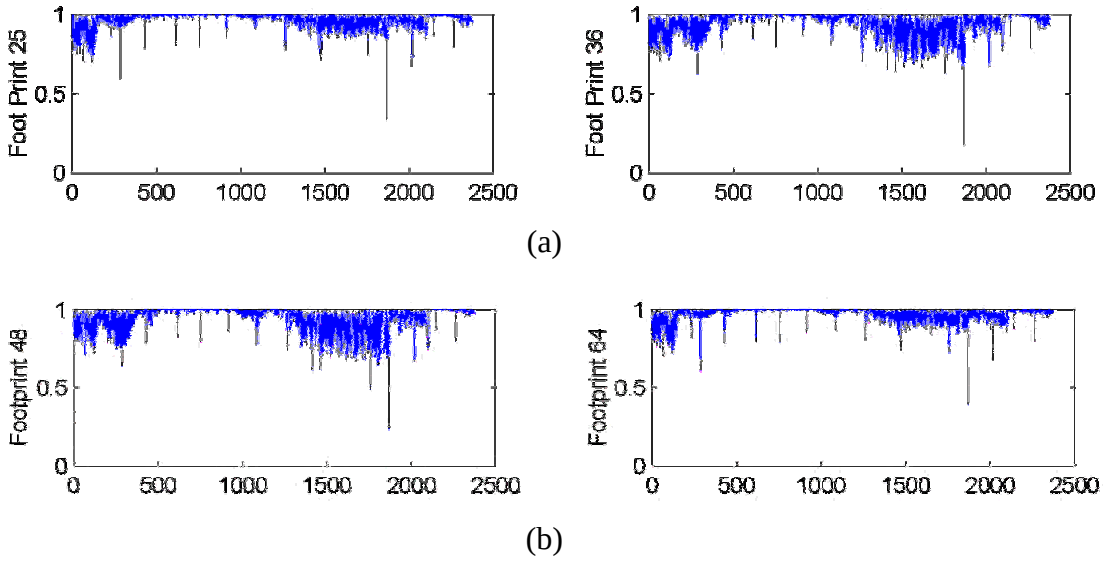


Figure 2.4: Correlations in AIRS Simulated Data. (a) Scan Line 1 (b) Scan Line 50

Table 2.1: Hyperspectral Instruments and their Specifications

Instrument	No. of Spectral Channels	Bits per pixel	Wavelength Range (nm)	Spectral Resolution	Spatial Resolution	Average Spectral Correlations
AVIRIS	224	12-16	370 - 2500	9.8 nm	2 to 20 m	0.89
Hyperion	242 (220 useable)	12	400 - 2500	10 nm	30 m	0.85
AIRS	2378	16	3740 - 15400	3.5-12 nm	15 Km	0.91

2.3 Three Dimensional Coding

The Space-born state-of-the-practice in compression, is to use the Rice encoder (i.e. CCSDS USES chip [107]). The Rice encoder provides lossless compression at generally accepted performance levels. The performance of the Rice encoder, however, is insufficient for high volume data transmission over most communication

downlinks (e.g. it was used in the Cassini mission, which was tailored to meet the capabilities of the Rice encoder by reducing the collected data volume). Most missions to date (such as in EO1, AVIRIS, AIRS instruments) have avoided even lossless compression or used 2D DWT/DCT coding for spatial bands, such that spectral redundancy is not exploited. Modulated Lapped Transform (MLT) [50] and PCA techniques are used mostly in department of defense (DoD) and NOAA applications for context retrieval. These techniques are lossy and hence do not meet data fidelity required by most NASA scientists. Hence, our objective was to investigate and develop hyperspectral data compression methods capable of lossless and lossy compression and suitable for on-board deployment (i.e. hardware implementation) that will significantly reduce the data volume necessary to meet science objectives in future deep-space and earth orbiting missions.

2.3.1 3D Wavelet coding

Since the wavelet transform has shown great results for compression of images, a discrete wavelet transform approach was chosen to meet our compression objectives. Wavelet based compression is also recommended by the CCSDS committee [52] and is currently used in the ICER compression software on-board the Mars Exploration Rovers (MER) [64][65]. Our 3D compressor extends an efficient 2D image compression algorithm, namely the ICER image compressor, to adapt to 3D hyperspectral data sets. It is progressive, and based on the reversible integer discrete

wavelet transform, making it capable of lossless and lossy compression in a single algorithm.

A block diagram of the compression process was shown in Figure 1.2. It should be noted that some of the individual components of our algorithm have been used for multispectral and hyperspectral image compression by other researches. The selective combination of our algorithm steps, our enhancement techniques, as well as the low-complexity and suitability for hardware implementations, provide our novelty. Our compressor starts by extending the 2D DWT decomposition to 3D DWT, then it adapts the ICER bit-plane encoding scheme to 3D subband cubes by using a priority scheme tailored to 3D data and applying a 2D context modeler to subband cubes plans followed by an interleaved entropy coder [63]. It was tailored for hardware implementation by employing design choices that simplify the computation complexity, enhance the hardware throughput and /or minimize the needed hardware resources. We will refer to this compressor as ICER-3D-HW.

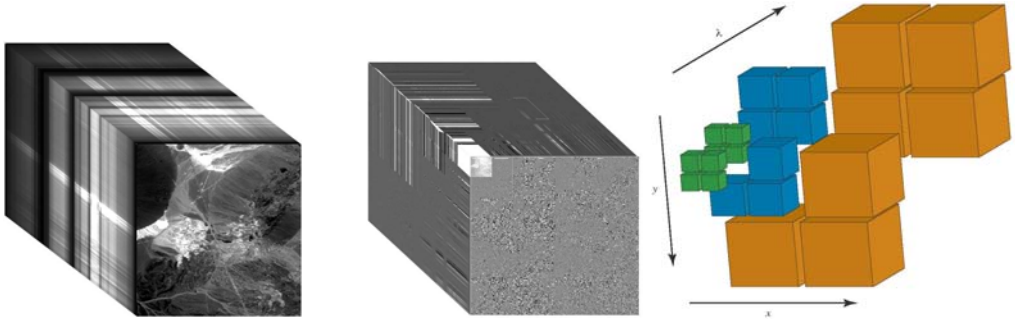


Figure 2.5: 3D wavelet decomposition for an AVIRIS data set covering Cuprite, NV site

The DWT component of the algorithm provides options for selecting among several popular DWT filters such as (5,3), (9,7), (2,10), (2,6), (5,11), and (13,7) [62][63][1]. Most of the experiments presented in this thesis are based on the performance of the (2,6) DWT due to its suitability for hardware implementations, as will be detailed in the following chapters. An example of a three level 3D DWT decomposition is shown in Figure 2.5.

2.3.2 Compression Results

The results shown in Figure 2.6 indicate that ICER-3D-HW achieves more effective lossless compression than simple two-dimensional approaches or the USES (Rice chip) compressor when used in its multispectral mode. ICER-3D-HW compression efficiency is also comparable to most 3D algorithms we benchmarked as will be shown in Chapter 4. It is outperformed, however, by the JPL fast lossless compressor of [71], which was designed as a lossless only compressor. However, it is reasonable to use ICER-3D-HW for lossless compression in an application where it is also required to perform lossy compression. Figure 2.6 shows results for AVIRIS 1997 calibrated data (16 bits) [20] compared to ICER [62], USES in multispectral prediction mode [107], and “fast lossless”.

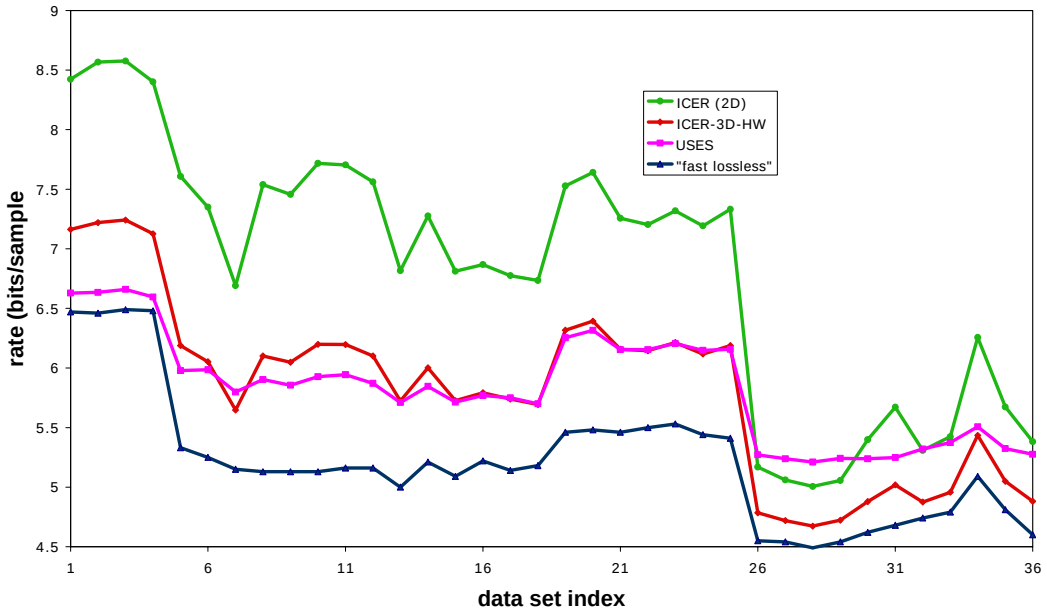


Figure 2.6: Lossless Results - Average rate in bits/pixel – ICER (2D): 6.61, USES (3D): 5.80, ICER-3D-HW: 5.63, "fast lossless": 5.10

Figure 2.7 shows the improved rate-distortion performance of ICER-3D-HW in comparison to the ICER 2D image compressor for a typical AVRIS image. To show this improved performance visually, Figure 2.8 displays details from false-color images produced from the reconstructed AVIRIS scenes after compression at 0.25 bits/sample. In chapter 4, we show more results and comparisons between ICER-3D-HW and other existing lossless and lossy 3D approaches.

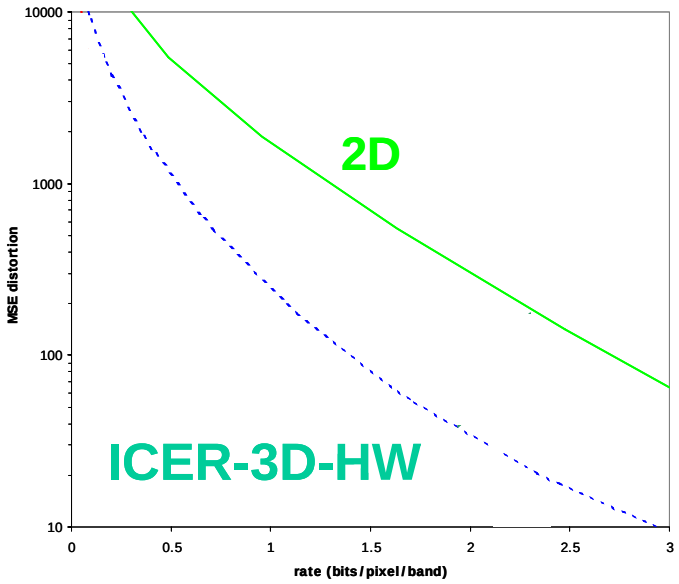


Figure 2.7: Achievable lossy compression for ICER (2D) and ICER-3D-HW

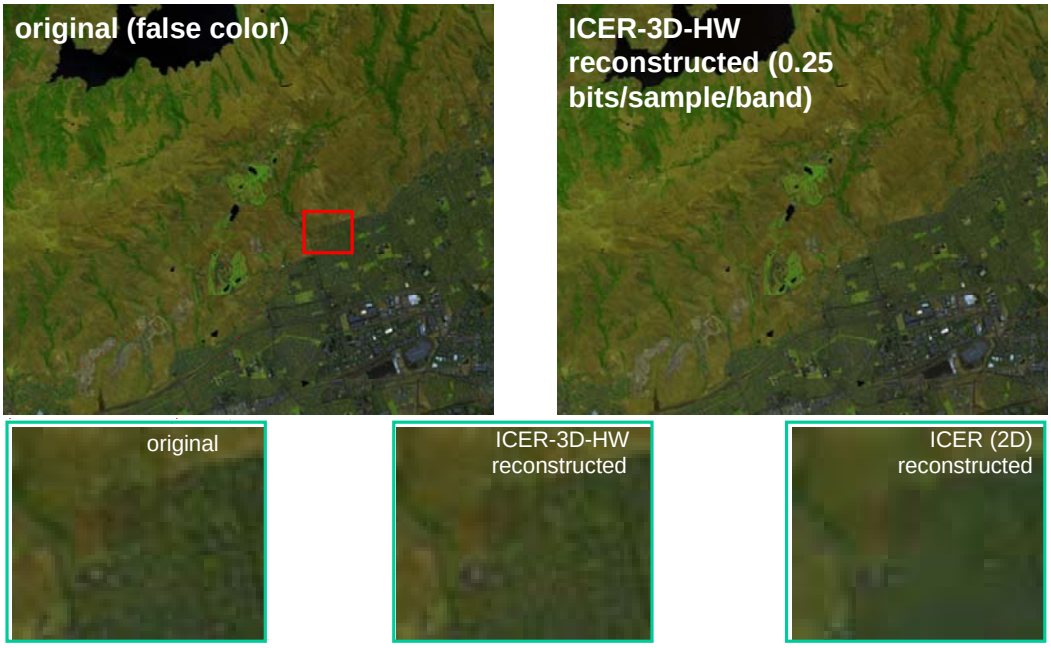


Figure 2.8: Lossy Compression with ICER (2D) and ICER-3D-HW

2.4 Hyperspectral Data Compression on Reconfigurable Platforms

Effective compression techniques tend to be computationally intensive and ICER-3D-HW is no exception. On-board deployment of such algorithms require high speed processors that can provide higher throughput but are power hungry. Dedicated hardware solutions are highly desirable for their high throughput and power efficiency.

Traditional VLSI implementations are power and area efficient, but they lack flexibility for post-launch modifications and repair, are not scalable, and cannot be configured to efficiently match specific mission needs and requirements. An efficient embedded and scalable architecture for the ICER-3D-HW compressor was developed and implemented. The implementation targets state-of-art FPGAs, such as Xilinx Virtex II pro, and can be easily extended to future Virtex generations such as Virtex IV and Virtex V families. These FPGAs provide design options with on-chip hard-wired PPC 405 processor cores [113] to form an embedded platform or a System on a Chip. Efficient partitioning of the algorithm into software and hardware modules was performed to take full advantage of the available reconfigurable hardware resources and the on-chip processors. While many researchers have addressed system implementations with SoC FPGAs, our implementation is unique

in addressing as complex and data intensive a system as is found in hyperspectral data compression.

2.4.1 FPGA Implementation of a Scalable Embedded Hyperspectral Data Compression Architecture

We implemented the algorithm as a hybrid Hardware/Software SoC FPGA. Our SoC methodology addresses the issue of SW/HW partioning of algorithm modules by allocating these functions after performing software profiling to identify appropriate candidates for hardware acceleration. Dynamic range expansion analysis was also performed to identify and select a suitable choice of a DWT filter pair for hardware implementation. Details of the SoC FPGA implementation are provided in Chapter 5.

The Implementation was coded in VHDL and ported to a prototype board targeting the Xilinx Virtex II Pro XC2VP70 chip. The hardware development system is shown in Figure 2.9. The throughput of the system was up to 1 sample/clock cycle when two copies of each of the three main hardware modules were running in parallel to perform lossless compression (lossy compression can run faster since not all bit planes need to be compressed). For the current clock speed of 50 MHz, this throughput is close to 2 orders of magnitude faster than the software code, which has a throughput of 610 Ksamples/sec on a Pentium Centrino 1.6MHz processor.

Device utilization shows that the implementation occupies less than 61% of FPGA resources with 2 copies of each module running in parallel. Power consumption for this implementation is 7.5 Watts. The final hardware implementation block diagram was shown in Figure 1-3.

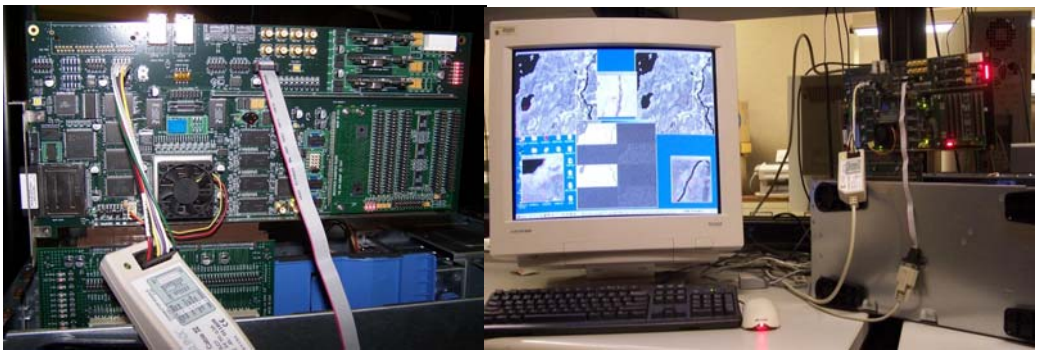


Figure 2.9: FPGA Hardware Development System

2.5 Conclusions

In this chapter we presented an overview of hyperspectral data and a novel low-complexity 3D compression system for hyperspectal images and sounders. Our system is based on the reversible DWT transforms, which is progressive and suitable for space based instruments that require both lossless and lossy data compression. We presented compression results for test images from AVIRIS data sets. We also presented an embedded and scalable implementation for the ICER-3D-HW compression algorithm on an SoC FPGA. The approach uses a co-design platform (SW/HW) with architecture-dependent enhancements to improve throughput, power and device utilization.

Chapter 3

Parallel FPGA Implementations of the 2D Discrete Wavelet Transform

3.1 Introduction

Recently, there has been a tremendous increase in the application of wavelets in many scientific disciplines. Typical applications of wavelets include signal and image processing [3][103][79], numerical analysis [22], biomedicine [92], satellite imagery and data compression [102][52][62]. While the wavelet transform offers a wide variety of useful features, it is computation intensive. Furthermore, in contrast to other transforms, such as Fourier transform or discrete cosine transform, it is not block based, which makes it difficult to implement in a parallel representation. Several VLSI and FPGA architectural solutions for the discrete wavelet transform [48] have been proposed in order to meet the real time requirements in many applications. These solutions include parallel filter architectures, linear array architectures, multigrid architectures [108][31], and 2D block based architectures [61]. Most of these implementations are special purpose parallel processors developed for specific wavelet filters and/or wavelet decomposition trees that implement high level abstraction of the standard pyramid algorithm. In addition,

some are complex designs requiring extensive user control. Knowles [75][76] proposed systolic-array-based architectures without multipliers for the 1-D and 2D DWT, but these architectures are not suitable for all wavelets. Vishwanath *et al* [109] proposed a systolic-parallel architecture for the 2D DWT based on the recursive pyramid algorithm, but due to the approximations involved these architectures cannot be used when exact reconstruction is required. Reza and Turney [86] proposed a sequential implementation of the polyphase representation of the DWT suitable for the Xilinx Virtex FPGAs. Yong-Hong *et al* [115] presented a parallel architecture that can compute low pass and high pass DWT coefficients in the same clock cycle. King-Ch *et al* [70] implemented the operator correlation algorithm of the 2D DWT. However, these FPGA implementations are aimed at specific filterbanks, do not support block-based transform, or do not handle block boundaries efficiently.

There is a clear need for a fast hardware DWT that allows flexibility in customizing the wavelet transform with regard to the filters being used and the structure of the wavelet decomposition. In many image processing applications, including compression, denoising and enhancement, it is critical to compute the 2D wavelet transform in real-time. Field programmable gate arrays (FPGAs) offer a suitable platform (cost effective and highly flexible) for such an implementation.

FPGA-based systems represent a new paradigm in the industry – a shift away from a full custom ASIC solutions for each application, to a single hardware assembly (FPGA) that can be reconfigured to accommodate multiple applications. This approach also provides many advantages over ASIC designs in terms of flexibility of field upgrades, reliability and fault tolerance via reconfiguration to repair and work around failures. In addition, it provides faster and cheaper design cycles. Future commercial and space applications can benefit from this flexibility to enable remote repairability and upgradeability.

In this chapter, we propose a methodology for an FPGA block-based parallel implementation which utilizes overlapping techniques based on the lifting factorization. Our proposed methodology produces architectures that are simple, modular, and cascadable for computation of the 2D data streams. The novelty of our work is that, in addition to improved performance over existing architectures, we provide flexibility to accommodate various DWT transforms; and we demonstrate that in addition to memory size reduction, as inherently provided by the lifting approach, we can also reduce external memory I/O access and hence the communication overhead induced by the parallel computation. Our proposed architectures can be implemented on any generic FPGA with external RAM memory banks, but perform best if implemented on FPGAs with adequate internal RAM

which can be used for on-chip storage of intermediate results, voiding most of the time consumed in external memory access operation.

The chapter is organized as follows. A review of both the lifting factorization for the discrete wavelet transform and the overlapping techniques is presented in section 3.2, along with examples of existing architectures. Our implementation methodology is introduced in section 2.3. An example of an FPGA implementation is provided in section 2.4 followed by the conclusions in section 2.5.

3.2 Lifting Factorization and 2D DWT Architectures for Hardware Implementations

The DWT, as represented by the Mallat style [78] multilevel octave-band decomposition system, which uses a two-channel wavelet filterbank, is very computation intensive. This decomposition can be implemented as a pyramidal recursive filtering operation using the corresponding filter banks as shown in Figure 3.1. We will refer to it as the standard algorithm. The process for the 2D DWT decomposition for each level is implemented with a cascaded combination of two 1-D wavelet transforms. The standard algorithm is constrained by large latency, a high computational cost and the requirement for a large buffer size to store intermediate results, which makes it impractical for real time applications with memory

constraints. An alternative representation, requiring fewer computations, is the lifting algorithm [94], which will be the basis of our implementations in this chapter.

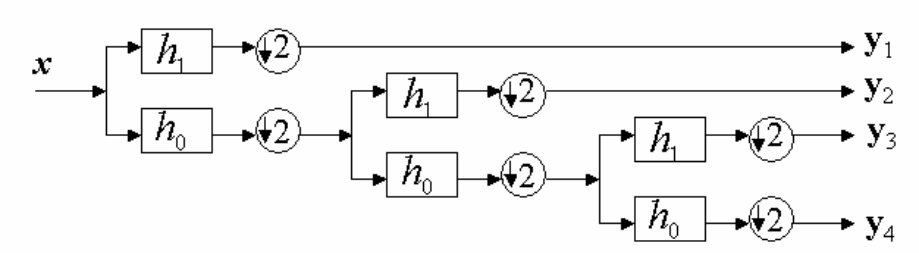


Figure 3.1: 1D Wavelet decomposition showing cascaded levels of filter banks

3.2.1 Lifting Factorization

The basic idea for the lifting algorithm is to model the DWT as a finite state machine (FSM), which progressively updates (or transforms) each raw input sample (initial state), via intermediate states, into a wavelet coefficient (final state). Daubechies and Sweldens [36] have shown that any FIR wavelet filters pair can be represented as a synthesis polyphase matrix, **Ps**(z), which can be factored into a cascaded set of elementary matrices (upper triangular and lower triangular ones) leading to a factorization in the form:

$$P_s(z) = \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \qquad (3.1)$$

for which the corresponding analysis polyphase matrix **Pa**(z) is:

$$P_a(z) = \begin{bmatrix} 1/K & 0 \\ 0 & K \end{bmatrix} \prod_{i=m}^{1} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \qquad (3.2)$$

where $s_i(z)$; $t_i(z)$ are Laurent polynomials and $m \leq L/2$ (L is the filter length) is determined by the specific factorization form. It has been shown that such a lifting-factorization based DWT algorithm is, asymptotically for long filters, twice as fast the standard algorithm [36][6].

From a computational point of view [94], there is no big difference among these elementary matrices, each of which essentially updates the input data samples using linear convolutions, allowing in-place calculations The filtering operation can then be seen as an FSM as shown in the following equation, where each elementary matrix $e^i(z)$ updates the FSM state $X^i(z)$ to the next higher level $X^{i+1}(z)$.

$$Y(z) = P(z)X(z) = e^{2m}(z)\cdots e^{1}(z)\underbrace{\underbrace{e^{0}(z)X^{0}(z)}_{X^{1}(z)}}_{X^{2}(z)} \qquad (3.3)$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{Y(z)=X^{2m}(z)}$$

3.2.2 Existing Architectures for FPGA based parallel implementation of the 2D DWT

Several 2D DWT architectures for parallel implementations were proposed recently, as wavelets gained popularity. Most of these architectures concentrate on saving hardware resources, memory and computations. For example the *1D folded* architecture by Chakrabati *et al* [32] reuses the same logic for both row and column transforms. While it achieves lower hardware resources, it requires high memory

bandwidth. For an *NxN* image, $2N^2$ read and write operations are needed for the 1st level DWT decomposition. The *Partitioned DWT* architectures by Ritter *et al* [87], partitions DWT into small 2D Blocks to achieve lower memory bandwidth and low on-chip storage, but it produces block artifacts along the boundaries between partitions. The *recursive pyramid* algorithm by Vishwanath et. al. [109], takes advantage of different clock rates at different DWT levels to intermix the next level computations with current calculations. It requires a large on-chip memory and complex scheduling for interleaving the DWT levels. The *Generic 2D biorthogonal* DWT by Benkrid *et al* [21], uses separate architectures to calculate each DWT level. It achieves full utilization of memory bandwidth – one write and one read per pixel, but with massive on-chip storage requirements. The *Modified folded* architectures for SPHIT image compression by Fry and Hauck [43], uses the same filter assembly for both rows and columns with pixels read from one memory port, transposed for the column transform, and written to another memory port. It achieves a DWT runtime of $¾\ N^2$ cycles for an *NxN* image, but it assumes 64-bit wide memory ports to allow filtering of 4 rows at a time of 16 bit pixels, which may not be practical for all systems.

3.2.3 Proposed Parallel Architectures for the 2D DWT

The standard DWT algorithm operates on the whole image in a sequential manner. An improved implementation would partition the image into several blocks and

operate on each block independently and in a parallel manner, and then would merge the results to complete the DWT. While this architecture still requires the same intensive computations of the recursive filtering operation and the same memory requirements, the computation can be sped-up if one uses a multi-processor system or identical parallel hardware implementations of the filtering blocks that can operate on multiple image blocks simultaneously. A known disadvantage of such an approach is that it requires data exchanges between neighboring blocks at each decomposition level of the discrete wavelet transform, and hence an additional overhead due to inter-processor communications.

We consider three parallel implementations based on the lifting factorization of the DWT. The standard overlapping algorithm shown in Figure 3.2 eliminates the blocking artifacts and imposes relatively simple control complexity, but has high computational cost and requires high on-chip buffering of data. For an *NxN* image, using DWT filters of length less than or equal to *L*, and partitioned into *S* Blocks, the number of additional filtering operations for a 1 level 2D DWT decomposition vs. a non-overlapped approach is: *2N*L*(S-1).*

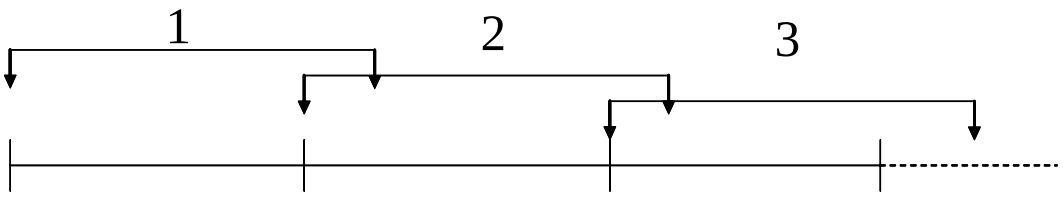


Figure 3.2: Overlapping

The overlap-save algorithm is shown in Figure 3.3. It requires saving the boundary data sets and exchanging them at every level of the DWT decomposition. It achieves lower computational cost but requires higher communication overhead at each level of DWT decomposition.

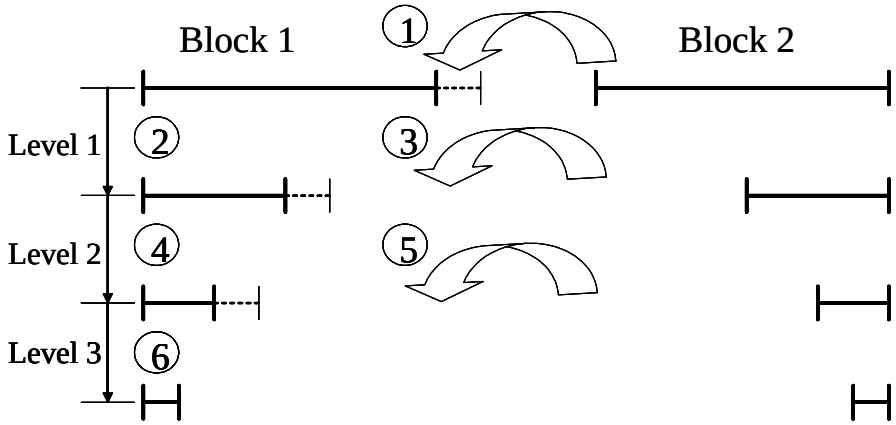


Figure 3.3: Overlap-Save

The overlap-state [53][54][55] algorithm is a block-based parallel implementation that uses the FSM lifting model. Raw input samples are updated progressively as long as there are enough neighboring samples present in the same block as shown in Figure 3.4. Data samples near block boundaries can only be updated to intermediate states due to lack of sufficient neighboring samples. Rather than communicating raw data samples before the start of the decomposition at each level, these partially updated boundary samples, which form the state information, are collected at each level and exchanged at the conclusion of the independent transform of each block. A post processing operation is then initiated to complete the transform for boundary samples. Using this technique, the DWT can be computed correctly, thus eliminating

blocking effects, while the inter-block communication overhead is significantly reduced.

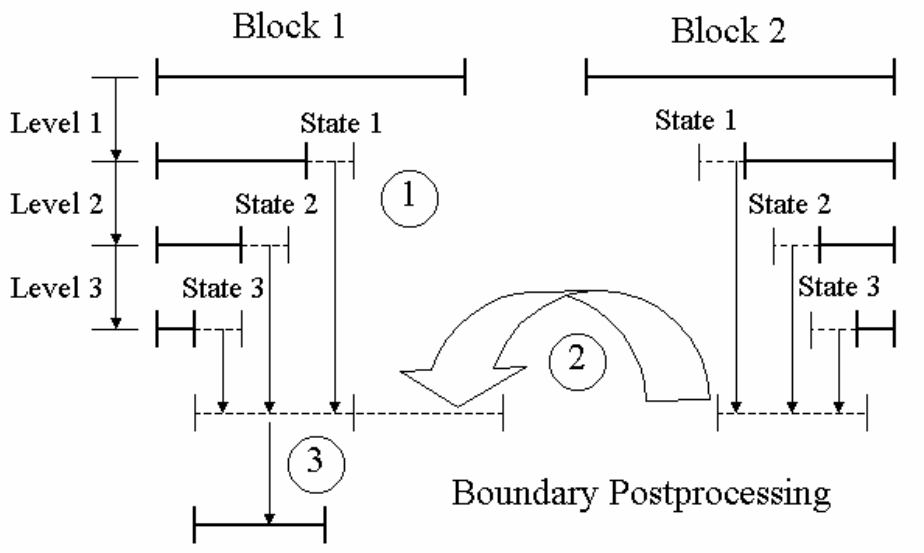


Figure 3.4: Overlap-State technique.

3.3 FPGA Implementation Methodology

Our methodology is depicted in the flow chart shown in Figure 3.5. While quite a few steps in the methodology can be considered generic in terms of the hardware design process, there are a few highlighted steps that require non-standard considerations involving design choices required to meet practical constraints. We will address each of the highlighted steps in the following sections.

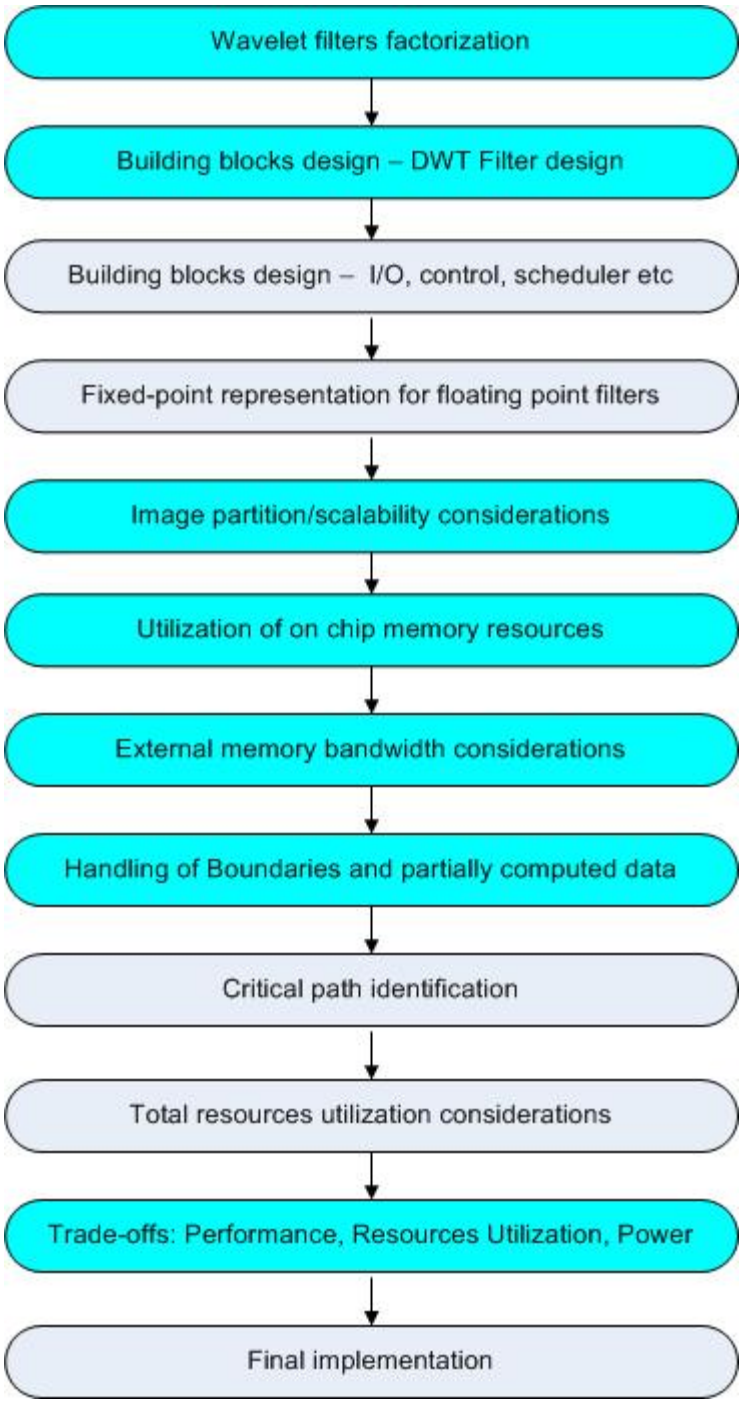


Figure 3.5: Implementation Methodology Flow Chart

3.3.1 Lifting Factorization Considerations

The lifting factorization of equations (3.1) and (3.2) is not unique, but can be optimized to provide architectures that are efficient for hardware implementation by maximizing the shift and addition operation and minimizing multiplications. The lifting factorization can then be expressed as follows:

$$P(z) = \prod_{i=N}^{M} \begin{bmatrix} 1 & s_i(2^j z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(2^j z) & 1 \end{bmatrix} \prod_{n=1}^{N-1} \begin{bmatrix} 1 & s_n(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_n(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \quad (3.4)$$

Where j is an integer and $N < M$.

In addition, in 2D implementations, the multiplications by constants K and $1/K$ in the last matrix can be combined after both row and column filtering are complete into operations of K^2 and $1/K^2$

3.3.2 2D DWT Filter Design

The filter design methodology has to decide among different well known architecture choices, such as folded architecture and cascaded architectures reviewed in section 3.2.2 [32][21][43]

For the folded architecture, for example, the external memory bandwidth is high but it does not require significant hardware resources or on-chip storage. For an image of

NxN pixels and DWT filters of maximum length of F_l, and a 1 level DWT decomposition, it requires memory bandwidth of $4N^2$ (writes and reads).

For the cascaded row-column pipelined architecture, when a desired number of rows are completed (typically, equal to the maximum length of the DWT filters), the column operation can begin. This requires additional internal storage for $F_l N$ pixels. For a 1 level DWT decomposition, the total memory bandwidth required is approximately $2N^2 + ½ F_l N$ (internal memory access cost is typically less than ½ that of the external access). This bandwidth is much less than the $4N^2$ required for the folded case since the maximum filter length, F_l, is small relative to image width N (typically F_l is less than 13 compared to 512 or more for N).

In most design cases, the performance bottleneck is the memory I/O access (i.e. bandwidth). To maximize performance of the implementation in 2D processing context, the preferred choice will be the cascaded architecture.

3.3.3 Image Partioniong and Design Scalability

Block-based implementations may lead to complex boundary post processing, and complex house keeping. Figure 3.6 shows options to partition the images into blocks or stripes.

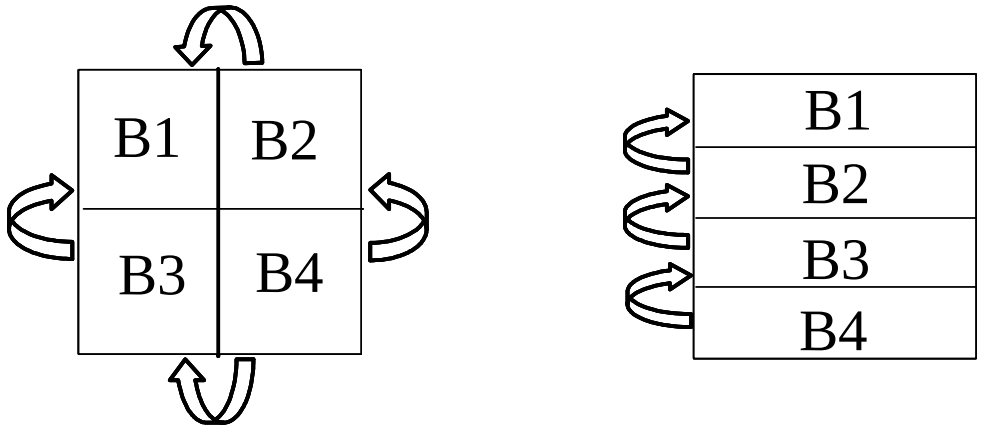


Figure 3.6: Options for 2D image partioning

For block-based parallel partioning, the boundary post processing latency for 1 level DWT latency is $F_l N$ filtering operations, which leads to complex control, as the exchange of boundary data is needed along both adjacent vertical and horizontal boundaries [56].

For stripe-based parallel partioning, the boundary post processing latency for a 1 level DWT is $\frac{1}{2} F_l N$ filtering operations. Additionally, this approach leads to simpler control, as management of boundary data exchanges is required only along the adjacent vertical boundaries.

Since speed and lower design complexity is our goal, we choose the stripe parallel implementation. Available on-chip memory storage and external memory bandwidth determine degree of design scalability (i.e. the number or stripes and number of DWT processing elements).

3.3.4 Memory Bandwidth Considerations and Storage Calculations

For a DWT filter pair and an image of $N \, x \, N$ pixels, denoting:

F_l - length of the longest filter

J - DWT decomposition levels

S - Number of DWT line processors (blocks/stripes)

Consider the stripe-parallel design shown in Figure 3.6. After the completion of 1 level DWT decomposition, the number of transitional boundary states generated at the first boundary of B1 and B2 is:

from B1:

$$m_{B1} = \left\lceil \tfrac{1}{2} F_l \right\rceil * N \qquad (3.5)$$

and from B2

$$m_{B2} = \left\lfloor \tfrac{1}{2} F_l \right\rfloor * N \qquad (3.6)$$

where memory is measured here in number of pixels, $\lceil \ \rceil$ and $\lfloor \ \rfloor$ are the ceiling and the floor operators to accommodate odd length DWT filters at the stripe boundaries. This results from the absence of image data along the boundaries of B1 and B2 required to complete the filtering operations. After the completion of 2 decomposition levels additional transitional boundary states are generated at the same boundary:

from B1:

$$m_{B1} = \left\lceil \tfrac{1}{2} F_l \right\rceil * N * \tfrac{1}{2} \qquad (3.7)$$

and from B2

$$m_{B2} = \left\lfloor \tfrac{1}{2} F_l \right\rfloor * N * \tfrac{1}{2} \qquad (3.8)$$

Hence the memory required to hold transitional boundary states for each boundary is:

$$m_1 = F_l \sum_{i=0}^{J-1} N * \left(\frac{1}{2}\right)^i \qquad (3.9)$$

and the total needed memory for all the boundary data is:

$$m_{total} = F_l \left[\sum_{i=0}^{J-1} N * \left(\frac{1}{2}\right)^i \right] * (S-1) \qquad (3.10)$$

To minimize external memory I/O bandwidth, the decision was made earlier to use the cascaded architecture, i.e., pipeline row and column filtering. Assuming a FIFO buffer length of N (image width), the memory required for row buffering is:

for one block

$$m_{FIFO} = F_l * N \qquad (3.11)$$

and total needed memory for FIFO buffers is

$$m_{FIFO-total} = F_l * N * S \qquad (3.12)$$

For example: for an image of 512x512 pixels, a 3 level (9,7) DWT decomposition with a partition size of 4 stripes, the total required on-chip RAM (BRAM) measured in pixels is: 9*(512+256+128)*4 + 9*512*4 = 42K bytes

Actual required BRAM may need to be 84K bytes to account for dynamic expansion in DWT domain.

3.3.5 Management of Memory and Boundary Data

The main remaining issue, once the previous design choices are made, is to design an efficient memory management scheme to avoid excessive post boundary processing I/O cost.

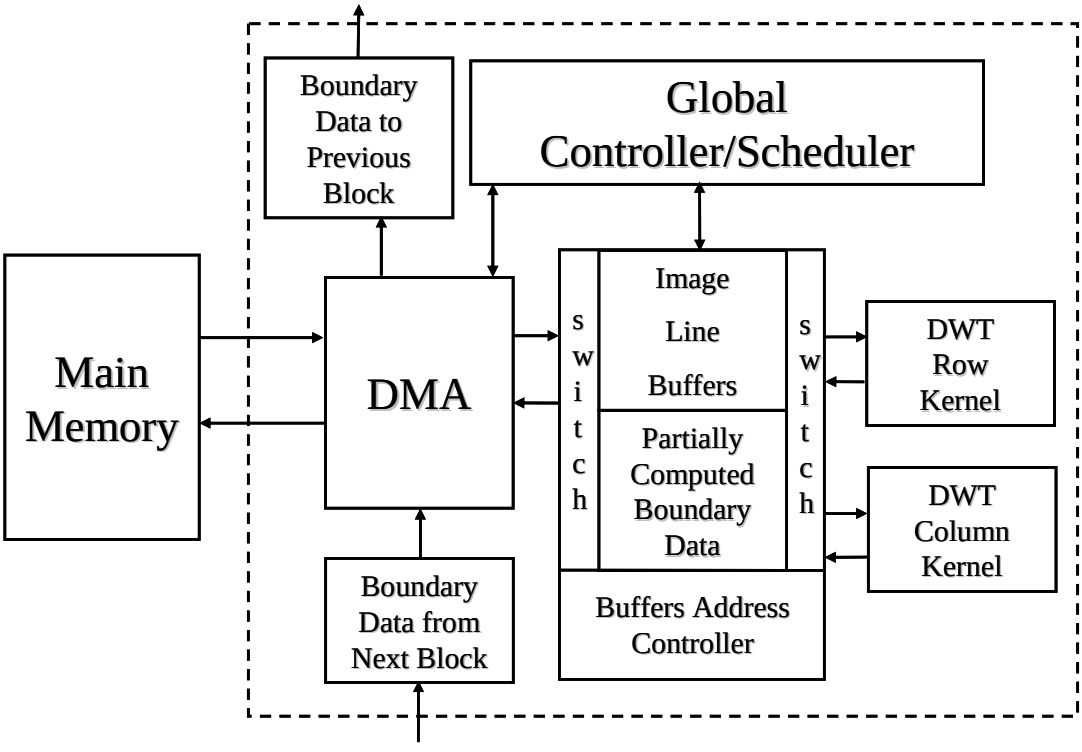


Figure 3.7: DWT Unit Block Diagram

A data transfer method that utilizes an efficient DMA was designed and implemented. Figure 3.7 shows a block diagram of one 2D DWT module. The module consists of a shared global controller, DWT row and column kernels, line and boundary data buffers and a DMA. We designed a custom DMA engine to handle data transfer from main external memory, from partially computed boundary data buffers and from neighboring processing units. It contains the following parameters for 1D linear addressing: starting address, intra row/column count, intra row/column step, intra row/column jump, pixel count, boundary data starting address and boundary data jump address for data from adjacent processing unit. This DMA design allows interleaving the DWT computations between adjacent processing elements, allowing the process to be completed in a seamless manner to the DWT kernels and hence minimizing the overhead usually associated with parallel implementations. Look-up tables are used to pre-store specific design parameters for a DWT implementation. These addresses are passed to the DMA by the global controller, which is shared by all DWT processing units.

The computation of the DWT is conducted in the following manner. The original image data is stored in main memory (off-chip) and loaded to the line buffers line by line through the DMA. Row transformed lines are completed and stored in-place in the line buffers. The DWT column kernel operates on the row transformed data once an adequate number of rows are completed (this depends on the lengths of the DWT

filters). All operations are completed in a pipelined manner. Completed DWT data is written back to the main memory through the DMA. Transitional (partially computed) boundary states are stored in boundary data buffers (on-chip) and fetched later to augment image data lines and/or boundary data from the neighboring stripe and then passed to the DWT kernels. Once all the DWT decomposition levels are computed, each DWT processing unit passes its upper transitional boundary data to the next top neighboring unit and receives lower transitional boundary data from the next lower unit to start the merge process to complete the DWT decomposition at the stripes boundaries. All processing units complete, in parallel, all the partially computed DWT coefficients (except for the last unit) starting with the column transformation of the first level decomposition and moving on the subsequent DWT decomposition levels. At each stage, transitional data (or states) is passed to the appropriate locations in the DWT kernels pipeline, and computations proceed in the same manner that was used for the initial DWT computations until all the DWT levels are completed.. Figure 3.8 below shows the pipeline and flow of boundary data (states) for a two adjacent DWT processing units. Boundary states as shown are from unit B2 are merged with those from unit B1 at B1 to complete the DWT computations.

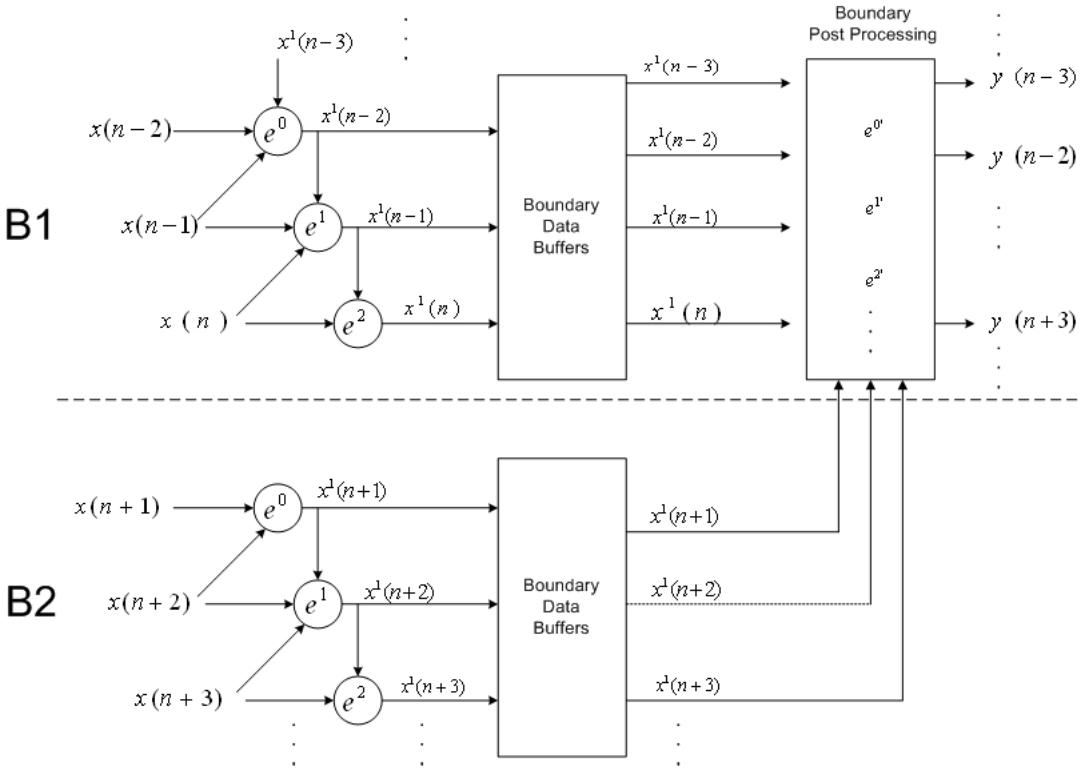


Figure 3.8: Pipeline and data flow for boundary states

3.3.6 Resource Utilization and Architectural Trade-offs

The final step of the methodology is to complete the trade-off analysis to select the appropriate architectural design approach given the practical constraints. For example, considering the three overlapping architectures introduced in section 3.2.3, and a platform that allows a separate external memory bank for each partition stripe. For highest throughput performance, the best choice is the overlap-state architecture, which achieves a performance of $\frac{1}{2}N^2$ (in run time cycles) and requires the following internal memory for data buffering and transitional data storage:

$$F_l * N * S \quad + \quad F_l * \left[\sum_{i=0}^{J-1} N \left(\frac{1}{2} \right)^i \right] * (S-1). \qquad (3.13)$$

where F_l, N, S and J are filter length, image width, number of partition stripes and number of DWT levels respectively. Memory is measured here in pixels.

This scheme also requires a relatively low external memory bandwidth of:

$$\frac{2}{S} * \left[\sum_{i=0}^{J-1} N^2 \left(\frac{1}{2} \right)^{2i} \right] \qquad (3.14)$$

which is effectively one read and one write operation per pixel for each 2D DWT computation performed in a stripe.

For systems where the overlap choice is not possible due to low internal memory availability, the overlap-save architecture is the next best choice. It achieves a performance of $(½N^2+½NSF_l)$, requires total storage for buffering input and boundary data of:

$$F_l * N * S \quad + \quad F_l * N * (S-1) \qquad (3.15)$$

and requires the same low external memory bandwidth of equation 3.14

Finally, for systems with very low on-chip storage that must deal with large images (i.e. 1024x1024 or larger), the conventional overlapping architecture is the best choice. It achieves a performance of $(½N^2+NSF_l)$ and requires a relatively low storage for buffering input of:

$$F_l * N * S \qquad (3.16)$$

This scheme, however, requires a relatively high external memory bandwidth of:

$$\frac{2}{S} * \left[\sum_{i=0}^{J-1} N^2 \left(\frac{1}{2} \right)^{2i} \right] + F_l * \left[\sum_{i=0}^{J-1} N \left(\frac{1}{2} \right)^{i} \right] * (S-1) \qquad (3.17)$$

Table 3.1 shows the trade-offs under consideration among the three parallel implementations, overlapping, overlap-save and overlap-state. Table 3.2 compares the overlap-state implementation to the 5 architectures we reviewed in section 3.2.2. As can be seen, the 1D folded architecture requires no internal memory storage, but performs the worst at $2N^2$ and requires large memory bandwidth. The modified folded architecture takes advantage of a wide memory bus to achieve low effective memory bandwidth and a performance of $3/4N^2$. Our overlap-state architecture outperforms all the architectures with performance of $\frac{1}{2}N^2$ and a low effective memory bandwidth (about ½ of that for the 1D folded), given the platform assumptions and storage considerations.

Table 3.1: FPGA Parallel DWT Implementations Trade-offs (for NxN image)

Architecture	FPGA 2D Overlapping	FPGA 2D Overlap-Save	FPGA 2D Overlap-State
Overlapping Rows	½ Longest Filter Length	None	None
External Memory I/O BW	$\frac{2}{S}\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i} + F_l\left[\sum_{i=0}^{J-1} N\left(\frac{1}{2}\right)^{i}\right](S-1)$	$\frac{2}{S}\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$	$\frac{2}{S}\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$
Internal memory required	Line FIFOs $F_l N S$ Pixels	$F_l NS +$ $F_l N(S-1)$ Pixels	$F_l NS +$ $F_l\left[\sum_{i=0}^{J-1} N\left(\frac{1}{2}\right)^{i}\right](S-1)$ Pixels
Additional Hardware Resources	None	Additional scratch pad memory Plus control (~4% resources)	Additional Block RAM (see above) Plus special DMAs (~10% resources)
Performance	Slowest ½ (N^2) + (NSF_l)	Faster (10%) ½ (N^2) + ½ (NSF_l)	Fastest (15%) ½ (N^2)

Table 3.2: Comparisons to other DWT FPGA Architectures (for NxN image)

Architecture	1D Folded	Partioned DWT	Generic 2D	Recursive	Modified Folded	Overlap-State
HW Logic Resources	Low	Moderate	High	High	High	High
External Memory Bandwidth	$4\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$	$2\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$	2 N^2	$2\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$	$\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$	$\frac{2}{S}\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$
Internal memory required	None	$F_l N$	$\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$	$\sum_{i=0}^{J-1} N^2\left(\frac{1}{2}\right)^{2i}$	None	$F_l NS$ + $F_l\left[\sum_{i=0}^{J-1} N\left(\frac{1}{2}\right)^{i}\right]*(S-1)$
Performance (run time cycles)	$2N^2$	3/2 N^2	3/2N^2	N^2	¾ N^2	1/2 N^2
Additional Complexity	None	None	None	Intermix of DWT Levels calculations	Transpose row DWT coefficients	Boundary post processing (complex DMAs)

3.3.7 Applying the Methodology under Different Platform Constraints

In the preceding sections, several assumptions were made regarding platform constraints and available memory resources. These assumptions led us to the overlapping cascaded architectures. We now consider whether our methodology would still be applicable if we had a different platform and different constraints.

Consider an example where we have low on-chip memory but high external memory bandwidth such as would be the case for accesses of 64 bits or four 16 bit pixels. The folded architecture in this case will be the desired choice due to limited on chip resources to store row or column transformed data. The high memory bandwidth compensates for the large number of external I/O operations. When making the choice for the image partioning, neither the block based nor the stripe based partioning takes full advantage of the high memory bandwidth available. A four-line based approach would utilize the 64 bit bus efficiently. However, to take advantage again of this 4-line based architecture, the row transformed DWT data needs to be transposed prior to writing it back to memory to be readily available for the column transform operations. This requires an additional hardware module to perform the transpose operations. The tree shown in Figure 3.9 identifies the most efficient design choices at each stage given the platform constraints. The resultant architecture in this case is very similar to the modified folded one chosen by Fry and Hauck [43]

for the DWT implementation of the SPIHT image compression mentioned earlier in section 3.2.2.

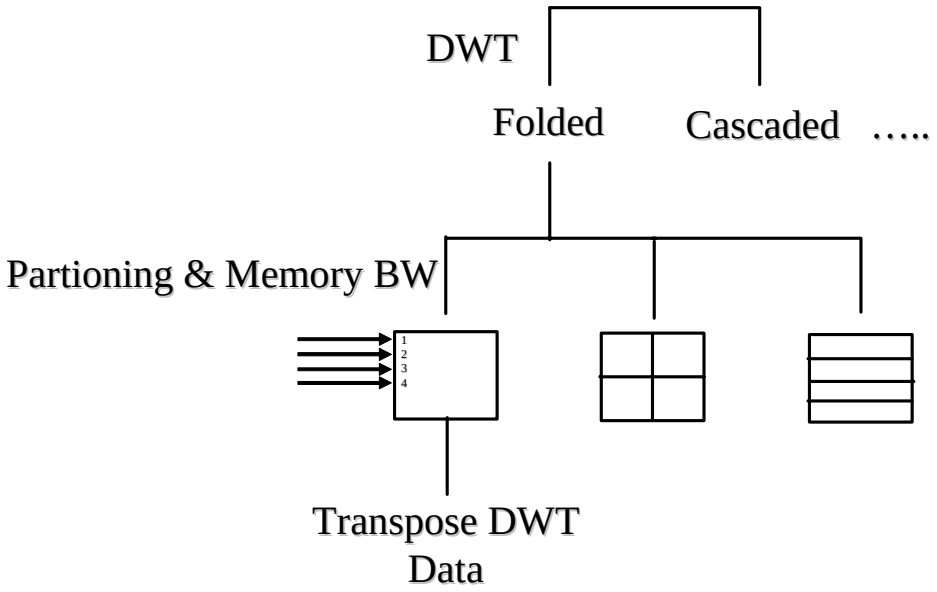


Figure 3.9: Design tree for an example with different platform constraints

3.4 FPGA Implementation Example - (9,7) DWT

The three overlapping algorithms presented in the previous section were implemented for the popular floating point Daubechies (9,7) DWT. While all these parallel architectures can be generalized to any wavelet filters pair, we selected the (9,7) filter pair for our initial evaluation due to its popularity and its use for lossy compression in the JPEG2000 standard [57].

The lifting factorization of the (9,7) filters yields [2]:

$$\begin{aligned} d_1[n] &= d_o[n] - \alpha_0 (s_0[n+1] + s_0[n]) \\ s_1[n] &= s_o[n] - \alpha_1 (d_1[n] + d_1[n-1]) \\ d_2[n] &= d_1[n] + \alpha_2 (s_1[n+1] + s_1[n]) \\ s_2[n] &= s_1[n] + \alpha_3 (d_2[n] + d_2[n-1]) \\ s[n] &= \beta_0 s_2[n] \\ d[n] &= \beta_1 d_2[n] \end{aligned} \qquad (3.18)$$

where

$$\alpha_0 \approx 1.586134, \alpha_1 \approx 0.052980, \alpha_2 \approx 0.882911,$$
$$\alpha_3 \approx 0.443506, \beta_0 \approx 0.812893, \beta_1 \approx 1/\beta_0.$$

A stripe-partitioning scheme was used to allocate input data sequence uniformly onto different processing units. Since, in this method, no segmentation is done in the row direction, the data to be exchanged, and the state information, will only appear along the vertical boundaries of each block.

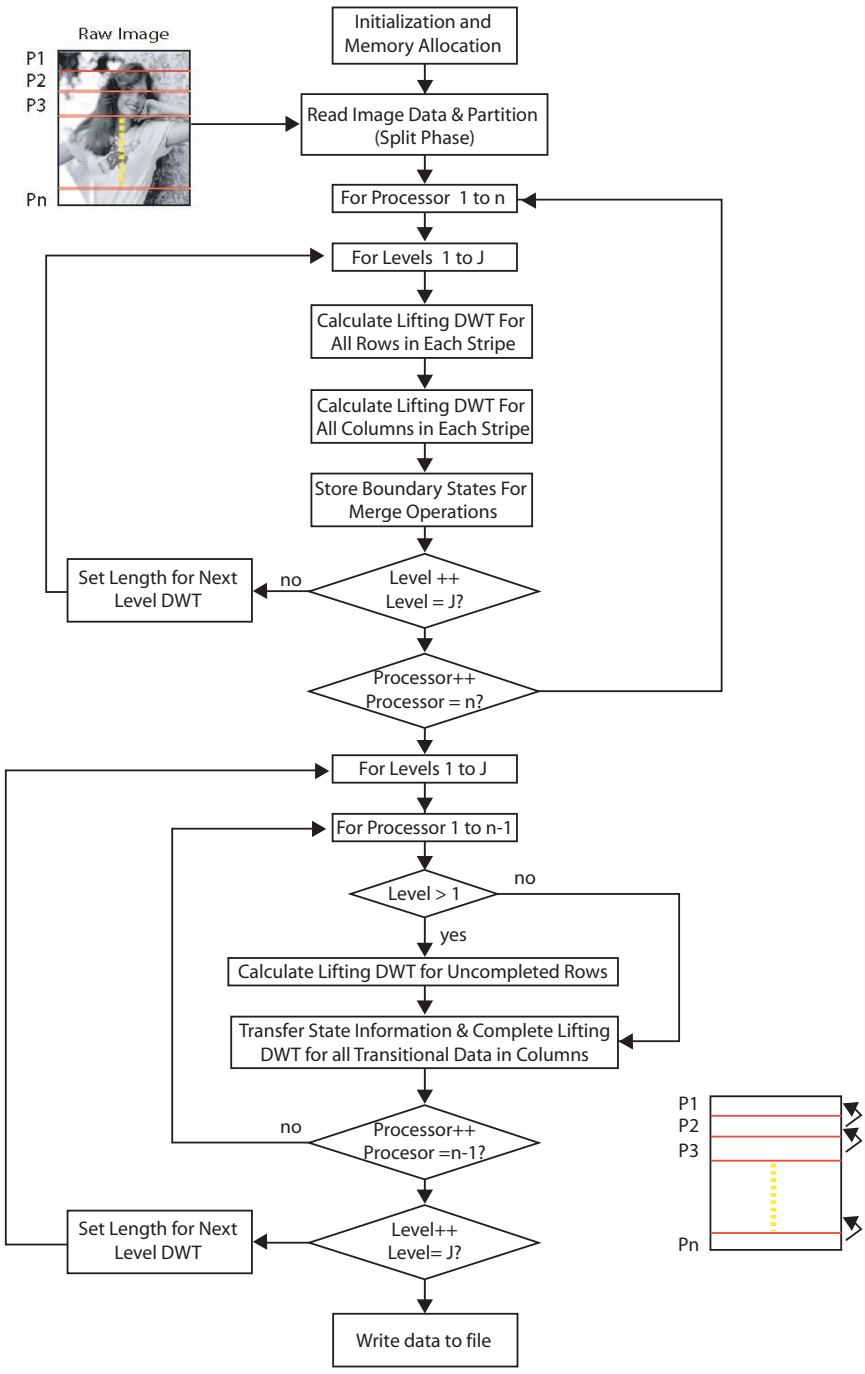


Figure 3.10: Flow chart for the Overlap-State DWT algorithm

The flow chart shown in Figure 3.10 details the implementation for the parallel overlap-state algorithm, since it is the most complex of the three parallel algorithms. The raw image is partitioned into n uniform stripes and allocated to n processor units. Each unit P_i computes its own allocated data up to the required wavelet decomposition level; we will refer to this stage as the split stage. During the first decomposition level, row transformations are completed first for all lines in a data stripe and stored back into their respective memory buffer locations. Columns are transformed in a similar manner. Once the column transformation is completed for a stripe, data along the stripe boundaries are now in transitional states. The state information for all the stripes other than the first stripe P_1, is stored in the allocated locations for each stripe. The state information will be communicated, during the merge operation, from a stripe to its upper neighbor (i.e., unit P_i sends its data to unit P_{i-1}). At this stage, we proceed to the next decomposition level in each stripe. The procedure is repeated until all decomposition levels are completed for all the units.

The output from this stage consists of two parts: (1) completely transformed coefficients and (2) the state information (partially updated boundary samples). During the next stage, the merge stage, a one-way communication is initiated, wherein the state information is transferred from each processing unit to its top neighbor. For each stripe, the state information from the neighboring block is then combined together with its own corresponding state information to complete the

DWT transform. The first step is to combine state information from unit P_i with that from unit P_{i-1}. Partial DWT computations are performed to complete level 1 decomposition (columns only) and the results are stored in their relative locations in P_{i-1} buffer to be ready for level 2 row transformation (note that level 2 row transformation needs completed data from level 1 decomposition). At level 2, we start with the uncompleted rows from the split stage that were awaiting completion of level 1 decomposition for the columns. Data is combined now from (i) level 2 state information from unit P_i, (ii) row transformed data just completed for level 2 from unit P_i, (iii) state information from unit P_{i-1}. As in level 1, partial DWT computations are performed to complete level 2, and the results are stored in their relative memory locations buffer in P_{i-1}. to be ready for level 3 row transformation. The procedure is repeated until all DWT levels are merged and completed.

A graphical description of the (9,7) overlap-state 2D DWT decomposition and its memory buffer management are illustrated in the example shown in Figure 3.11 In this example an image of 512x512 is split into 4 equal stripes of 512x128 each. After the merge operation for one level DWT decomposition, the transformed data appears distributed in stripes of length 132, 128, 128, and 124 for P_1, P_2, P_3, and P_4 respectively. The P_1 unit receives state information data from P_2. The DWT transformation is completed and stored in P1 buffer extending its contributing length to the transformed image to 132 pixels. Stripe P_2 and P_3 send and receive data from

their neighboring stripes which results in a net of unchanged contributing length. However, P_4 sends its state data to P_3 resulting in a smaller contributing length of 124 pixels. Similarly, a 2 level DWT yields the results shown in Figure 3.11 bottom left. The contribution to the computation of the transformed image from each stripe/processor is shown in Figure 3.11 bottom right.

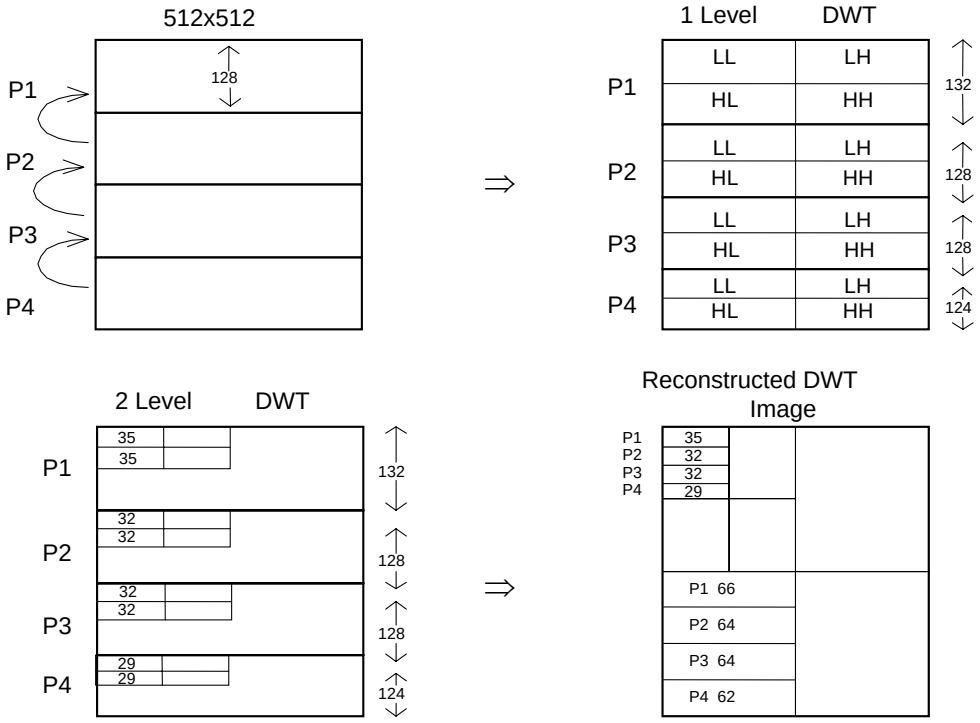


Figure 3.11: Overlap-state implementation of the (9,7) DWT and memory management

Our software simulations were evaluated with images from the Signal and Image Processing Institute image database at the University of Southern California. Results presented in this section were collected using the *Vegas* image of size 512x512 shown in Figure 3.12. Results can be also be extended to color images. The results of

a 2 level DWT decomposition completed by each processing unit and the final 3 level DWT image are also shown in Figures 3.13 and 3.14 respectively.

Figure 3.12: "Vegas" – Original 512x512 image

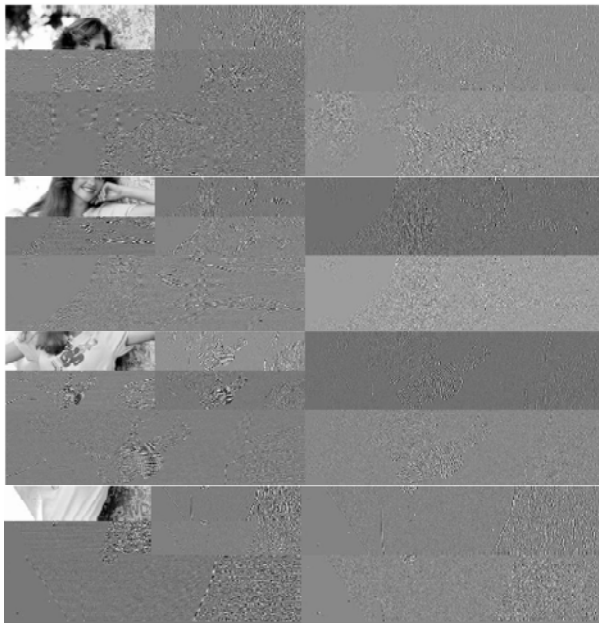


Figure 3.13: Two-level DWT transformed stripes for "Vegas" image from the 4 processing units

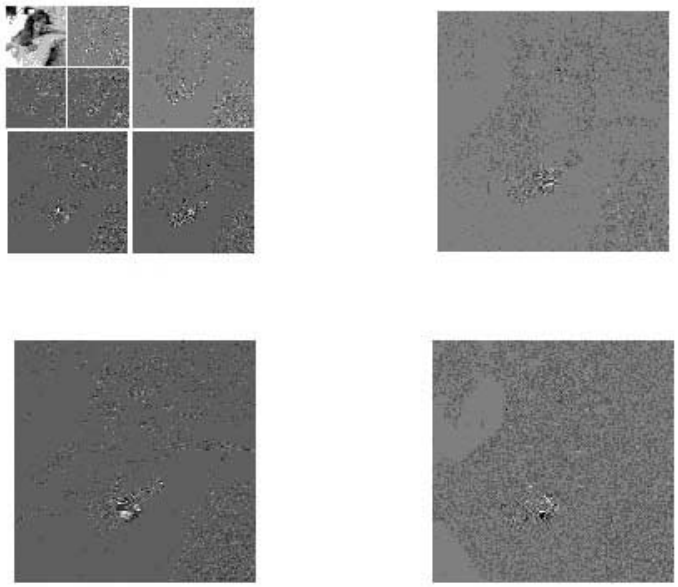


Figure 3.14: Final Subbands for 3-level DWT transformed "Vegas" image

We evaluated the performance for the three parallel implementations in terms of memory I/O bandwidth and total number of multiplies and accumulates (MACs). Our results show that the overlap-state and overlap-save has a much lower memory bandwidth than the overlapping algorithm. Also, as shown in Figure 3.15 the overlap-state algorithm provides significantly better performance when compared to the other two implementations. We also studied the effects of scaling of the overlap-state algorithm. Partitions of 2, 4, 8 and 16 blocks were simulated for the 512x512 test image. It can be seen from Figure 3.16 that performance scales well as the number of partitions increases.

In conclusion, our simulations demonstrate that the overlap-state technique has moderate on-chip storage requirements, has better performance, minimizes the inter-block communications and hence memory I/O operations, and is fully scalable.

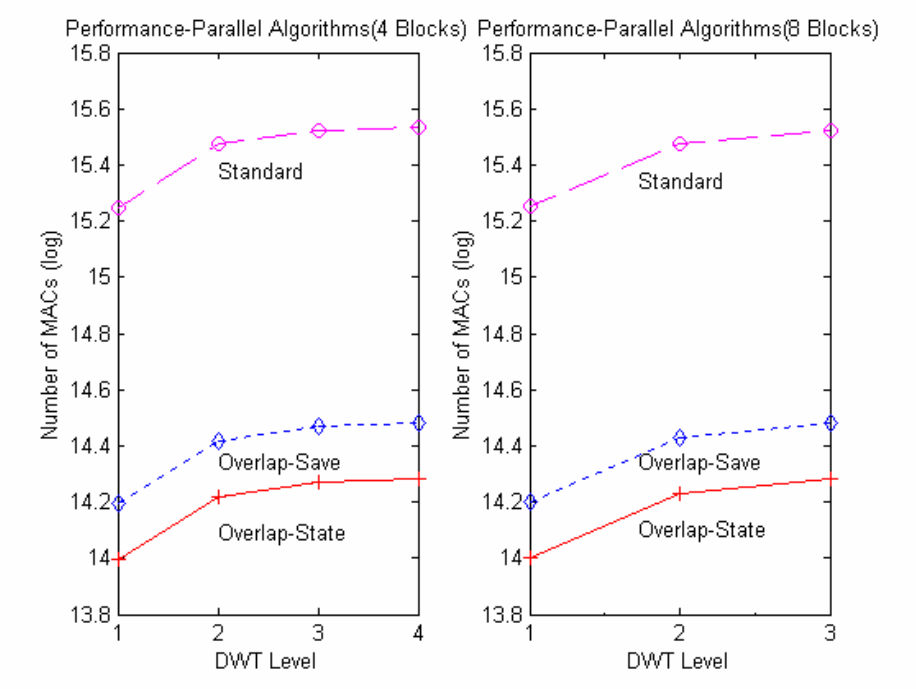


Figure 3.15: DWT – Parallel Implementations – Performance

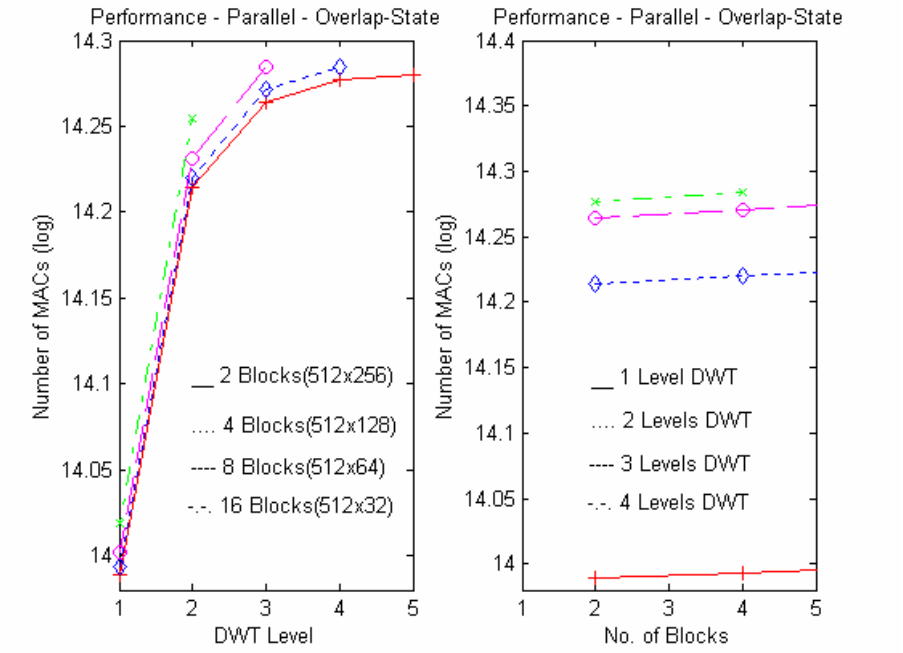


Figure 3.16: DWT – Parallel implementations – Partitioning effects

3.4.1 Parallel 2D DWT System

This section describes the architecture of a parallel 2D-DWT system designed for low-power, real-time image encoding and decoding [7]. Based on a highly parallel, SIMD (Single-Instruction/Multi-Data)-like architecture, the parallel 2D DWT system incorporates multiple processors operating in parallel to achieve high processing throughput. The parallel 2D DWT architecture exploits the unique properties of the overlapping algorithms, especially the overlap-state, enabling a highly memory-efficient and scalable design, and is particularly well suited for FPGA implementation. The parallel 2D DWT design supports dynamic in-situ system reconfiguration for efficient performance under various operating parameters and hardware resources. .

3.4.2 System Architecture

The parallel 2D DWT system comprises a master processor (or global controller) and an array of slave coprocessors operating in a SIMD-like configuration, as shown in Figure 3.17. The master processor can be a hard or soft core micro processor or a custom designed hardware unit and it will be referred to as the global controller (GB). The coprocessors will be referred to as DWT line processors (DLPs). The number of coprocessors can be scaled depending on system performance requirements and available hardware resources.

The GB manages the overall operation of the 2D DWT system. The GB primary tasks include executing the top-level control and processing functions of the overlapping algorithms, as well as scheduling, supervising, and monitoring the processing activities of the DLP coprocessors and managing internal data transfer between the DLPs and external data transfer between the 2D DWT system and external memory and an external host. The GB also provides a host bus interface to an external host processor and I/O devices attached to the host bus. The GB initiates and supervises all DLPs processing activities by dispatching commands to the DLPs. The DLPs are special-function processing units optimized to perform high-speed computation of the 1D-DWT. The DLPs can perform their processing in parallel, independently of other DLPs. The 2D DWT system can operate in either a synchronous or non-synchronous mode. In the synchronous mode, the GB instructs the DLPs to perform identical processing tasks in locked step. In the non-synchronous mode, the DLPs processing is staggered. As instructed by the GB, a DLP begins processing as soon as data is received. At the same time, the GB continues with downloading of data to other DLPs. The non-synchronous mode provides a higher level of concurrency, but requires more complex scheduling and control logic to ensure processing and data coherency.

The GB communicates with the DLPs via a high-speed system bus and initiates all system bus activities, which include reading from and writing to the DLPs local

registers and memories. The system bus consists of a data bus, an address bus, and a set of control signals. The data bus is a bi-directional bus which can be driven by the GB or DLPs. The address bus is only driven by the GB to address DLPs registers or local memories during a read or write operation. The DLPs appear on the system bus as "memory-mapped" devices. Hence, each DLP is assigned a unique, fixed and equal size address space. In addition, a global address space is also defined to globally address the DLPs. Commands or write data addressed to a location in the global address space are written into all corresponding locations of the DLPs.

The GB also performs various top-level processing tasks including: (1) system/global data initialization, (2) image boundary handling, (3) boundary data initialization control, (4) row and column transform control, (5) boundary data transfer control, and (5) decomposition level control. The GB issues commands to the DLPs to perform various low-level DWT processing tasks including (1) data line extension, (2) boundary data initialization, and (3) row/column 1D-DWT.

The GB's basic processing flow is as follows: The GB partitions the input data stream from external I/O into data blocks and commands DMAs in each DLP for image line access and performing partial buffering of the input data if necessary. In the non-synchronous mode, once a complete block has been downloaded to a DLPs local memory, the GB instructs the DLP to perform the 2D DWT on interior block

data. The GB then continues to download the next data block to another DLP for processing, and so on. In the synchronous mode, the GB holds off DLP processing until data blocks are downloaded to all DLPs. When the 2D DWT on the interior block data is completed, the GB instructs the DLPs to merge the boundary data (by utilizing the DLPs DMAs) to complete the 2D DWT of the entire data blocks. Finally, the GB reads and outputs the transformed data from a DLP local block memory to the host processor. The GB communicates with a host processor via the host interface. The host interface supports high speed I/O data transfer to host processor through the host processor bus. It also supports DMA data transfer to external I/O devices attached to the host bus.

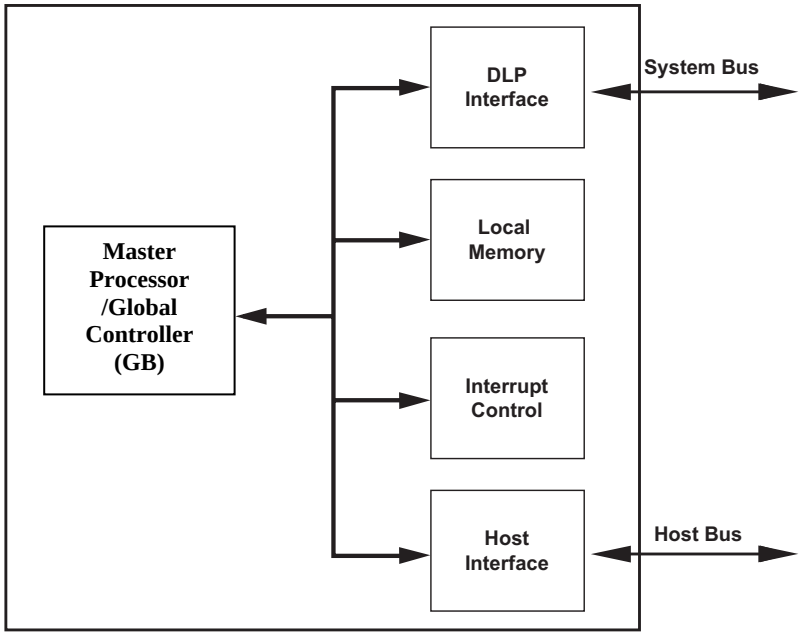


Figure 3.17: Master processor with system and host communication busses.

The GB communicates with the DLPs via the DLP interface unit. The local memory stores the parallel control and sequencing firmware, system parameters, user application configuration data, and provides partial buffering of external I/O data. The GB issues commands to DLPs to initiate DLP processing by writing to the DLP command registers. The commands include (1) reads and writes to the DLP local memories and registers, (2) initiation of 1D-DWT processing, and (3) DLP reset.

3.4.3 DWT Line Processor Architecture

The DWT line processor (DLP) is a special-function coprocessor designed for high-speed computation of the 2D DWT. The DLP performs 2D DWT on a data block as a sequence of row and column 1D DWTs. The row/column 1D DWT is performed in three basic processing steps: (1) boundary extension, (2) boundary initialization, and (3) DWT filtering with lifting.

The DLP comprises a controller, a set of local registers, a local memory, a DMA unit, a pipelined arithmetic unit (PAU), and a GB interface. The DLP controller performs various control and sequencing operations. Based on a multiple state-machine design, the controller is optimized for high processing concurrency and low latency. It decodes GB commands from the GB interface and generates the required sequence of control signals to perform the various DWT processing functions. The DLPs local registers include control, status, and data registers which can be read or written by the GB. The local memory mainly stores the input data block. After a

1D-DWT is performed, the input data block is replaced with the output processed data. The memory also provides a line buffer for the 1D DWT processing. The DMA unit facilitates the transfer of data with the GB and with neighboring DLPs during boundary data merge operation. The DMA unit provides separate input and output data ports for two neighboring DLPs. The ports contain internal buffers to allow parallel data transfer between DLPs.

The DLP communicates with the GB over the system bus via the GB interface unit and appears as a "memory-mapped" device on the bus. The GB interface latches and buffers the address, data, and control signals on system bus during an active bus cycle. The address is decoded to determine if the current bus cycle is a local memory or register access. The SMP interface generates all the required handshake signaling as well as requests to perform the 2D-DWT boundary merge when operating in pipelined mode.

The PAU is a fixed-point arithmetic accelerator designed to perform the numeric intensive 1D-DWT filtering operation using the in-place lifting technique. The PAU is based on a pipeline design and incorporates a set of multiplier-accumulator units (MACs) and data shift registers. The PAU pipeline cycle consists of loading two input data samples and reading out two output samples. The PAU operation starts by shifting in two input samples into the input registers of the first MAC unit in two

clock cycles. However, data in the input registers of the remaining MACs are shifted in the first clock cycle. Additional clock cycles are used to perform the MAC operation. Two output samples from the last MAC unit are then read out to complete the pipeline cycle

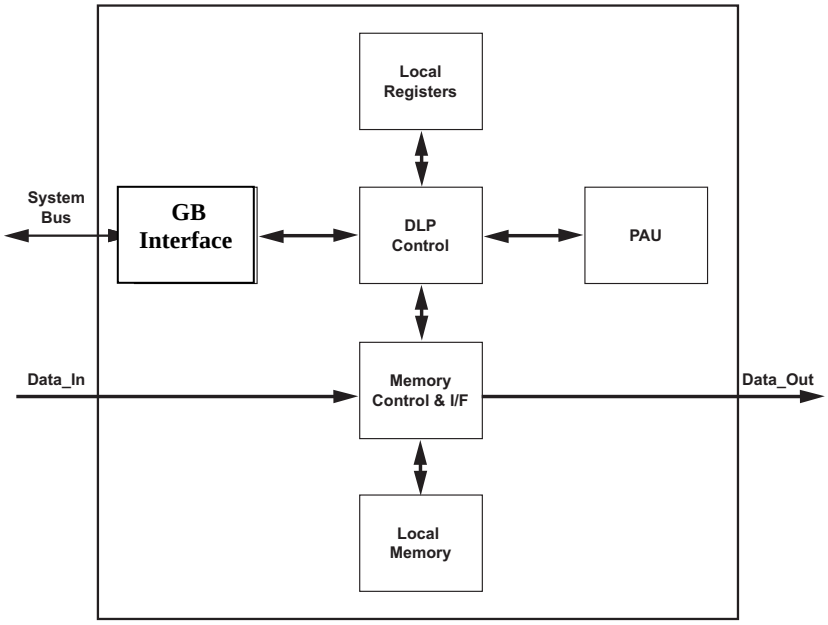


Figure 3.18: DWT line processor with inter processor communication bus.

The length of DWT filter determines the number of MAC units and consequently the PAU pipeline latency. Internal data registers are provided in the PAU for storing the DWT filter coefficients. The shift register unit provides one- and two-clock cycle delayed data to the next pipelined stage in the PAU. The MAC, designed for high speed and low latency, consists of an array multiplier and an accumulator with fast carry-chain logic for high speed performance.

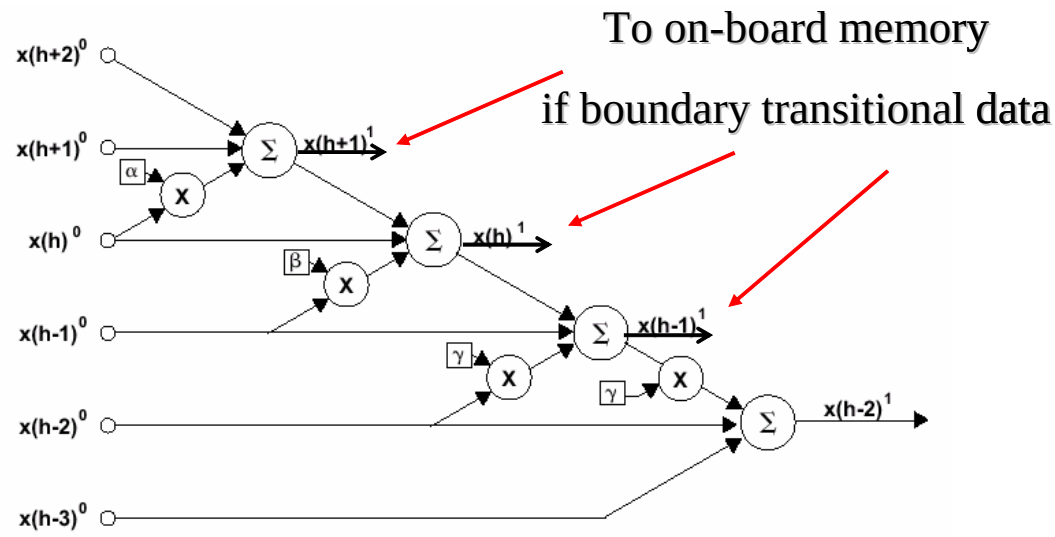


Figure 3.19: DWT filtering with lifting flow graph for the (9,7) DWT

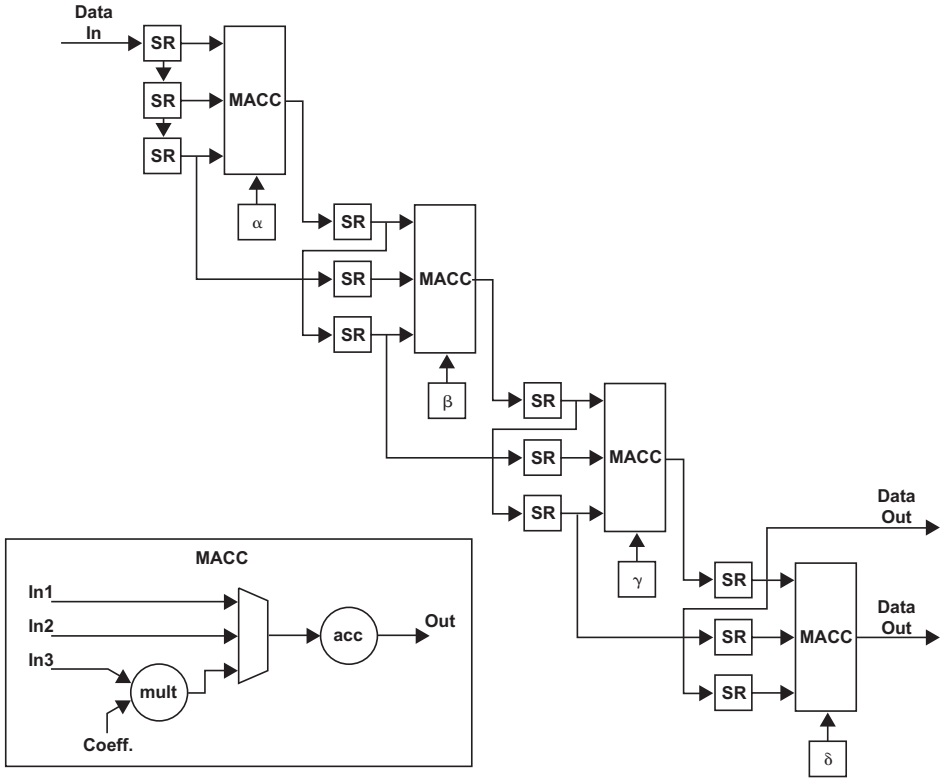


Figure 3.20: Pipelined Arithmetic Unit (PAU) for the (9,7) DWT

3.4.4 Implementations and Performance

The three parallel implementations for the (9,7) DWT, overlapping, state-save and overlap-state, were coded in VHDL and ported to the Xilinx Virtex II Pro (XC2VP70) field programmable gate array (FPGA) using a commercial board from the Dini Group. Evaluations were made with images of size 512x512 pixels and system clock frequency of 100 MHz. The FPGA utilization was ~43% (for 4 DWT parallel processors in the case of the overlap-save). Table 3.3 shows comparisons of the resource utilization for various number of parallel DWT modules. Performance benchmarks show more than 2 orders of magnitude acceleration over the c-code implementation and more than 3 times speed-up as compared to the parallel implementation of the standard algorithm, as can be seen in Figure 3.21. Further performance improvements are possible with additional parallel DWT modules. Table 3.4 shows that our implementation has throughput improvements of 1.4 to 3 times over other optimized implementations such as UCI's "software pipelines" [116], modified folded for SPIHT [43], and commercial IP such as Amphion [5] and Cast [59]), but with higher resources utilization.

Table 3.3: Resources Utilization for the Overlap-State Implementation

	One DWT Module	Two parallel DWT Modules	Four parallel DWT Modules
Number of Slices	3380 (10% of total available slices)	7267 (22%)	14196 (43%)
Number of Slice Flip Flops	3488 (5%)	7499 (11%)	14650 (22%)
Number of 4 input LUTs	5455 (8%)	11729 (18%)	22912 (35%)
Number of BRAMS	26 (8%)	56 (17%)	109 (33%)
Number of 18x18 Multipliers	16 (5%)	34 (10%)	67 (20%)

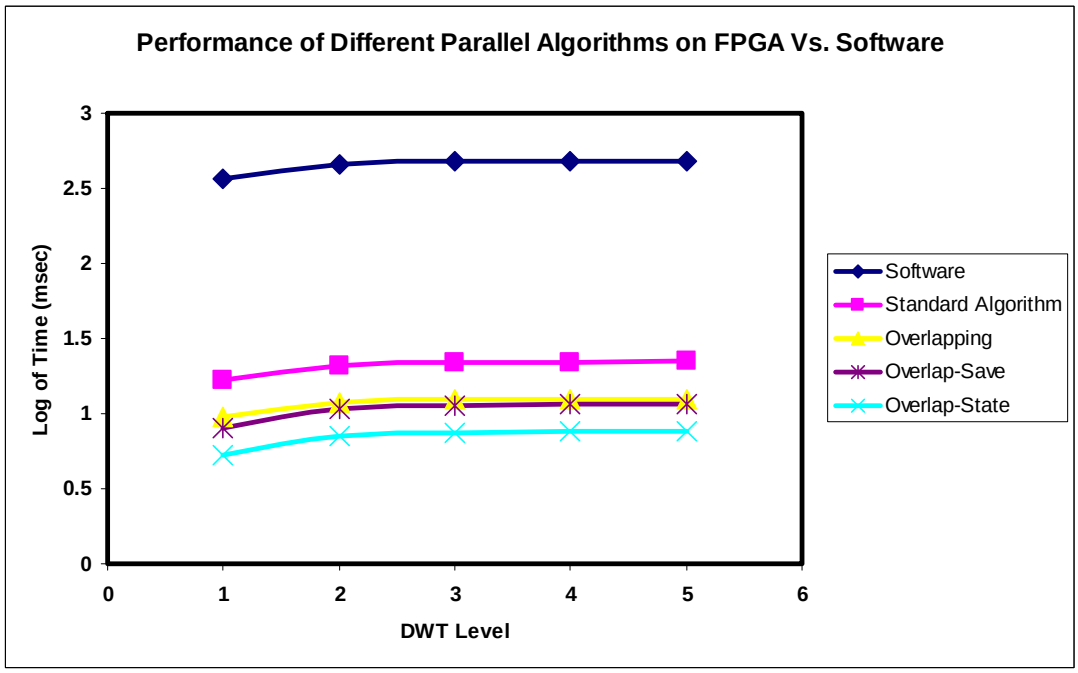


Figure 3.21: FPGA Parallel Implementations Performance for the (9,7) 2D DWT

Table 3.4: Resources Utilization and Throughput Comparisons to other Optimized Methods

Architecture	Cast Inc. (LB-2D)	Amphion	UCI "Software-pipelined"	Modified Folded for SPIHT	Our Overlap-state (Four parallel Modules)
Device	Virtex II 2V500	Virtex E-8	Virtex II 2V500	Virtex-E	Virtex II Pro XC2VP70
Number of Slices	2227	3784	986	(62% of chip resources)	14196 (43% of chip resources)
Clock Frequency (MHz)	51	-	98	75	100
Performance (Msamples/sec)	9	55	98	100	138
Image Size	256x256	128x128	1024x1024	512x512	1024x1024
DWT Trans. Level	5	5	5	5	5

3.5 Conclusions

We presented, in this chapter, a methodology for parallel implementation of the lifting DWT on FPGAs. We investigated and analyzed parallel and efficient hardware implementations targeting state-of-the-art FPGAs. We addressed practical considerations and various design choices and decisions at all design stages to achieve an efficient DWT implementation, subject to a given set of constraints and limitations. We presented a specific lifting representation for the DWT that provides architectures suitable for efficient hardware implementation, and a novel data transfer method that provides seamless handling of boundary and transitional states associated with parallel implementations.

We demonstrated our methodology with an implementation example for the (9,7) DWT, and also showed that it can be extended to operate under different platforms and constraints. We analyzed the implementations parameters to provide the best performance subject to practical considerations and platform constraints. We provided trade-off analysis for various implementations and comparisons to other existing implementations.

Chapter 4

Three Dimensional DWT coding for On-Board Hyperspectral Data Compression in Space Applications

4.1 Introduction

Imaging spectrometers or hyperspectral sensors are becoming increasingly common in deep space and Earth orbiting missions. Spatial and spectral resolutions of such instruments are on the rise to seek better data quality and improve the scientific or strategic value of the gathered information. The main limiting challenges to such new instruments are the available transmission bandwidth and on-board storage capacity. This makes compression of much greater value and a crucial part of the acquisition system.

Several approaches to hyperspectral data compression have been proposed in the literature. They include transform techniques based, in general, on a hybrid combination of two transforms. Typically they use KLT or Principal Component Transform to decorelate the spectral bands, and either DCT or DWT to spatially compress the selected high energy principal components [89][25]. Another example is based on the Modulated Lapped Transform (MLT), proposed by H. Hou [50], followed by DWT. Factorization methods were also proposed which include principal components, Gram Schmidt, stochastic modeling of the atmosphere, and

convex factorization [47][49][93]. These techniques are generally for content retrieval, are only used for lossy compression, and are often overly complex for on-board hardware implementation. Prediction based techniques such as differential pulse code modulation (DPCM) in both spectral and spatial domains are common [26][85]. All these techniques are mostly either lossy or lossless only and are non-progressive.

Recently, researchers proposed 3D DWT transform based coding such as 3DSPIHT (Set Partitioning in Hierarchical Trees) [40], 3DSPECK (Set Partitioned Embedded bloCK) [95] and 3D Tarp Coding (modified 3D DWT coding) [110]. While these techniques produce good compression results compared to 2D based algorithms, they may not meet future mission requirements due to their need for relatively complex post transform processing, which may not be suitable for on-board hardware deployment (due to speed and power impact). Locally Optimal Partitioned Vector Quantization (LPVQ) [84][88], recently proposed by Motta and Rizzo, has excellent lossy and lossless compression effectiveness, but its adaptive features are also computationally intensive, making them impractical for on-board spacecraft deployment.

In this chapter we investigate a wavelet based approach for three dimensional coding of hyperspectral data cubes suitable for on-board processing and hardware

implementation on reconfigurable platforms. Ideally, hyperspectral data compression should be lossless, to preserve the scientific data value. However, lossless compression may be limited in terms of achievable compression ratios due to noise inherent in such high-resolution sensors. Hence, a lossless/virtually lossless approach is adopted to address such applications. We adapt the 3D data sets to an efficient 2D wavelet based image compression, the ICER image compressor [62], by extending the wavelet decomposition to three dimensions and extending the bit-plane encoding scheme to operate with 3D data. The computationally intensive nature of compression-effective algorithms makes them impractical for software on-board deployment. Hence, our motivation throughout this research was suitability for hardware on-board implementation, which guided our design choices at every step of the algorithm development.

This chapter is organized as follows. Section 4.2 details our compression strategy, algorithm description, issues related to adaptation of 2D image compression to 3D data sets and issues encountered in porting the design to FPGA hardware platforms. In section 4.3 we present our experimental compression results for AVIRIS data sets. Section 4.4 covers extensions and applications related to hyperspectral data compression and section 4.5 concludes this chapter with a summary and conclusions.

4.2 Three Dimensional Coding

4.2.1 Compression Strategy

Our strategy was to develop a general-purpose methodology for hyperspectral data compression that efficiently exploits spectral (inter-band) correlations as well as spatial correlations. To addresses requirements of various instruments – i.e. lossless/lossy compression, progressive compression, real-time constraints and low memory utilization. In order for it to be considered for space missions insertion, it needs to be low-complexity and suitable for on-board hardware implementation (low power and mass impact). In addition, the compressed bit stream should be suitable for progressive browsing, target detection and classification, and extendedable to accommodate region-of-interest (ROI) compression. Suitability for push-broom type sensors is also important since hyperspectral data is mostly collected in band interleaved by pixel (BIP), or sometimes in band interleaved by line (BIL) formats. Finally, the strategy should address various applications' distortion metrics - objective metrics, in terms of mean squared error (MSE) or peak signal to noise ratio (PSNR), and subjective metrics, in terms of visualization, signature extraction, or classification. To pursue these goals we started our investigation by adapting 2D compression-effective algorithms, such as 2D discrete wavelet transform (DWT) based algorithms, to 3D hyperspectral data.

4.2.2 From 2D to 3D Wavelet Coding

The chosen algorithm, based on the reversible integer DWT, leads to a progressive encoder capable of lossless and lossy compression in a single system. Our approach relied on extending an efficient 2D image compression algorithm, namely the JPL ICER [62] image compressor, to 3D data sets. This required extending the 2D wavelet decomposition to 3D decomposition as shown in Figure 4.1 below. The resultant subband cubes required a major modification of post DWT transform coding, such as the bit-plane coding, which will be described in the following sections. A block diagram of the compression process is shown in Figure 1.4. We used a line based DWT scheme, operating in scan-mode, to accommodate pushbroom sensors. It should also be noted that low-complexity and portability to FPGA hardware implementation was a major driver at every step of the algorithm design.

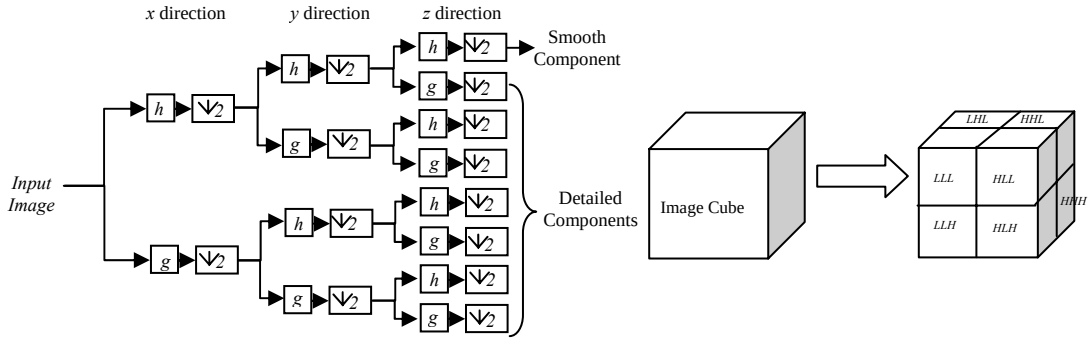


Figure 4.1: Integer based DWT is applied to all three dimensions of the image cube

4.2.2.1 From ICER-2D Image compression to ICER-3D-HW

The ICER [62] image compressor was developed at JPL to meet the requirements of the Mars Exploration Rover mission (MER). It is currently deployed on the Spirit and Opportunity rovers and continues to be used to send most of their images to Earth from Mars [64][65]. We extended the ICER algorithm to hyperspectral data by expanding the discrete wavelet decomposition to three dimensions and adapting its bit-plane encoding scheme, which uses a context modeler similar to the EBCOT coder in JPEG2000 [98], followed by an interleaved entropy coder. The ICER 2D image compressor block diagram is shown below in Figure 4.2.

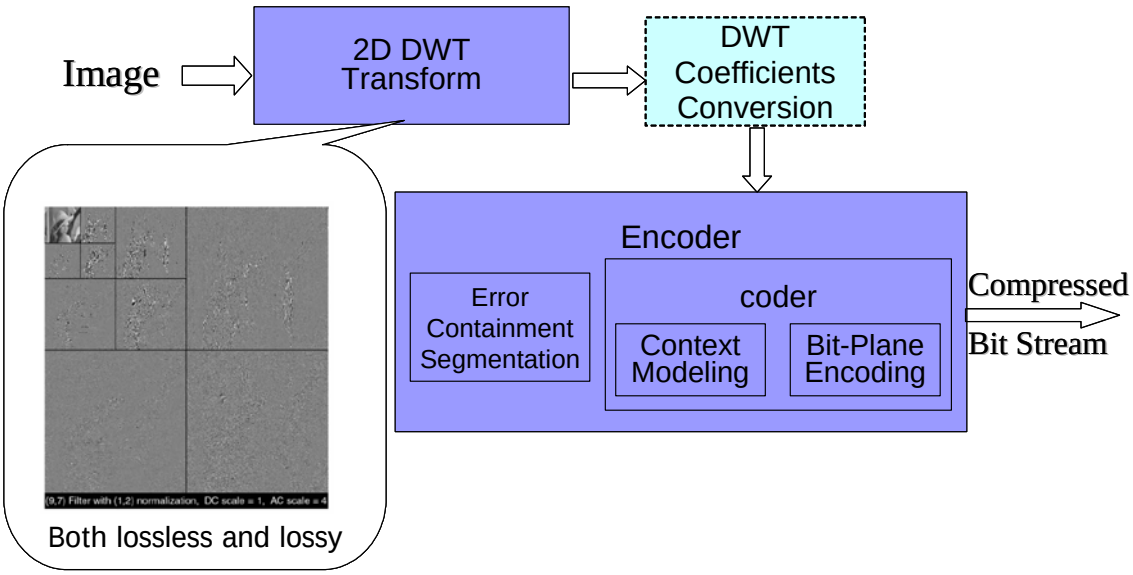


Figure 4.2: ICER 2D Image Compression

4.2.2.2 Bit Plane Encoding for 3D data sets

The extension of ICER-2D bit-plane encoding to 3D decompositions was done in the following manner. After the wavelet decomposition, each subband cube is assigned an index, with indices numbered starting from 0. The index assignment is used to determine the order in which different subband bit planes are compressed. Let L and H denote number of stages of high-pass and low-pass filtering used to form a subband, indices are assigned by sorting the subbands according to the following subband-ordering scheme [63]:

(1) A subband with a larger value of $L - H$ has a higher index. This has the effect of giving higher indices to subbands with higher priority bit planes as will be seen below.

(2) For two subbands with the same value of $L - H$, if one of them has fewer coefficients, then it is given a higher index (equivalently, subbands formed through a larger number of wavelet-filtering operations, i.e., larger value of $L + H$, are given a higher index.).

(3) If the two subbands are equivalent in the preceding considerations, then a higher index is given to a subband that is low-pass in the vertical direction.

(4) If the two subbands are equivalent in the preceding considerations, then a higher index is given to a subband that is low-pass in the horizontal direction.

Bit-plane priority assignments:

For each subband we can determine a weight that indicates the approximate relative effect, per coefficient of the subband, on mean squared error (MSE) distortion in the reconstructed image [63]. These weights determine the relative priorities of subband bit planes. The weight is expressed in terms of the number of stages of high-pass and low-pass filtering operations, *H* and *L*, used to form a subband. For example, let's consider the front top right green subband cube shown earlier in Figure 2.5. To form this subband cube, we apply low-pass filtering in all dimensions (horizontal, vertical and spectral) twice, then high-pass filtering in horizontal direction followed by high-pass filtering in both the vertical and spectral dimensions. Thus $H = 1$ and $L = 8$ for this subband. The weight w assigned to bit plane *b* of a subband depends on *H* and *L*:

$$\omega = 2^b.(\sqrt{2})^{L-H} = (\sqrt{2})^{2b+L-H} \qquad (4.1)$$

Bit planes in a subband are indexed starting with $b = 0$ for the least significant bit. This weight scheme is a 3D extension of the weight scheme used in ICER [62] for the 2D case. Any monotonic function of the weight in equation (4.1) can be used to determine the relative importance of subband bit planes, so rather than keeping track of real-valued weights given by equation (4.1), we define integer "priority" values p of subband bit planes, given by:

$$p = 3 + \log_{\sqrt{2}}(\omega) = 2b + L - H + 3 \qquad (4.2)$$

This definition produces a minimum priority value of 0, since $H \leq 3$ *and* $L \geq 0$ for all

subbands. As an example, if $H = 2$ and $L = 6$ for a subband cube, bit plane b of this subband is assigned priority value $p= 2b + 7$. Thus, all bit planes of this subband have odd priority value, with minimum value of 7. Take another subband, $H = 2$ and $L = 1$. Bit plane b of this subband has priority $p = 2b +2$. For this subband all bit planes have even priority value, with a minimum value of 2. Since L and H are fixed for a given subband, all of the bit planes in a subband have even-valued priority, or all have odd-valued priority.

Mean subtraction and Sign-Magnitude Coding:

The mean of each low frequency subband plane is subtracted (similar to the case in ICER 2D, but extended to cover planes of the low frequency subband cube) in preparation for the next stage of coding. Each DWT coefficient is converted to sign-magnitude form. Magnitude bit planes of subbands are compressed one at a time; when the first '1' magnitude bit of a coefficient is encoded, the sign bit is encoded immediately afterwards. Compressed bit planes of different subbands planes are interleaved, with the goal of having earlier bit planes yield larger improvements in reconstructed image quality per compressed bit. Subband bit planes are compressed in order of decreasing priority value according to the simple priority assignment scheme described earlier. Bit planes having the same priority value (which are from

different subbands) are compressed in order of decreasing subband index, using the ad hoc index assignment scheme described earlier, which aims to improve compression performance while maintaining low-complexity [63].

Context Modeler and Entropy Coder:

Driven by the low-complexity requirements, the context modeler designed for the 3D coding algorithm (which will be referred to as ICER-3D-HW, since this is the version that was ported to hardware), is similar to the one deployed by ICER 2D, but works on planes of the subband cubes. Before encoding a bit, the encoder calculates an estimate of the probability that the bit is a zero. This probability-of-zero estimate relies only on previously encoded information from the same plane. The bit and its probability-of-zero estimate are sent to the entropy coder, which compresses the sequence of bits it receives. For entropy coding, ICER-3D-HW uses an interleaved entropy coder; the same as that used by ICER and described in [62]. Probability estimates are computed using a technique known as context modeling. With this technique, a bit to be encoded is first classified into one of several contexts based on the values of previously encoded bits. The intent is to define contexts that divide bits with different probability-of-zero statistics into different classes, for which separate statistics are gathered. The compressor can then estimate these probabilities-of-zero reasonably well from the bits it encounters in the contexts. The simple adaptive

procedure used by ICER, and described in [62], to estimate probabilities, was also extended to 3D data sets.

ICER-3D-HW employs a two-dimensional context model relying on eight (spatial) neighbors in a subband plane. Coding of a subband bit plane proceeds from one spatial plane to the next, and in raster scan order within a spatial plane. Pixels are assigned categories 0 through 3 as shown in Figure 4.3 and bits are classified into one of 17 contexts (this is derived from the EBCOT encoder and JPEG2000) [98][57].

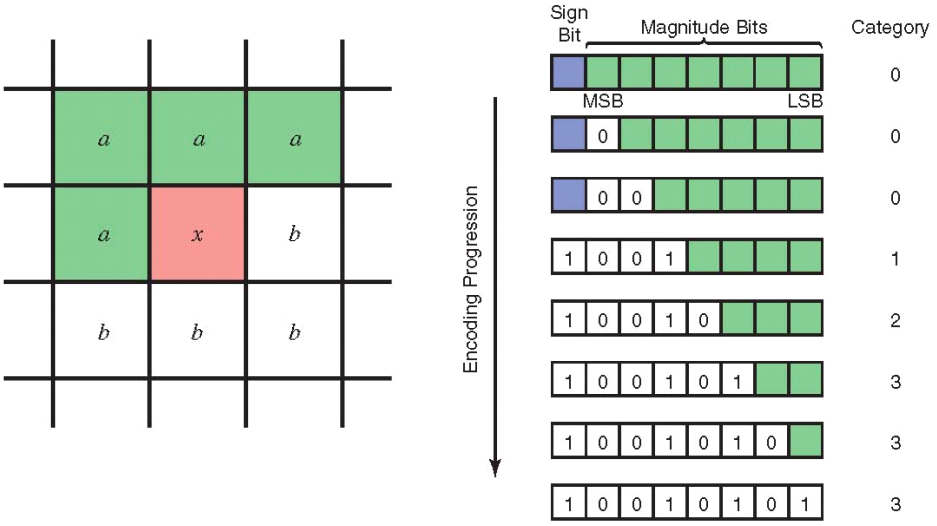


Figure 4.3: Progression of categories of a pixel as its magnitude bits and sign are encoded

For each bit *bi* in a pixel of a subband, the context modeler produces an estimate *pi* of the probability that *bi* = 0. The entropy coder uses theses estimates to produce an encoded bit stream. The design choice for an entropy coder can ideally be an

adaptable binary coder (arithmetic coding), or a low-complexity approximation to it. The interleaved entropy coder of ICER was adopted here due to some speed advantages. The coder compresses a binary source with a bit-wise adaptive probability estimate by interleaving the output of several different variable-to-variable length binary source codes. Later optimization of algorithm resulted in a 3D context modeler and a new ICER-3D compressor described in [63], but it has not been modified yet for hardware implementation and is not discussed here.

ICER-3D-HW, inherits, from ICER, a segmentation scheme for error containment that encodes segments of the subband planes, rather than the whole plane, to allow partial reconstruction of the decoded image when an encoded packet is lost. The segmentation scheme is scalable to allow different levels of error containment. Details of this segmentation scheme are described in [62][63].

4.2.2.3 Spectral Ringing Artifacts in 3D DWT Coding

3D DWT Limitations

A straightforward extension of wavelet-based two-dimensional image compression to hyperspectral image compression, based on a three dimensional wavelet decomposition, results in compression-ineffective coding of some subbands and can lead to reconstructed spectral bands with systematic biases. Thus, using a wavelet transform for spectral decorrelation of hyperspectral data does not account for systematic differences in signal level in different spectral bands. In addition, the

spectral dependencies are not limited to the small spectral neighborhood exploited by the wavelet transform and this will require further modification to the decomposition or the coding scheme.

The "Spectral Ringing" Problem

When the Mallat decomposition shown in Figure 4.4 is used as 3D DWT for decorrelation, "ringing" artifacts in the spectral dimension can cause the spatially low-pass subbands to have large biases in the individual spatial planes that manifesting themselves as systematic biases in some reconstructed spectral bands. Specifically, spatial planes of spatially low-pass subbands contain significant biases that vary from plane to plane [66][72][73]. These biases appear in the spatially low-pass subband as can be seen in 4.5. This problem is somewhat unique to multispectral and hyperspectral data; an analogous artifact does not generally arise in images. The encoding scheme adapted from ICER [62] assumes that, except for the low-pass subband cube, the means of subband planes of all DWT subband cubes are zero. Histogram analysis for planes from the spatially low-pass subband cubes shows that often the means are not zero, as can be seen in Figure 4.6. This phenomenon hurts the rate-distortion performance at moderate to low bit rates (~1 bit/pixel/band and below) and occasionally introduces disturbing artifacts into the reconstructed images. Figure 4.7 shows an example of these artifacts.

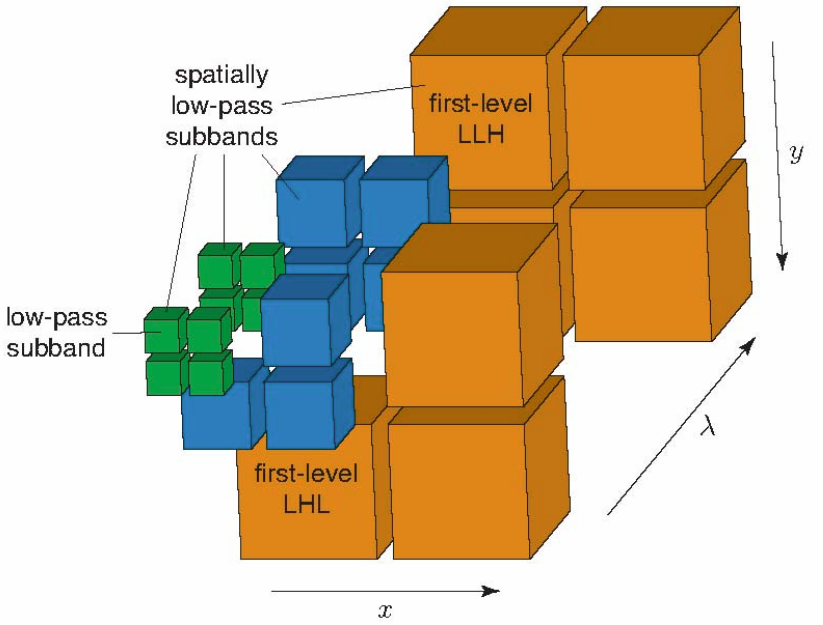


Figure 4.4: 3D Mallat DWT Decomposition

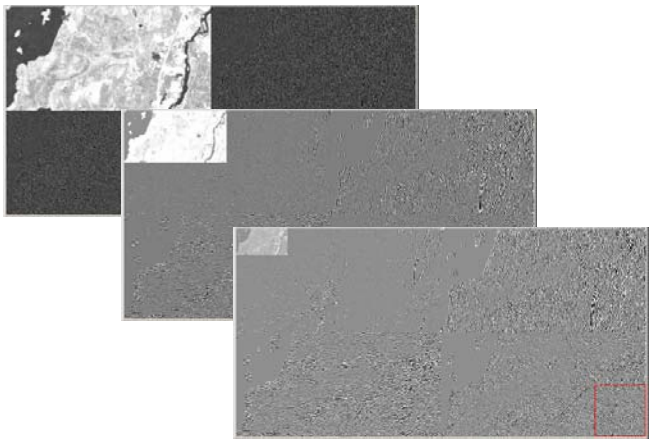


Figure 4.5: Sample Planes from Subband Cubes from 3 Different DWT Stages

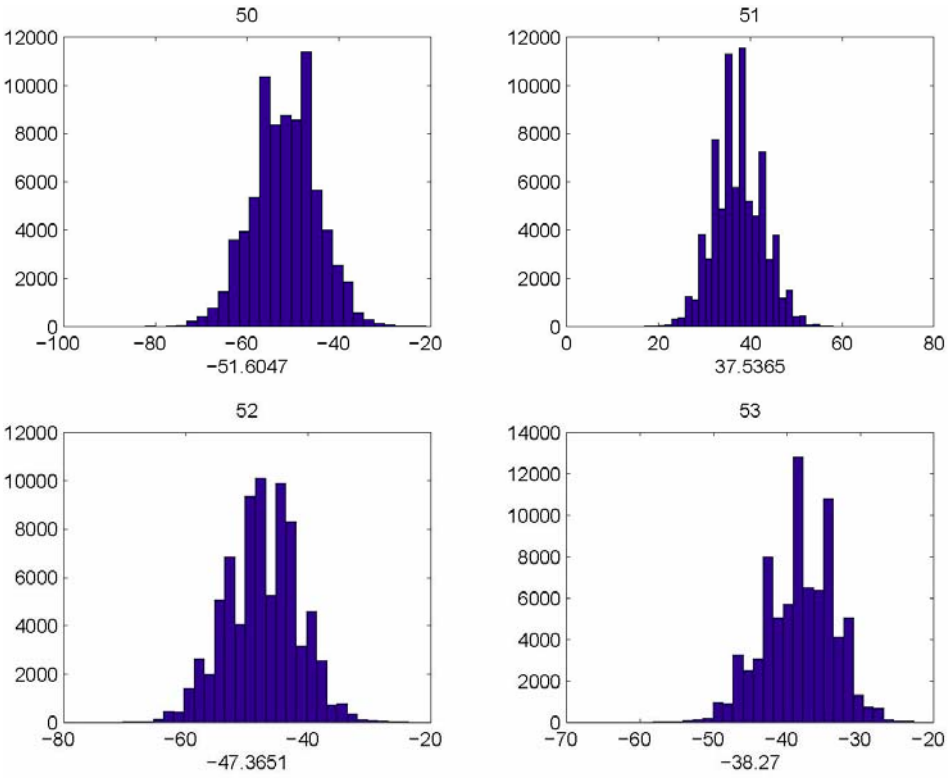


Figure 4.6: Histograms of DWT coefficient values in 4 subbands planes from AVIRIS Cuprite scene. All planes are from the first level LLH subband (planes 50 to 53) showing a non zero mean.

Figure 4.7: Spectral Ringing: Original image (right) and reconstructed from 0.0625 bits/pixel/band compressed AVIRIS image

Mitigation of Spectral Band Signal Level Variations

Two methods were developed for mitigating the spectral ringing effects described above. Compression results illustrating the benefits of these methods are presented in section 4.3.3. The two methods are as follows:

Mean Subtraction.

The basic idea of this method is simply to subtract the mean values from spatial planes of spatially low-pass subbands prior to encoding, thus compensating for the fact that such spatial planes often have mean values that are far from zero. The resulting data are better suited for compression by methods that are effective for subbands of 2D images such as the ones used for ICER [62] and described earlier in this chapter.

Additional DWT Decompositions

An alternative approach to mitigate this problem is to perform additional DWT decompositions. Not only is the low-pass subband further decomposed, but spatially low-passed, spectrally high-pass subbands are also further spatially decomposed.

These two methods can be combined, i.e., we can perform the modified decomposition and then subtract the mean values from spatial planes of the spatially low-pass subbands. In the context of ICER-3D-HW, the mean subtraction method is

easy to implement as follows. After the 3D wavelet decomposition is performed, mean values are computed for, and subtracted from, each spatial plane of each error-containment segment of each spatially low-pass subband cube. The resulting data is converted to sign-magnitude form and compressed as in the baseline ICER-3D-HW. The mean values are encoded in the compressed bit-stream and added back to the data at the appropriate decompression step. The overhead incurred by encoding the mean values is only a few bits per spectral band per segment, which is negligible because of the huge size of hyperspectral data sets. Note that it is important to subtract the means after all stages of subband decomposition; otherwise if two adjacent error-containment segments have significantly different means, a sharp edge would appear after subtracting the means, artificially increasing high-frequency signal content in further stages of spatial decomposition.

Other researchers have also used modifications to the Mallat decomposition for hyperspectral image compression. For example, in 3D tarp coding and 3D SPIHT [110] [96] the wavelet decomposition used is equivalent to a 2D Mallat decomposition in the spatial domain followed by a 1-D Mallat decomposition in the spectral dimension. The resulting overall decomposition has further decomposed subbands compared to our modified decomposition with the same number of stages. Because all of the transform steps of our modified decomposition are included in the decomposition of [110][96], the latter enables a similar advantage in compression

effectiveness. Alternatives to the Mallat 3D wavelet decompositions have also been used for compression of 3D medical data sets [114], and video coding [69][101].

4.2.3 From Software to Hardware – FPGA Implementation Considerations

In general, when moving an algorithm from software to hardware, major modifications are needed to tailor the algorithm to a hardware platform, in our case the FPGA platform, in order to take full advantage of the high performance of the target hardware platform. These changes include precision analysis, simpler architectures for coding and schemes that minimize I/O operations. Keeping these issues in mind in the design phase of the algorithm makes this transition simpler. Since the ICER-3D-HW is lossless and lossy, precision and fixed point analysis are not needed due to the fact that our hyperspectral sensory data and our DWT filters are integer-valued. However, dynamic range expansion in the DWT may occur after several filtering operations for certain filters, resulting in excessive memory requirements or the need to quantize the DWT coefficients and make the algorithm lossy. The next section will describe the analysis and design choices used to overcome this issue. Other issues we consider for migration from software to hardware are the choices for the context modeler and mitigation techniques for the spectral ringing artifacts in 3D DWT.

4.2.3.1 Dynamic Range Expansion for DWT Data

In general, the range of possible output values from a reversible DWT can be larger than the range of input values [81]; such an increase can be seen as a dynamic range expansion. The amount of dynamic range expansion can increase with the number of filtering operations. Dynamic range expansion can be an issue because storage of wavelet-transformed samples may require binary words that are larger than those used for the original samples. In particular, one must pay attention to the degree of dynamic range expansion if the wavelet decomposition is performed in-place, i.e., when memory locations originally used to store image samples are subsequently used to store DWT coefficients, as is the case in most hardware implementations of the DWT.

For the filters used in ICER and ICER-3D-HW, low-pass filtering does not expand the dynamic range, but high-pass filtering does. The dynamic range expansion following a single one dimensional high-pass filtering operation can be described [62] by the approximation

$$h_{\max} - h_{\min} \approx (x_{\max} - x_{\min})a. \qquad (4.3)$$

Here *Xmax* and *Xmin* denote the maximum and minimum possible values input to the DWT, and *hmax*, *hmin* denote the maximum and minimum possible values output from the (one dimensional) high-pass filtering operation. As noted in [62],

$h_{\max} \approx -h_{\min}$. The constant $\boldsymbol{\alpha}$ is equal to the sum of the absolute values of the filter taps for the linear filter that approximates the particular high-pass filter. Thus, each additional stage of high-pass filtering results in dynamic range expansion by a (filter-dependent) factor $\boldsymbol{\alpha}$, or $\log_2 \boldsymbol{\alpha}$ bits. Under the decomposition structure used by ICER-3D-HW, each subband is produced using at most one high-pass filtering operation in each of the three dimensions *(x, y, or λ)*, so the worst-case dynamic range expansion comes from three high-pass filtering operations. Table 4.1 shows the dynamic range expansion resulting from up to three high-pass filtering operations for the filters used by ICER-3D-HW.

The last column of Table 4.1 can be used to determine the binary word sizes required to accommodate dynamic range expansion for a given source bit depth, or conversely, determine the restriction on source bit depth for a given storage word size. For example, 16-bit words are sufficient to store the coefficients produced by applying a 3D decomposition, using the (2,6) DWT filter pair (filter A) on 12-bit data (such as uncalibrated AVIRIS data). But the other filter choices may produce DWT coefficients that cannot be stored in 16-bit words following 3D wavelet decomposition.

However, if such a filter pair is not used and expansion does occur for the transform coefficients, there are some techniques that may be used to relax the requirements,

with minor costs. For example, quantization of DWT output could be performed at intermediate stages of the decomposition to reduce the dynamic range as needed. This method sacrifices the ability to perform lossless compression, and it may slightly decrease compression effectiveness at high rates, but it may be quite practical when lossless or near-lossless compression is not needed. For all examples presented in this chapter, as well as for the hardware implementation presented in the next chapter, wavelet transforms are performed using filter A, which is the integer (2,6) DWT filter pair described in [62], [1] and [90].

Table 4.1: Approximate Dynamic Range Expansion following 1, 2 and 3 filtering

Filter	One High-Pass Filtering Operation		Two High-Pass Filtering Operation		Three High-Pass Filtering Operation	
	a	$\log_2 \boldsymbol{a}$ bits	a^2	$\log_2 a^2$ bits	$\boldsymbol{a^3}$	$\log_2 \boldsymbol{a^3}$ bits
A	5/2	1.32	25/4	2.64	125/8	3.97
B	11/4	1.46	121/16	2.92	1331/64	4.38
C	25/8	1.64	625/64	3.29	15625/512	4.93
D	41/16	1.36	1681/256	2.72	68921/4096	4.07
E	47/16	1.55	2209/256	3.11	103823/4096	4.66
F	51/16	1.67	2601/256	3.34	132651/4096	5.02
Q	11/4	1.46	1/16	2.92	1331/64	4.38

The aforementioned analysis can also be used to aid filter designers in the selection of filter coefficients that are hardware and memory friendly when used for compression or analysis of data. As mentioned earlier, low-pass filters do not cause dynamic range expansion and no constraints need be applied here. For high pass filters, if the maximum desired dynamic range expansion is B (in our case B = 4), the

sum of the absolute values of the filter coefficients must satisfy the following equation:

$$\log_2\left(\sum_{0}^{n-1} |\, h_i \,|\right)^J \le B \qquad (4.4)$$

Where *hi* is a coefficient of a high pass filter of length n, and J is the maximum number of levels in the DWT decomposition.

4.2.3.2 Context Modeler Design for HW Implementation

While a 3D context modeler that covers the third dimension of the subband cubes has noticeably better compression-effectiveness than a 2D context modeler (as was shown by JPL researchers [63]), the increased computation complexity and data I/O makes it impractical for hardware implementation without major modifications. ICER-3D-HW employs the two-dimensional context model described in Section 4.2.2.2, relying on eight (spatial) neighbors in a subband plane and operating on one subband plane at a time.

4.2.3.3 Mitigation of Spectral Ringing Artifact in HW

Seeking the low-complexity solution for the FPGA hardware implementation, the mean subtraction method described in Section 4.2.2.3 was selected for the hardware implementation described in the next chapter. The alternative approach to mitigate

this problem, i.e., the use of additional DWT decompositions, is far more complex and produces minor improvements in comparison to the mean subtraction method (as will be shown in the next section). The hardware design complexity and the additional resources (and thus, higher mass and power) required, are not justifiable for our applications.

4.3 Experimental Results

In this section we present our compression results for both lossless and lossy hyperspectral data compression performed on various AVIRIS data sets. We show comparisons to other 3D coding methods and state-of-the-art 2D coding algorithms. We also show results demonstrating the effects of techniques to mitigate the spectral ringing artifacts discussed earlier in this chapter.

4.3.1 Lossless Compression

Tests were performed using available AVIRIS data sets (calibrated and uncalibrated). Data sets were divided into image cubes of 512 lines x 614 pixels each. These calibrated data sets represent 1997 scenes from Moffett Field (vegetation, urban, water), Cuprite (geological features), Jasper Ridge (vegetation), Lunar Lake (calibration), and Low Altitude (high spatial resolution) [20]. Table 4.2 shows the lossless compression performance of ICER-3D-HW on these five calibrated AVIRIS radiance data sets. For comparison, Table 4.2 also shows results

for the "fast lossless" compressor from [71], the Rice compressor used in the Universal Source Encoder for Space (USES) chip using the multispectral predictor option mentioned in [52], ICER-2D applied independently to individual spatial planes, JPEG-LS image compressor [111], JPEG2000, and locally optimal partitioned vector quantization (LPVQ) [84]. 3DSPIHT, 3DSPECK [97] and JPEG2000 multi-component [58] results are available and shown here for scene 1 of the Jasper Ridge 1997 reflectance scene. The results of Tables 4.2 indicate that ICER-3D-HW provides more effective lossless compression than simple two-dimensional approaches, the USES multispectral compressor and all 2D approaches. But ICER-3D-HW is outperformed by the simpler fast lossless compressor of [71], which was designed to be a lossless compression only, and LPVQ which is highly complex due to its data dependent adaptive nature.

Table 4.2: Lossless compression results (bits/sample) for calibrated 1997 AVIRIS data sets

Dataset	ICER-3D-HW	fast lossless	Rice/USES multil	ICER (2D)	JPEG-LS (2D)	JPEG 2K	LPVQ	3D-SPECK	3D-SPIHT	JPEG2K Multi
Cuprite	5.80	4.95	6.04	6.95	7.24	8.37	5.28	*	*	*
Jasper Ridge	6.12	5.04	6.17	7.60	7.78	8.96	5.42	*	*	*
Low Altitude	6.35	5.34	6.47	7.36	7.66	8.89	5.76	*	*	*
Lunar Lake	5.72	4.97	5.99	6.79	6.97	8.16	5.25	*	*	*
Moffett Field	5.96	5.07	6.13	7.22	7.46	8.79	5.51	*	*	*
Jasper Scene1 (Reflectance)	6.81	6.07	6.63	8.42	*	8.59	*	6.70	6.72	6.9
Average	6.13	5.24	6.24	7.39	7.42	8.63	5.44			

Figure 4.8 shows results for uncalibrated (raw) data compressed by ICER and ICER-3D-HW. The table shows that higher compression-effectiveness can be achieved on raw data as compared to encoding calibrated data. This indicates that calibrated data has additional artifacts that the algorithm does not adapt to. On-board

compression will operate on raw sensory data coming from the spectrometer; hence further investigation of performance on calibrated data is not needed.

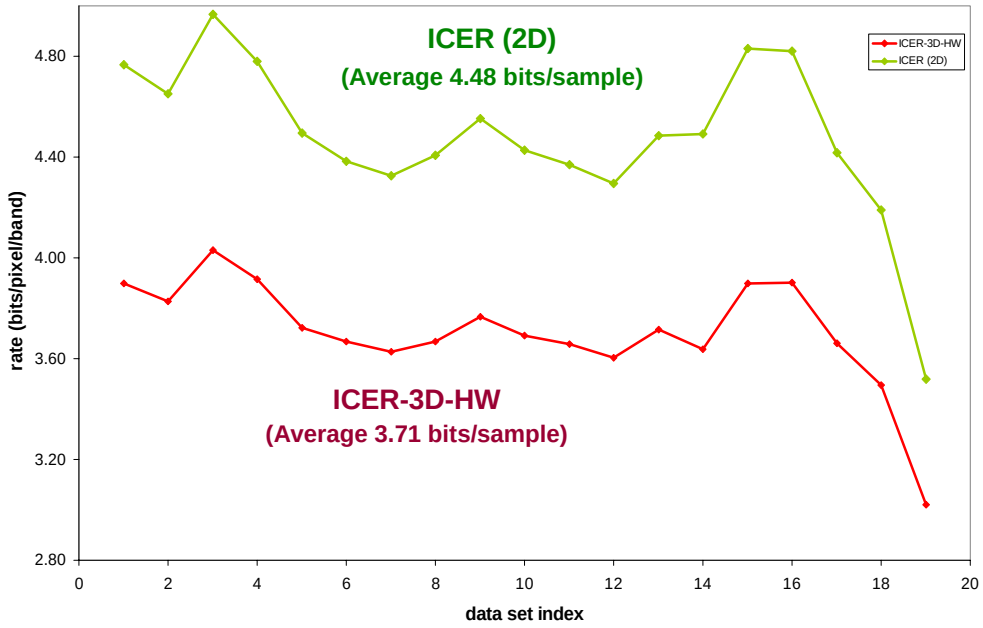


Figure 4.8: Lossless Results with uncalibrated AVIRIS Data - Tests using 512 line scenes from uncalibrated (raw) AVIRIS data sets(Original data 12bits/sample)

4.3.2 Lossy Compression

Figure 4.8 shows the rate-distortion performance comparison of ICER-3D-HW and ICER-2D for the AVIRIS '97 Cuprite scene. In both cases, compression was performed using three stages of wavelet decomposition. ICER-2D results were obtained by applying ICER independently to individual bands. ICER-3D-HW results were obtained by the 3D extension of ICER described earlier; specifically, using a 3D Mallat decomposition combined with spatial context models. Figure 4.9 shows a comparison between ICER-3D-HW and a baseline 3D DWT approach that uses

DWT coefficients quantization and entropy coding. The figure demonstrates the effectiveness of the context modeling approach used in ICER-3D-HW.

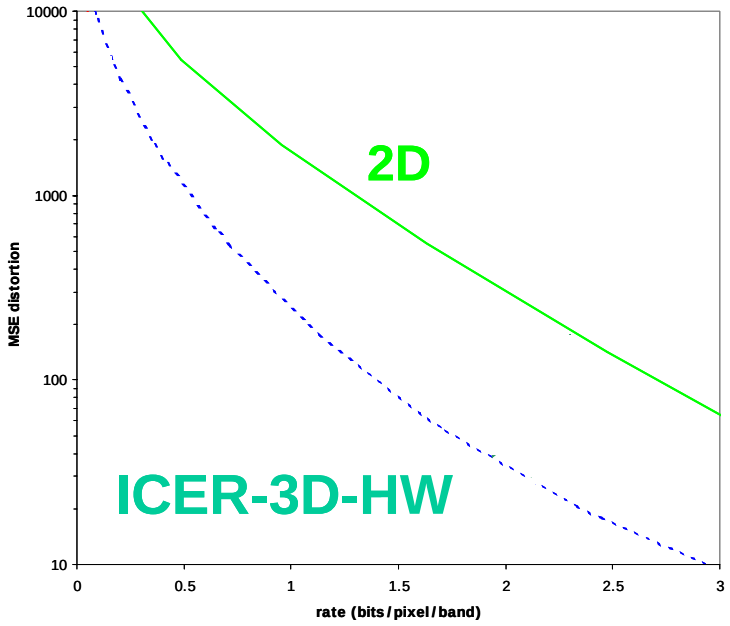


Figure 4.9: Comparison of Lossy Compression between ICER-2D and ICER-3D-HW

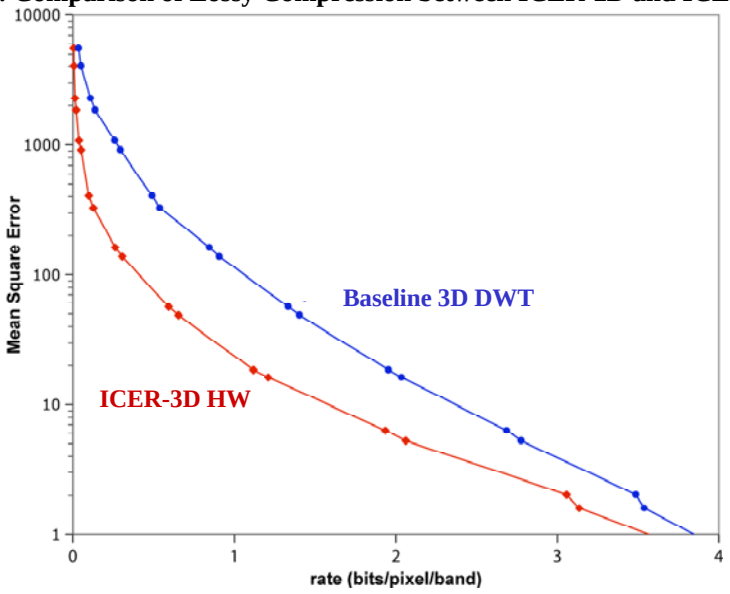


Figure 4.10: Comparison of Lossy Compression between a baseline approach and ICER-3D

4.3.3 Results from Mitigation Techniques of the Spectral Ringing Artifacts

The methods described earlier for the mitigation of the spectral ringing artifacts provide a noticeable improvement in rate-distortion performance compared to the baseline approach, especially at moderate to low bit rates (roughly 1 bit/pixel/band and below). In Figure 4.11 we compare the rate-distortion performance of these methods to the baseline approach. Results shown are for a 512 line radiance data scene of Cuprite, Arizona. The points shown on the curves were produced by compressing all bit planes up to a specific level of significance. It can be seen that mean subtraction and the modified decomposition provide very similar rate-distortion performance, and give roughly a 10% improvement in rate compared to the baseline method at 1 bit/pixel/band. When the number of wavelet decompositions is small, the rate-distortion performance of modified decomposition alone is slightly worse than using mean subtraction or the combination of the two methods.

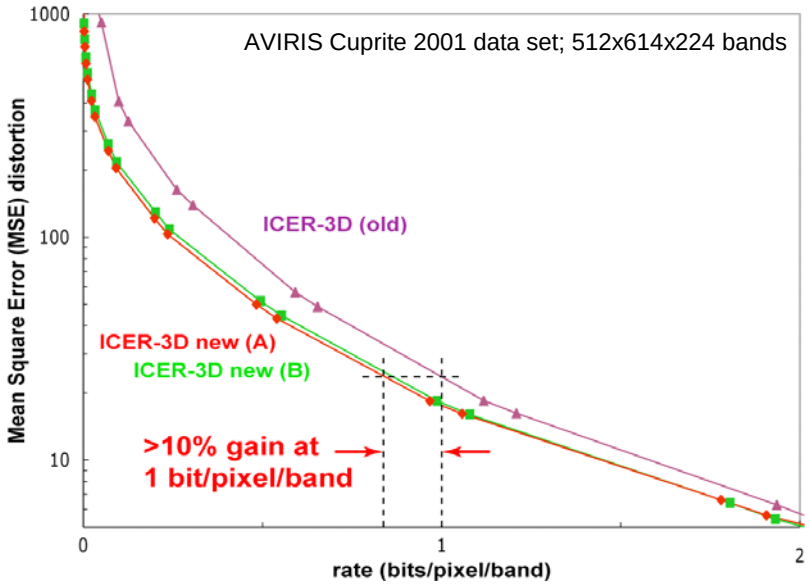


Figure 4.11: Rate-distortion performance and baseline ICER-3D-HW for the Cuprite scene. (A) Mean subtraction (B) Additional DWT decompositions.

Overall, the use of either method from Section 4, with ICER-3D-HW, provides a moderate subjective image quality improvement consistent with the improvement in mean squared error (MSE) distortion [72][73]. In some cases, however, the improvement is more dramatic, especially with regard to reduction of bias in reconstructed images when compressed at low bit rates. This is illustrated in the false-color images of Figures 4.12. Band 176 was deliberately chosen because its reconstruction exhibits a noticeable bias when using the baseline ICER-3D-HW on these scenes. This bias can be seen as an apparent overall color change under the baseline ICER-3D-HW and, to a somewhat lesser degree, under mean subtraction. The mean subtraction method was incorporated into ICER-3D-HW and ported to the FPGA implementation as detailed in the next chapter.

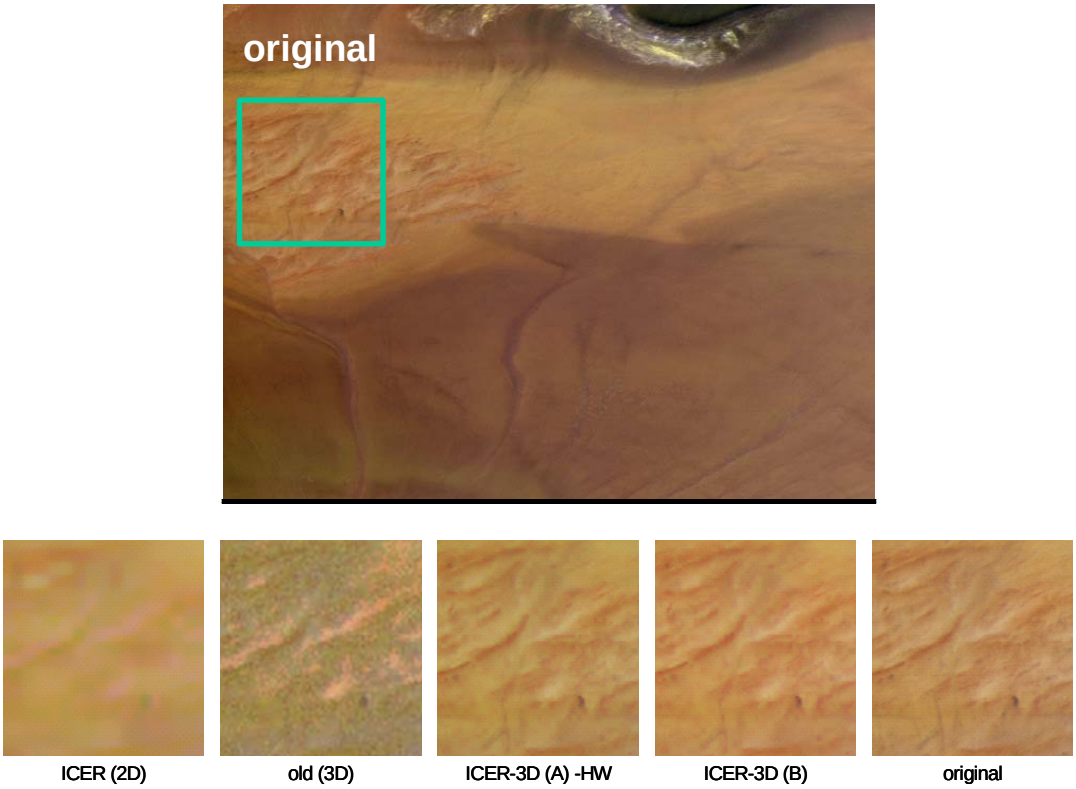


Figure 4.12: Comparison of detail region using different compressors at 0.0625 bits/pixel/band. (A) Mean subtraction (B) Additional DWT decompositions

4.4 Applications and Metrics

4.4.1 Region-of-Interest coding for 3D data sets

Typically, spacecraft imagers and remote sensors have the capability to collect far more data than can be transmitted to earth. Remote sensing image users are usually interested in only partial regions of the image sequence. That is to say, certain regions are more important than other regions. On-board processing algorithms can

recognize relevant features in the collected data, and hence it is unnecessary to treat all image pixels equally. Insignicant regions should be highly compressed or assigned to zero bit to minimize the total number of bits, thereby reducing transmission time and cost without losing the analysis quality of the image sequence. The available bandwidth can then be reallocated by spending more bits in the regions of interest (ROIs), which speeds up and facilitates browsing of large datasets for remote sensing applications.

Several algorithms have been proposed for ROI image compression. Progressive data compression algorithms such as wavelet-based image compression can be used for this purpose. During progressive compression, the image data is parsed into hierarchical data segments that yield continual but diminishing improvement of fidelity with each segment. The JPEG2000 image coding standard defines two kinds of region of interest (ROI) compression methods; the general scaling based method and the maximum shift method [99]. The two methods reduce compression efficiency by increasing the dynamic range (or number of bit planes) of wavelet coefficients, and they do not have the special protection for the ROI against the bit errors in communication, Fukuma *et al* [44] proposed to use a wavelet transform composed of two wavelet filter sets with different tap lengths, the shorter-length set to code an ROI of an image and to the longer-length one for the remainder of the image. While such an approach results in improved overall compression efficiency it

adds an additional level of complexity to the system. Ding *et al* [37][38] introduced an approach based on Wyner-Ziv theorem (source coding with side information). In such approach, the reconstructed low quality ROI image is treated as side information and can be utilized by a turbo decoder to decode the high quality ROI. While it improves the compression efficiency as well as efficiently protecting the ROI against bit errors, it is not progressive.

Our interest in this development is in schemes for progressive compression that produce data segments specially tailored to "regions of interest" (ROIs) identified in the images. Our approach, as a natural follow through to the ICER-3D development, extends ICER-ROI [39], the 2 dimensional region-of-interest version of the ICER image compressor, to hyperspectral data. It will be called ROI-ICER-3D. It uses the same priority map for all spectral bands (this was extended later by other JPL researchers to assign separate maps for each spectral band or group of bands).

ROI-ICER-3D takes as input both the raw image data and a data prioritization map. A data prioritization map (or priority map, for short) is an assignment of a priority number to each pixel of an image. In our implementation, a priority number is an integer, with higher numbers indicating higher priority. A difference of some number b between two priority numbers indicates that the higher priority pixel should be reconstructed to roughly b more bits of precision than the lower priority pixel. The

priority map is generated by identifying and classifying features of the source image that are of interest to the end-users of the data. A priority map might be based on information contained entirely within the image being compressed, or it might be based on additional information, e.g., from recognizing changes from an earlier acquisition of the same scene. To be most effective, the classification and prioritization algorithms should be tailored to the specific objectives of the collected data. Foe example, a geologist analyzing hyperspectral images would most likely consider any image areas corresponding to cloud cover useless, whereas a meteorologist may be of the opposite opinion. Both scientists would probably give low priority to image areas corresponding to visible ocean surface, but an oceanographer may think otherwise.

The hyperspectral dataset and priority map are transformed using a wavelet transform. Priorities are accommodated by left-shifting (scaling by powers of 2) wavelet-transformed pixels according to their corresponding priorities. Output compressed data form a progressively coded “chain”. The chain consists of successive bit planes of priority-scaled levels of wavelet decomposition. Truncation of chains at different points determines the compression rate-distortion tradeoff. Due to the 12 to 16 bit nature of hyperspectral data and its DWT transform, a scheme of *virtual shifting* was designed to avoid expansion of memory requirements.

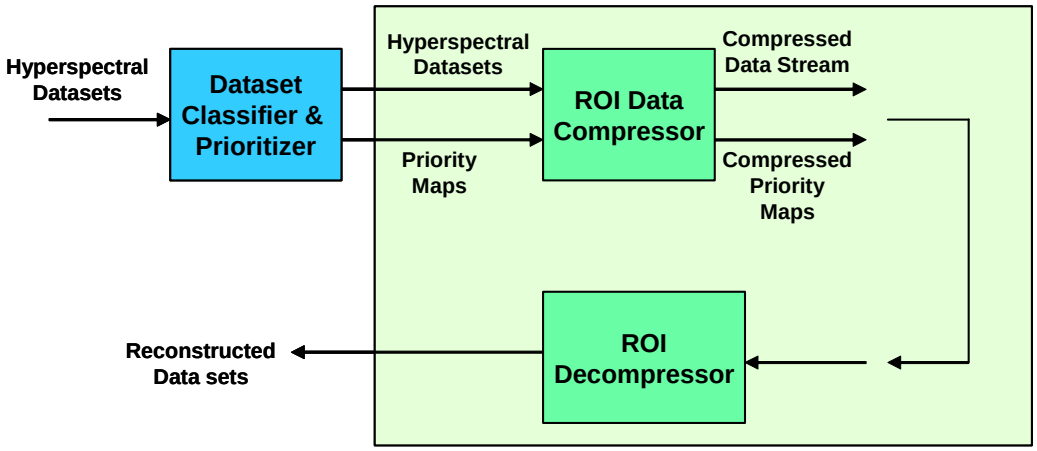


Figure 4.13: Region of Interest (ROI) Hyperspectral Data Compression

"Virtual" ROI scaling rather than actual ROI scaling in ROI-ICER-3D avoids expansion of dynamic range due to actual scaling, reduces memory requirements and improves compression performance. As shown in Figure 4.4, assigning an ROI priority scale of 2 (i.e. left shift high priority pixels by 2) adds two bit planes to the data, b_8 and b_9, when the old actual scaling scheme is used. The grey shaded zeros represent the extra bits that will be scanned during the bit plane encoding, affecting compression effectiveness as well requiring as much as twice the memory needed to hold the scaled data. In "virtual scaling", no new bit planes are introduced, the algorithm reads the priority map to determine if a pixel has a higher priority and scans first the bit planes of the high priority pixels skipping all the ones with lower or no priority assignment. The scanning method shown on the right of Figure 4.4 scans in its first path the b_7 bits of pixels 3 and 4, and then proceeds to scan b_6 bits of the same pixels. b_7 bits of the rest of the pixels get scanned in the third bit plane pass.

In addition, scanning of pixels 3 and 4 is shifted by 2, eliminating the need to scan zeros for the lower bit planes as was the case for the actual scaling.

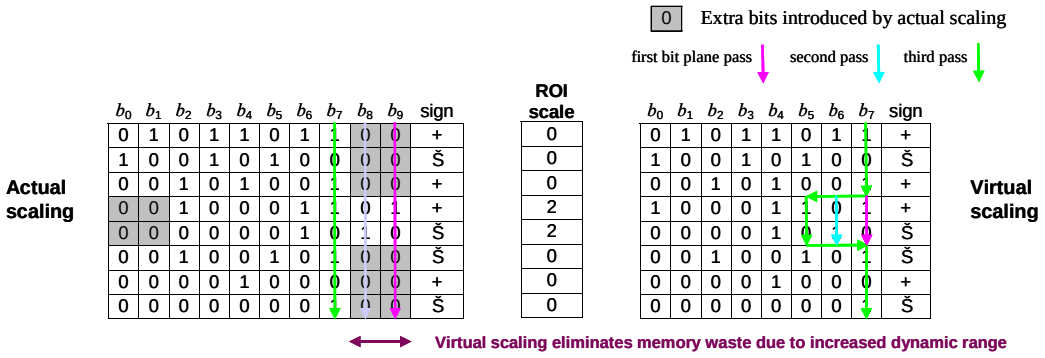


Figure 4.14: Virtual Scaling for ROI-ICER-3D

The benefits of "virtual scaling", which required major software redesign, are reduction of memory requirements (more desirable for hardware implementations as well as software) and enhancement of compression effectiveness. Table 4.3 demonstrates the compression performance improvements due to eliminating the extra bits introduced by actual scaling.

Table 4.3: Improvement for lossless coding comparing virtual and actual scaling

ROI scaling factor	Actual scaling	Virtual scaling
(bit planes)	Rate (bits/pixel/band)	Rate (bits/pixel/band)
0	5.00	5.00
1	5.30	5.07
2	5.56	5.10
3	5.79	5.11
4	6.02	5.10

Figures 4.15 and 4.16 show an example of compression using ROI-ICER-3D and the associated rate distortion curves for an AVIRIS image [8].

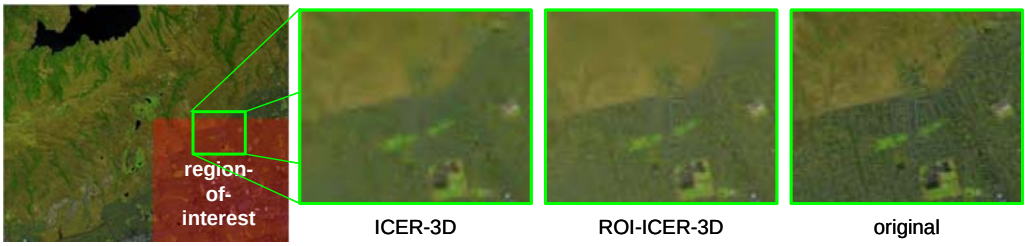


Figure 4.15: Example of ROI-ICER-3D compression of hyperspectral data set

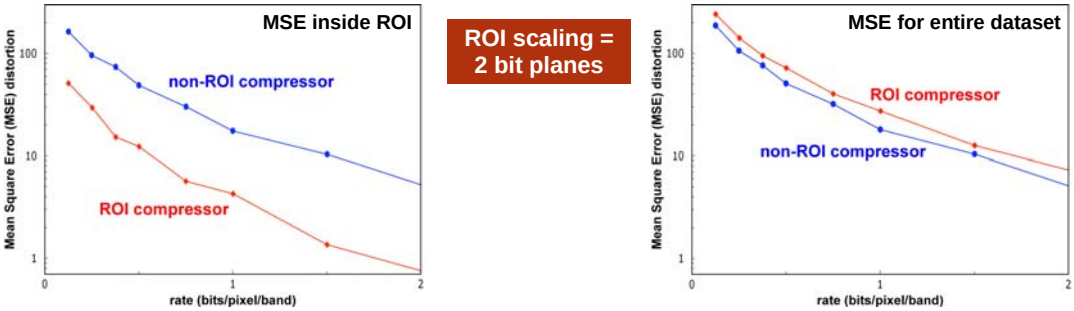


Figure 4.16: Performance comparisons on AVIRIS test image ROI-ICER-3D vs. (non-ROI) ICER-3D

4.4.2 Classifications and Signature Extractions

In many hyperspectral applications, classification and signature extraction are the end result. Classification accuracy for image cubes reconstructed after being compressed with ICER-3D-HW was tested on AVIRIS data sets and demonstrated completely successful classification down to .4bits/pixel (with minimum of 10

classes). An example of a signature extraction before and after compression is shown in Figure 4.17.

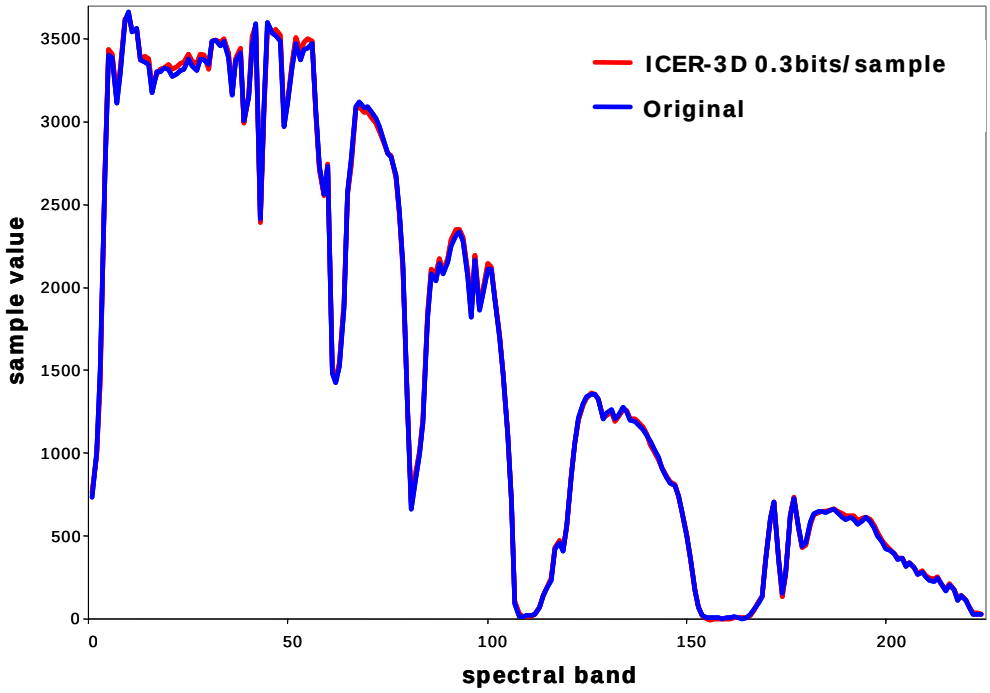
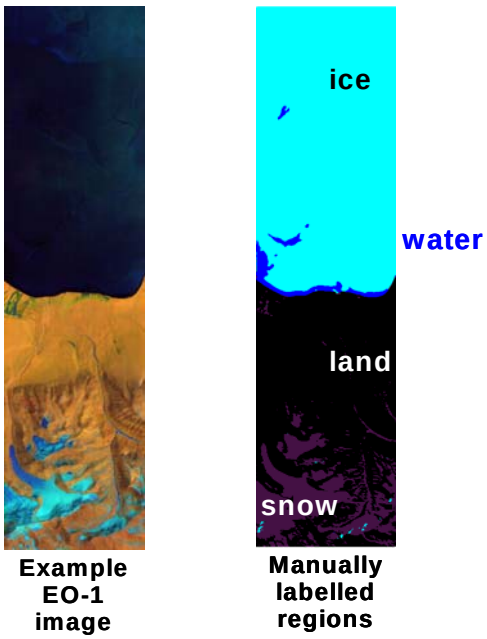


Figure 4.17: Example of Spectral line – Original and Reconstructed after Compression

We also tested the classification accuracy with an EO1 Hyperion test image. The classification algorithm separates the data into four classes: ice, water, land, and snow, and uses the support vector machine (SVM) pixel based classifier [30][35]. Figures 4.18 demonstrates compression down to 0.01bit/pixel with ICER-3D-HW, without significant degradation in classification accuracy.



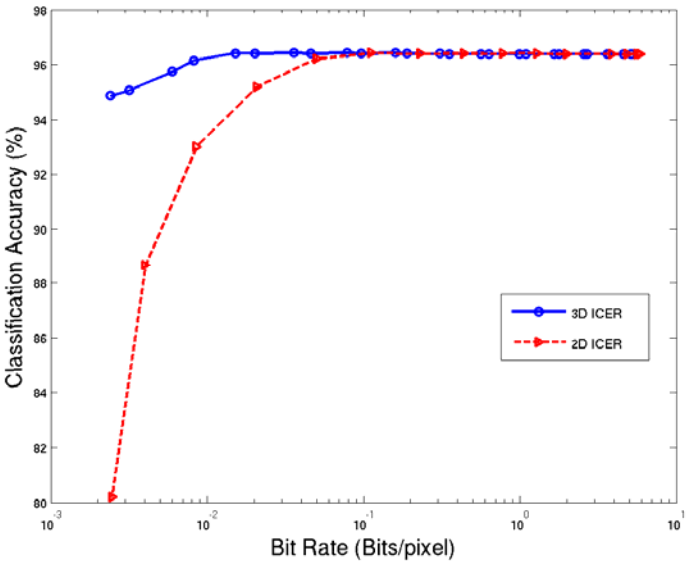


Figure 4.18: Example of lossy ICER-3D-HW performance in classification

4.5 Conclusions

In this chapter we presented a compression-effective, low-complexity 3D compression approach for hyperspectral imagers and sounders, suitable for on-board hardware implementation. Our approach is based on the reversible DWT transform, and extends a state-of-the-art 2D image compressor, ICER, to 3D data sets. We presented implementations based on the progressive 3D DWT. We looked into issues related to extending a 2D DWT approach to 3D DWT and porting the design to hardware. We also looked into 3D DWT limitations, such as the "spectral ringing artifacts" and provided a practical solution to mitigate the problem and provide better compression results at low bit rates. We presented an algorithm for region of interest (ROI) hyperspectral data compression that utilizes "virtual scaling" which has lower storage requirements and provides better compression effectiveness than standard scaling techniques. We presented compression results of test images from AVIRIS and compared them to other state-of-the art compression techniques. Metrics in terms of MSE and classification accuracy were addressed and test results were provided.

Chapter 5

Hyperspectral Data Compression on Reconfigurable Platforms

5.1 Introduction

Current NASA hyperspectral instruments either avoid compression or make use of only limited lossless image compression techniques during transmission. For example, the current state-of-the-practice is to use the Universal Source Encoder for Space (USES) chip [52]. USES implements the standard lossless CCSDS, which is based on the Rice algorithm, and has a multispectral mode, extending its operation to 3D data sets. The USES chip performance, as was shown in Chapter 4, has low compression effectiveness as compared to other existing techniques and lacks the flexibility be efficiently tailored to specific instruments needs. The main reasons for such practice by NASA are: the limited downlink bandwidth, the need to reduce the risk of corrupting the data-stream needed for accurate science processing, and the lack of a viable on-board platform to perform significant image processing and compression. Future instruments with more sensors and much larger number of spectral bands will collect enormous volumes of data that will far outstrip the current ability to transmit it back to Earth (data rates for some instruments can go to several hundreds of Gbits/sec [17][45][42]). This gives rise to the need for efficient on-board

hyperspectral data compression. Software solutions have limited throughput performance and are power hungry. Dedicated hardware solutions are highly desirable, taking load off the main processor while providing a power efficient solution at the same time. VLSI implementations are power and area efficient, but they lack flexibility for post-launch modifications and repair, they are not scalable and cannot be configured to efficiently match specific mission needs and requirements. FPGAs are programmable and offer a low cost and flexible solution compared to traditional ASICs.

While the benefits of FPGAs in general are significant, as was briefly discussed earlier in this thesis, the new capabilities of recent FPGAs offer an important new opportunity for achieving high performance. For example, the Xilinx Virtex II Pro, [113] with embedded Power PC processors, can operate at clock speeds up to 300 MHz, has multiple high performance serial interconnects and an extensive array of reconfigurable logic.

Fry and Hauck presented an FPGA implementation of the 2D SPIHT for hyperspectral data compression [43]. The implementation, due to its 2D nature, does not take advantage of the spectral correlations in the data. While SPIHT offers options for lossless compression by using reversible integer filters, this specific FPGA implementation was designed for lossy data compression and it targets a

prototype board with 3 FPGAs, one for the DWT and two for bit plane and entropy encoders, which results in high power and mass. An enhanced version of the 2D SPIHT algorithm, which uses band ordering and spectral predictive coding, was presented by Miguel *et. al.* [82] as a candidate for FPGA implementation. Miguel's implementation uses the 2D SPIHT FPGA compressor developed by Fry and Hauck as the base implementation, and extends it for the band ordering and prediction to be implemented in a separate, fourth, FPGA. While this proposed implementation yields improved compression efficiency due the interband prediction scheme, it comes at high power and mass.

As is the case for most efficient compressors, software implementation of the ICER-3D-HW compressor, described in the previous chapter, suffers from real-time processing difficulties. In this chapter we present an efficient embedded and scalable architecture for the ICER-3D-HW compressor, which we prototyped and implemented in the Xilinx Virtex II pro FPGA platform. The implementation takes advantage of the FPGA embedded PowerPC core and the on-chip bus architecture. Such platforms allow efficient partitioning of the algorithm into software and hardware modules to take full advantage of the available hardware resources and provide a system on a chip (SoC) solution for the hyperspectral data compression problem. Contrary to the two implementations of SPIHT mentioned earlier, our

implementation aimed for a single chip solution that can be readily ported to any instrument hardware platform.

In this chapter, we also present a methodology for a scalable embedded FPGA based implementation for a complex 3D compression system. We extended the wavelet transform methodology presented in Chapter 3 to hybrid Hardware/Software SoC FPGA implementations, addressing issues of SW/HW partitioning of algorithm modules, scalability of design, and trade-offs to meet practical considerations constraints.

This rest of this chapter is organized as follows. Section 5.2 details our system implementation methodology. Section 5.3 describes the ICER-3D-HW SoC FPGA implementation details and performance. Section 5.4 presents our summary and conclusions.

5.2 Implementation Methodology for a Scalable Embedded Hyperspectral Data Compression Architecture

Our methodology is depicted in the flow chart shown in Figure 5.1. We extend the 2D wavelet transform methodology developed in the first part of this thesis to hybrid Hardware/Software SoC FPGA implementations. As in the case of the 2D DWT methodology, several steps can be considered generic in terms of hardware design.

In addition to the steps that address the limitations and design choices listed for the 2D DWT methodology, our SoC methodology addresses the issue of SW/HW portioning of algorithm modules through a process of performing software profiling to identify appropriate candidates for hardware acceleration. Dynamic range expansion studies for the DWT are also performed to identify and select a suitable choice for the DWT filter pair. Finally, scalability of design, and trade-offs to meet practical considerations constraints, are performed to complete the design

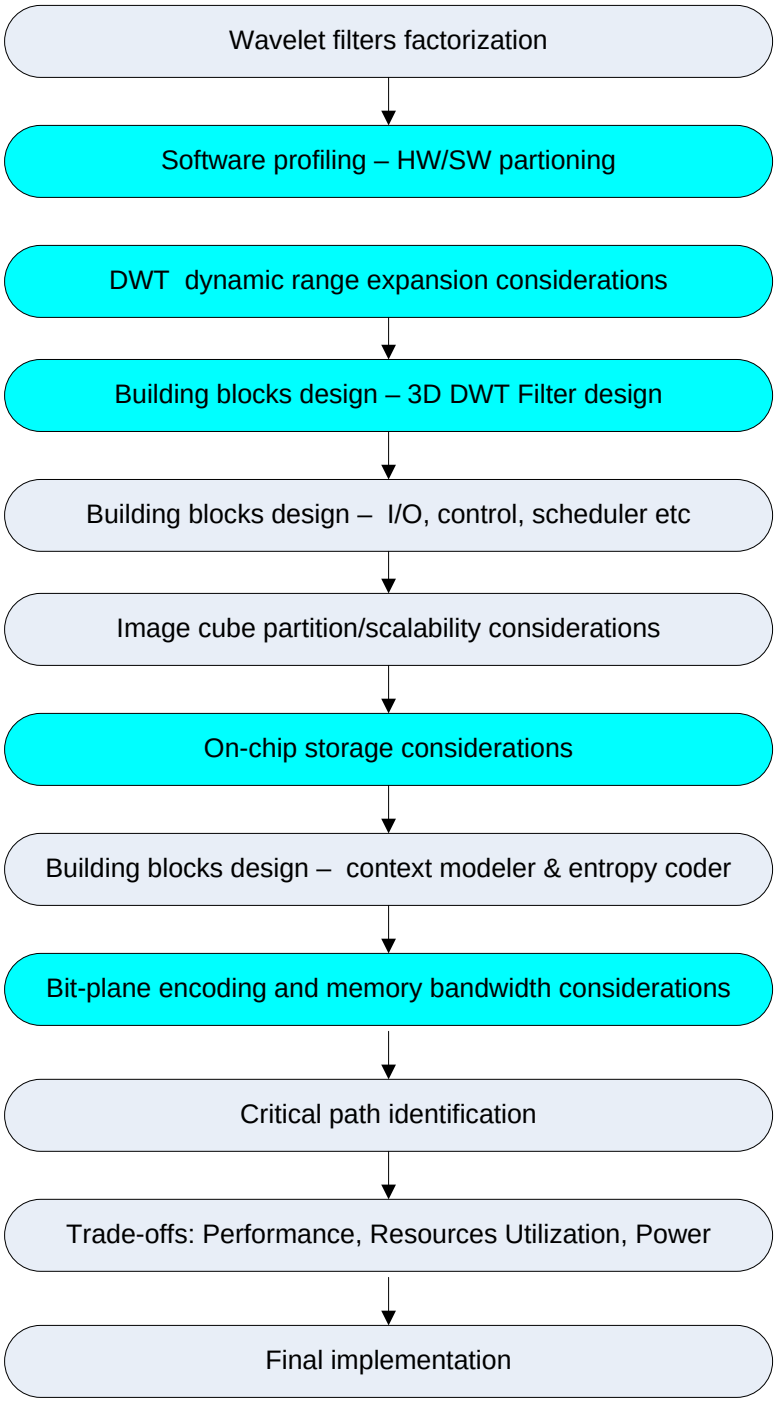


Figure 5.1: Implementation Methodology Flow Chart for the SoC FPGA implementation

5.2.1 Software Profiling and HW/SW Partitioning

The ICER-3D-HW c-code was profiled for software estimation and subsequent HW/SW portioning. Figure 5.2 shows that the system can be partitioned into 4 main blocks: 3D DWT, segmentation and conversion module, context modeler, and entropy coder. From Figure 5.3 it is apparent that the most time consuming blocks are the 3D DWT and the context modeler. Therefore the 3D DWT and the context modeler modules are the primary candidates for hardware implementation due to their computational complexity, while the other blocks can reside in software on the PPC processor. For a scalable design that may have more than one module performing the DWT and the context modeling, a hardware implementation of the entropy coder may be needed to maintain the desired throughput.

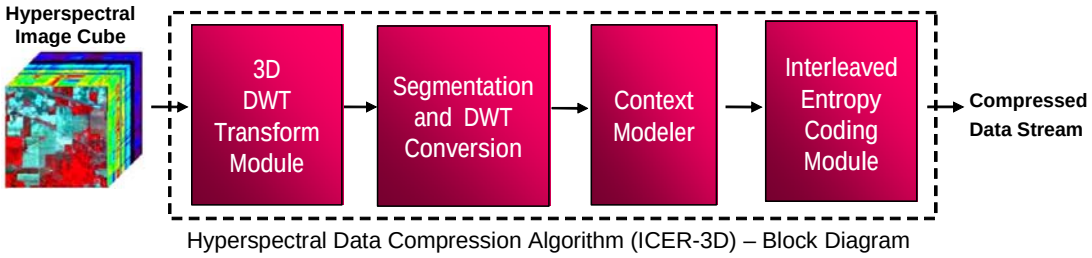


Figure 5.2: ICER-3D-HW Compressor-Block Diagram

Our approach for the full implementation was incremental. Initial candidate modules, such as the 3D DWT, will be implemented on the FPGA fabric as individual cores (IP cores or intellectual property), while the rest of the modules will run on the PowerPC. The PPC will also act as the global controller, managing the overall

operation of the compression system, including executing the top-level control and processing functions as well as scheduling, supervising and monitoring the processing activities and managing internal and external data transfers. New hardware modules were added in the form of hardware cores attached to the system bus, with the corresponding functionalities removed from the PPC software tasks.

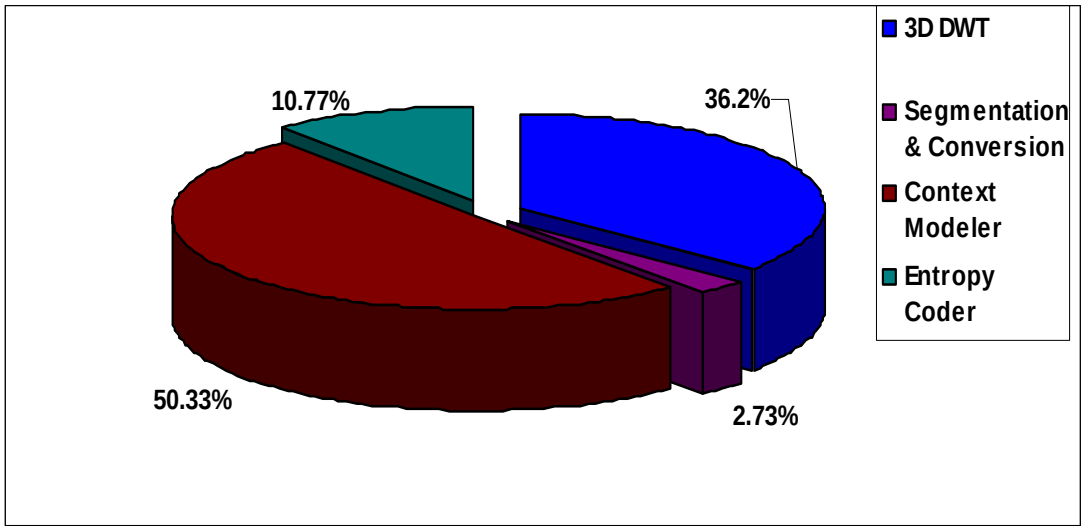


Figure 5.3: Software Profiling of ICER-3D-HW

5.2.2 Dynamic Range Expansion for DWT Transformed Data

While dynamic range expansion analysis was detailed in Chapter 4, such analysis is listed here as part of the implementation methodology since it plays an important part in matching the design to the available hardware resources.

Dynamic range analysis expansion, as detailed in section 4.2.3.1, showed that 16-bit words are sufficient to store the coefficients produced by applying a 3D DWT decomposition, using filter A (the (2,6) DWT filter pair) [63] to 12-bit data (such as uncalibrated AVIRIS data). However, other filter choices in ICER-3D-HW software may produce DWT coefficients that cannot be stored in 16-bit words following 3D wavelet decomposition. Hence, the DWT filters we used in this hardware implementation were the (2,6) filter pair.

5.2.3 Three Dimensional DWT Hardware Architecture

Similar to the 2D DWT implementations, the filter design methodology has to choose among different well known architectures or produce a custom architecture to match the given constraints. While the cascaded architecture was the ideal choice for the 2D case, the massive buffering requirements for a 3D DWT cascaded design makes the choice impractical.

For an image cube of width W, length of L lines, and λ spectral bands, and DWT filters of length less than or equal to F_l , the cascaded 3D pipelined architecture for a pushbroom sensor (BIP or BIL data format), require internal storage of (measured in pixels to store row-column DWT transformed planes) :

$$F_l * W * \lambda \qquad (5.1)$$

For AVIRIS data sets of dimensions 224x614x512, and the (2,6) DWT filter pair, we need 1.65 Mbytes of storage for each DWT processing unit. A typical large FPGA usually has about 1M Byte of on-chip RAM (BRAM). Hence, the practical choice is a hybrid design comprising a two phase architecture. As shown in Figure 5.4, phase 1 consists of a cascaded row-column DWT decomposition, followed by a folded architecture for the 3rd dimension DWT in phase 2.

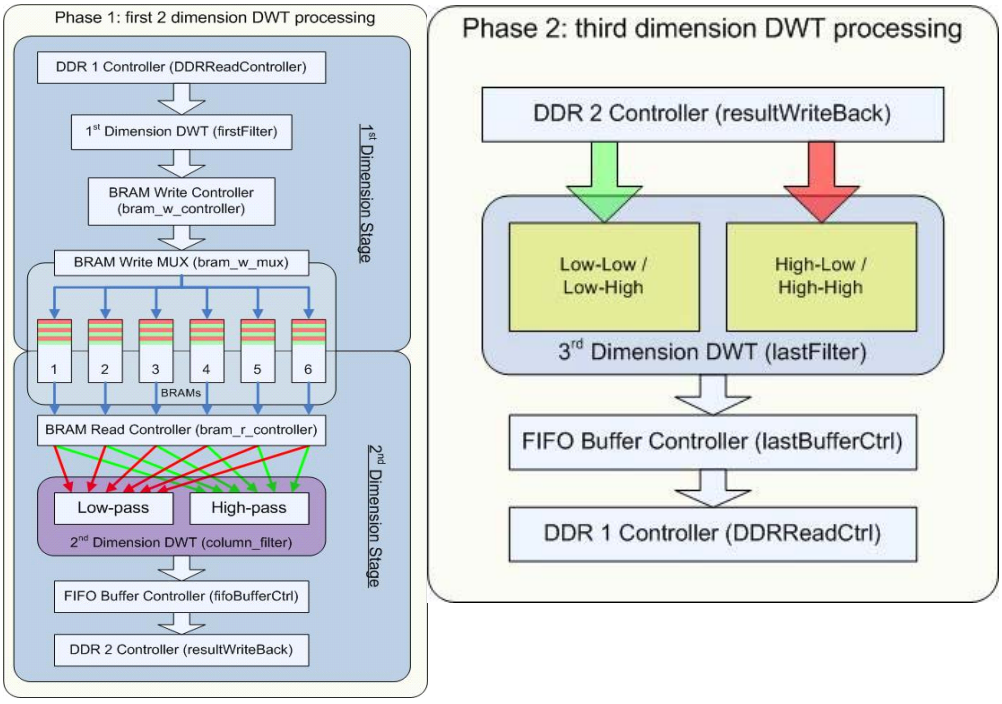


Figure 5.4: Block Diagram of the 3D DWT

5.2.4 On-Chip Storage Calculations for the 3D DWT

For a DWT filter pair and an image cube of $N \times L \times \lambda$ pixels, and denoting

F_l – the length of the longest filter

J - DWT decomposition levels

S - Number of DWT line modules

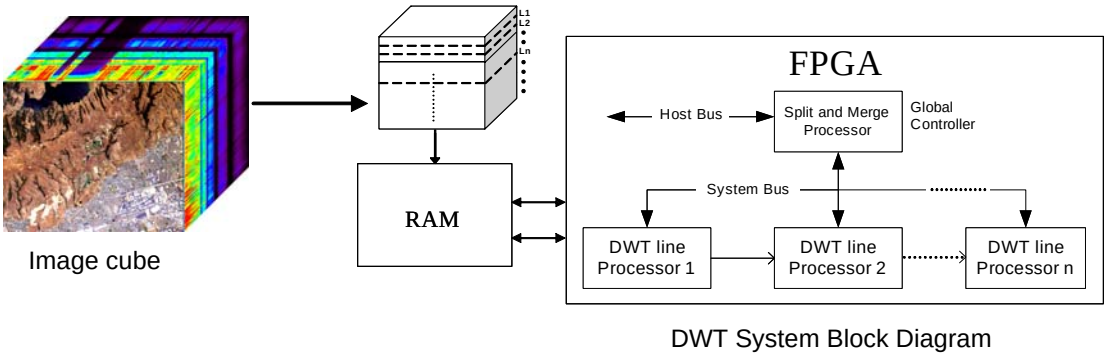


Figure 5.5: Three Dimensional DWT Hardware Platform

Consider the stripe-parallel design shown in Figure 5.5. After the completion of 1 level DWT decomposition, the number of transitional boundary states generated at the first boundary of stripe 1 and stripe 2 is:

from stripe 1:

$$m_{B1} = \left\lceil \tfrac{1}{2} F_l \right\rceil * N * \lambda \qquad (5.2)$$

and from stripe 2

$$m_{B2} = \left\lfloor \tfrac{1}{2} F_l \right\rfloor * N * \lambda \qquad (5.3)$$

where memory is measured here in number of pixels, $\lceil\ \rceil$ and $\lfloor\ \rfloor$ are the ceiling and the floor operators to accommodate odd length DWT filters at the stripe boundaries. This results from the absence of image data along the boundaries of B1 and B2 required to complete the filtering operations. After the completion of 2 decomposition levels additional transitional boundary states are generated at the same boundary:

from stripe 1:

$$m_{B1} = \left\lceil \tfrac{1}{2} F_l \right\rceil * N * \lambda * \tfrac{1}{2} \qquad (5.4)$$

and from stripe 2

$$m_{B2} = \left\lfloor \tfrac{1}{2} F_l \right\rfloor * N * \lambda * \tfrac{1}{2} \qquad (5.5)$$

Hence the memory required to hold transitional boundary states for each boundary is:

$$m_1 = F_l \sum_{i=0}^{J-1} N * \lambda * \left(\frac{1}{2}\right)^i \qquad (5.6)$$

and the total memory (measured in pixels) required to hold transitional boundary states for the overlap-state algorithm for all the boundary data is:

$$m_{total} = F_l \left[\sum_{i=0}^{J-1} N * \lambda * \left(\frac{1}{2}\right)^i \right] * (S-1) \qquad (5.7)$$

For the overlap-save algorithm, we only save boundary states once and exchange at every level of DWT decomposition, hence the internal memory required is:

$$m_{total} = F_l * N * \lambda * (S-1) \quad . \quad (5.8)$$

For a typical AVIRIS image, of dimensions 614x512x224, J=3, F_l=6, and S=4, we need about 8.5Mbytes of internal memory for the overlap-state, and about 4.8 Mbytes for the overlap-save, making both techniques impractical.

The practical design choice for this implementation is the straightforward overlapping architecture, which requires no internal storage for boundary states, even though it requires additional DWT computations when compared to other architectures.

For the cascaded DWT design of phase 1 shown earlier, the first 2 DWT dimensions, the analysis for the storage requirements is the same as that detailed in section 3.3.4, yielding the following buffering storage requirements:

$$m_{FIFO-total} = F_l * N * S \quad (5.9)$$

For the same typical AVIRIS image used for the earlier illustration, the total required on-chip RAM (BRAM) for phase 1 measured in pixels is: 6*614*3 = 22Kbytes

(note that there is no need to account for dynamic expansion in DWT domain since we used the (2,6) filter pair).

5.2.5 Bit-Plane Encoding and Memory Bandwidth Considerations

The context modeler and entropy coder operate on bit planes of segments of the 3D DWT transform (as was detailed in section 4.2.2.2) [63]. This implies a memory bandwidth of 16 read and write operations to compute the contexts and encode one single pixel. This problem is similar to what researchers encountered in implementing the JPEG2000 EBCOT encoder. Several approaches based on massive buffering of bit planes were proposed [77][33][34] and one could choose to implement such a choice for the design of our context modeler. An alternative approach, however, is to utilize a priority and encoding scheme to format the bit planes post the 3D DWT transform and store them in external memory (RAM), transposed and localized, readily available for the context modeler and entropy coder stage, as will be explained in section 5.3.2.

5.3 ICER-3D-HW Implementation and Performance

We applied our methodology to the ICER-3D-HW hyperspectral compression algorithm to produce the implementation detailed in this section.

5.3.1 Implementation of the 3D (2,6) DWT

The FPGA implementation of ICER-3D-HW reflects the Mallat decomposition of the 3D DWT, modified to compute the mean subtraction of spatially low-pass filtered subbands as detailed in section 4.2.2.3 to mitigate the spectral ringing artifacts. The first module designed and implemented was the 3D DWT. In addition to performing the 3D DWT, the last stage of this module calculates and subtracts the means of low pass filtered subbands prior to writing their data to the external memory, and hence saves computational costs. We designed cascaded line-based wavelet transform modules, which allow the wavelet transform in the 3D DWT case to be computed as the lines of the image data cube arrive, rather than waiting for an entire frame of data, thus accommodating pushbroom sensors. The parallel DWT modules operate on slices of the image cube using the overlapping scheme detailed in section 3.2.3. Figure 5.5 illustrates the parallel DWT system. The 3D DWT implementation provided 16:1 speed-up versus software and increased to 30:1 with two modules of the DWT running in parallel, as benchmarked on our FPGA prototype board.

5.3.2 Implementation of Context Modeler and Entropy Coder

The context modeler and the interleaved entropy coder were designed for throughput performance, with the design of priority–based data formatting and localization techniques that transpose bit-planes post the 3D DWT decompositions, and store them in memory in contiguous form, to be readily available for the context modeler and the indexed bit-plane encoding. At the completion of the 3D DWT transform, and according to the band indexing and the bit plan priority encoding scheme detailed in section 4.2.2.2, bit planes are read across blocks of 16 DWT coefficients and transposed in-place. The transposed data is written to external memory in a contiguous fashion in preparation for the context modeler stage. For example, in Figure 5.6, for the DWT segment planes shown, *I* and *J*, let segment *I* have a higher index than segment *J*, and b_n denote bit plan *n*. Let the bit plane priority values be sorted from higher to lower as I_{bn}, I_{bn-1}, J_{bn}, ..., I_{bn-2}, J_{bn-1},J_{bn-2}, ... The formatting scheme transposes the bit planes and stores them in external memory in the order of the priority values as shown. This design accelerates the encoding scheme by a factor of more than 10:1.

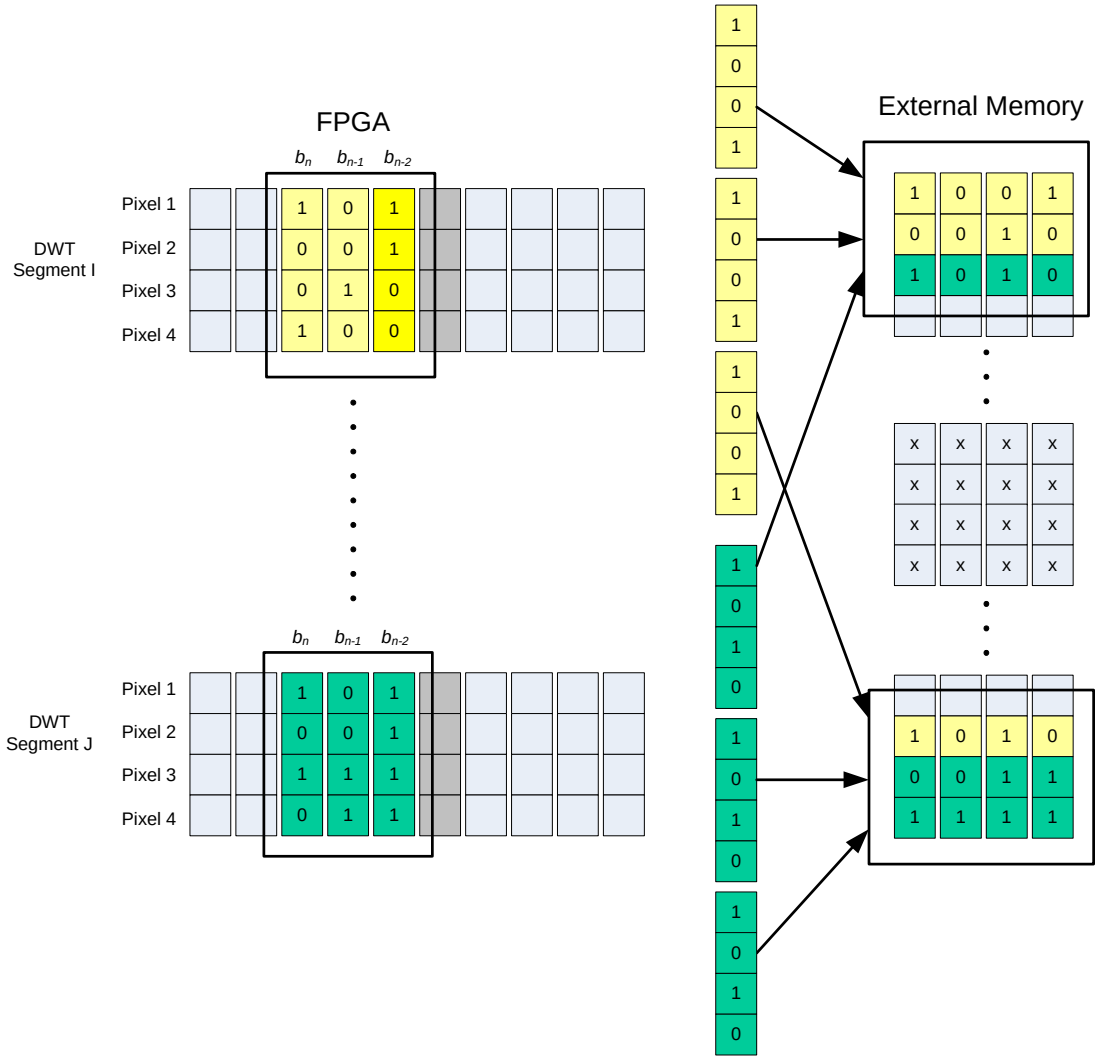


Figure 5.6: Bit-Plane Formatting and Storage

The context modeler itself was designed to be a pipeline that computes contexts of multiple bits of the same bit plane from different pixels. If the required compressed quota is reached, the context modeler (and entropy coder), conclude the encoding process. The entropy coder utilizes look-up tables stored in on-chip BRAM. Speed-up of more than 10X was obtained vs. software implementation for this module. The

design is scalable and allows the use of multiple versions of the module simultaneously. Figure 5.7 shows the design and data flow for the context modeler.

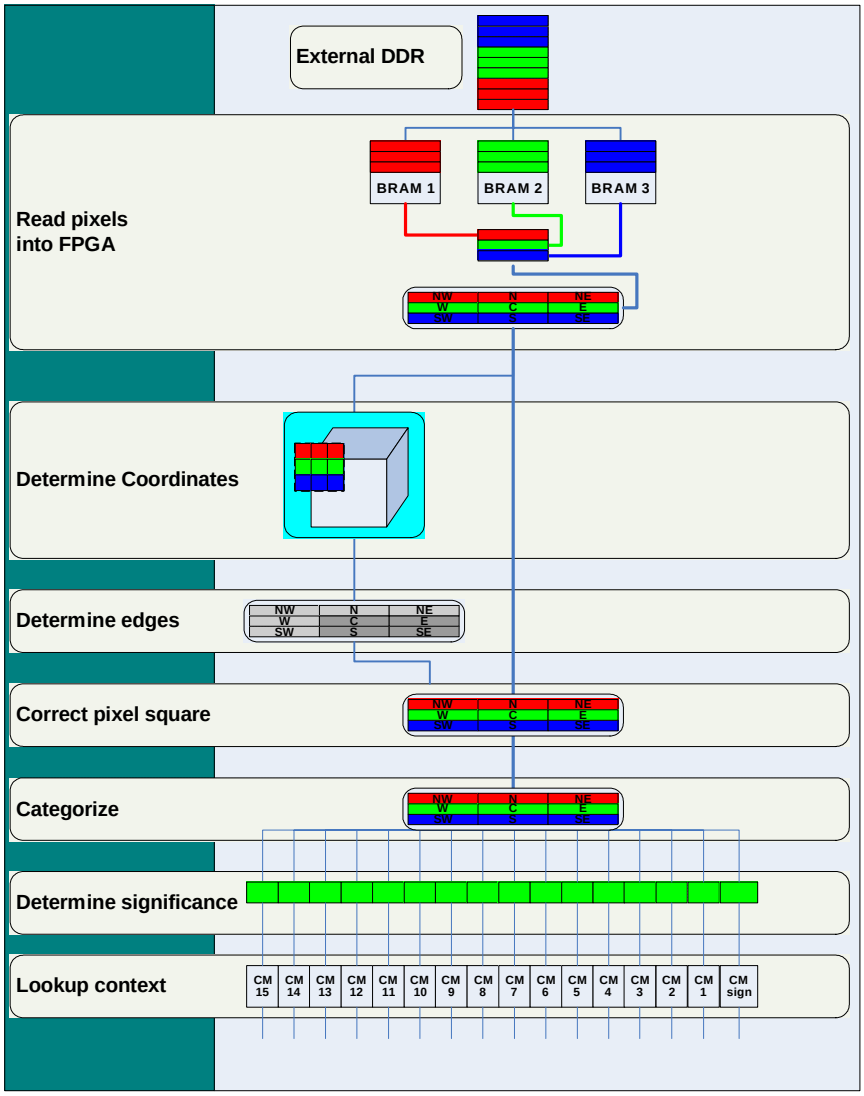


Figure 5.7: Context Modeler -FPGA Design - Pipeline and Data Flow

The diagram in Figure 5.8 illustrates the parallel architecture of the coding modules and the data flow. Bit planes are read from different segment blocks residing in RAM. Their contexts are computed independently and fed through to entropy coding

modules to generate encoded bit planes. The compressed bit planes are interleaved and written back to RAM. The modules are stand-alone units that utilize an efficient DMA to access the external DDR memory. Modules can also operate in parallel on different segments of the wavelet-transformed image.

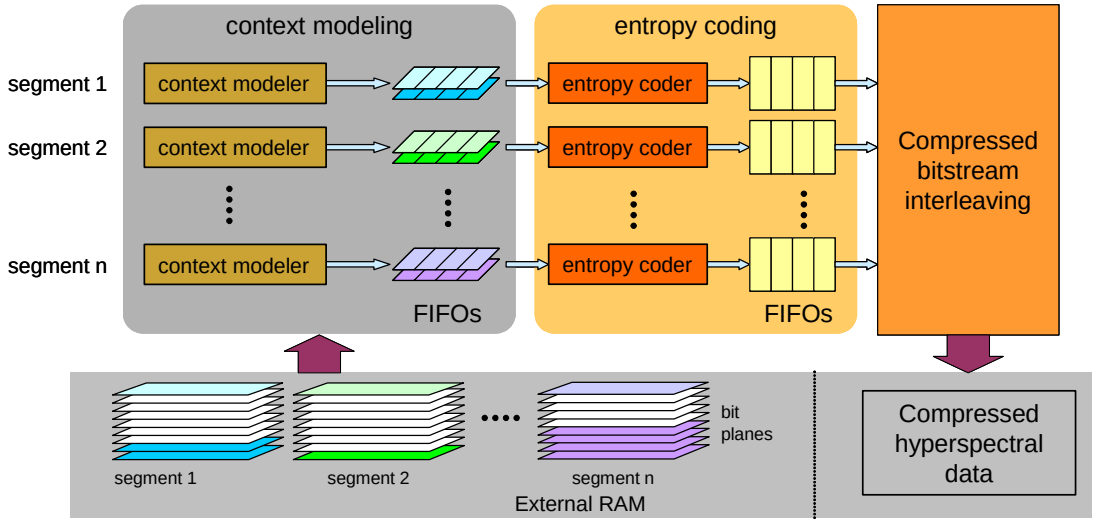


Figure 5.8: Parallel Design for the Context Modeler and Entropy Coder

5.3.3 Data Flow and Memory Management

Figure 5.9 shows the system data flow and the external memory bandwidth. Output of the 1st dimension DWT is pipelined into the 2nd dimension, eliminating external memory access.

Means computations and DWT coefficients conversion are performed as part of step 3. Segment blocks of DWT coefficients are formatted according to the priority scheme described earlier for faster memory access by the context modeler and

entropy coder. This provides a total memory bandwidth of no more than 6 total read and write operations per pixel.

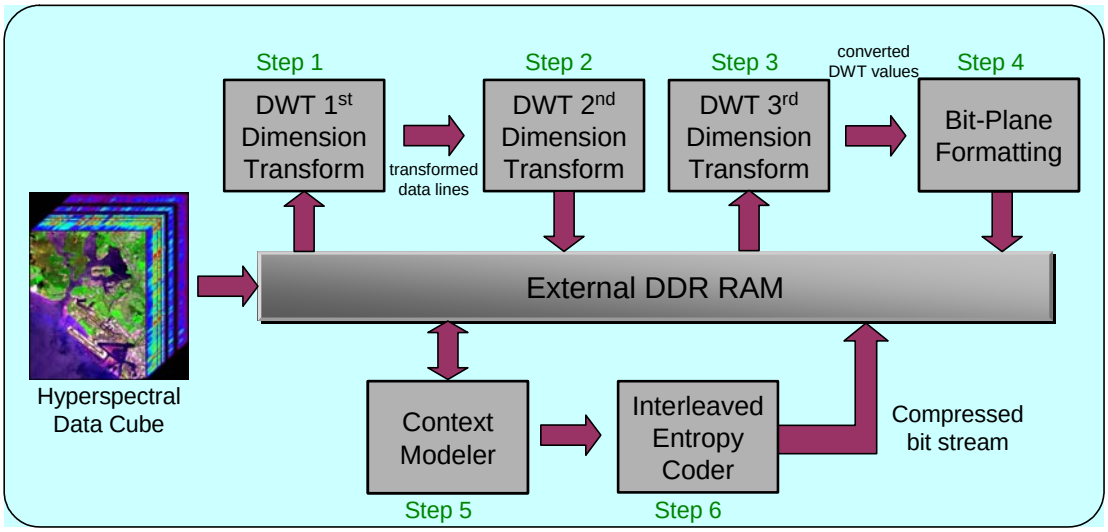


Figure 5.9: Hyperspectral Compressor – Data Flow

5.3.4 FPGA Prototype and Performance

The implementation of ICER-3D-HW was designed by applying the methodology described earlier [9][10][11]. It was then coded in VHDL and ported to a DINI Group PCI prototype board targeting the Xilinx Virtex II Pro XC2VP70 chip. The final SoC architecture is shown in Figure 5.10. The hardware development system was shown in Figure 2.9. With one copy of each module, we obtained a throughput of 4.5 Msample/sec for lossless compression running at a clock speed of 50 MHz (lossy compression performance is slightly faster since not all bit planes need to be compressed). When the implementation was scaled up to two copies of each of the

three main modules, 3D DWT, context modeler and entropy coder, running in parallel, the throughput increased to 8 Msample/sec. This throughput is more than an order of magnitude faster than the software code, which runs at about 610 Ksamples/sec on a Pentium Centrino 1.6MHz processor. A slow memory interface specific to the prototype board resulted in a substantial reduction of memory bandwidth. Simulations with an improved memory interface design show substantially increased throughput to 1 sample/clock cycle (i.e. 50 Msample/sec for the current 50 Mhz clock design), resulting in 2 orders of magnitude speed-up vs. the software implementation. The device utilization of table 5.1 shows that the full implementation of the compressor occupies less than 61% of FPGA resources with 2 copies of each module running in parallel. Power consumption for this implementation is 6.5 Watts with one copy and increases to 7.5 Watts with two copies of each of the hardware modules.

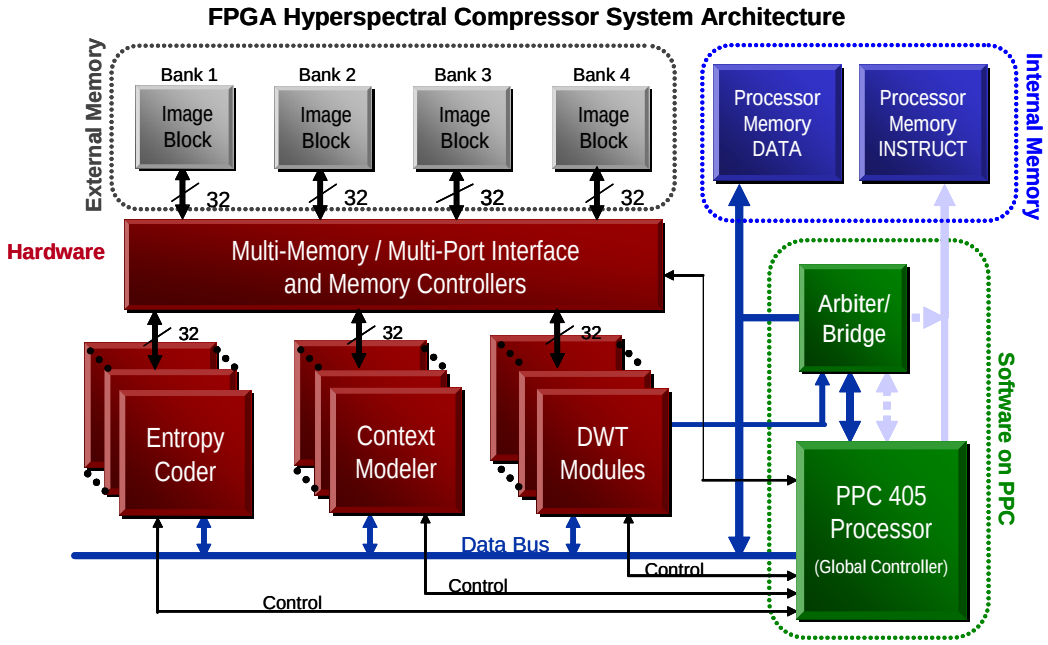


Figure 5.10: System on Chip FPGA implementation Hyperspectral Data Compressor

Table 5.1: ICER-3D-HW on Virtex II Pro XC2VP70 FPGA - Resources Utilization

	One 3D DWT Module	Full System with Two 3D DWT Modules	Full System with 2 Copies of Each Module
Number of Slices	5070 (15% of available resources)	13411 (41%)	20117 (61%)
Number of Slice Flip Flops	5232 (7%)	16032 (24%)	24048 (36%)
Number of 4 input LUTs	8183 (12%)	17505 (26%)	26257 (40%)
Number of BRAMS	26 (7%)	106 (32%)	158 (48%)
Number of 18x18 Multipliers	16 (4%)	32 (10%)	48 (15%)

5.4 Conclusions

In this chapter we presented an embedded and scalable implementation for the ICER-3D-HW compression algorithm in FPGAs. The approach uses a co-design platform (SW/HW) with architecture-dependent enhancements to improve performance. We addressed challenges in this design related to the intensive I/O of the algorithm, the 3D nature of the data and its volume. Solutions to these challenges were proposed by choosing efficient DWT architectures and a novel bit-plane priority-based data formatting and localization technique that provided more than 10x in throughput efficiency compared to standard techniques. Finally, we presented an extension to our FPGA implementation methodology described in Chapter 3 to a system on a chip (SoC) FPGA-based implementations.

Chapter 6

Conclusion and future work

We presented, in this thesis, a methodology for parallel implementation of the lifting DWT on FPGAs, and investigated and analyzed parallel and efficient hardware implementations targeting state-of-the-art FPGAs. We addressed practical considerations and various design choices and decisions at all design stages to achieve an efficient DWT implementation, subject to a given set of constraints and limitations. We presented a novel data transfer method that provides seamless handling of boundary and transitional states associated with parallel implementations, and demonstrated our methodology with an implementation example for the (9,7) DWT.

We presented an efficient low-complexity 3D on-board compression approach for hyperspectal images and sounders. We presented implementations based on the progressive 3D DWT. We addressed issues related to adapting an efficient 2D DWT approach to 3D DWT and porting the design to hardware implementations. We looked into 3D DWT limitations, such as the “spectral ringing artifacts” and provided practical solution to mitigate the problem and provide better compression results at low bit rates. We also presented an extension of the algorithm for region of interest (ROI) hyperspectral data compression, which utilized a “virtual scaling”

approach to improve compression efficiency and reduce memory requirements. We provided results and comparisons to other state-of-the art compression techniques.

We presented an embedded and scalable SoC implementation for the ICER-3D-HW compression algorithm on FPGAs. We addressed challenges related to the intensive I/O of the algorithm and the 3D nature of the data and its volume, and provided solutions to speed up the design. We extended our FPGA implementation methodology to a system on a chip (SoC) FPGA-based implementations.

Future work will concentrate on insertion of this new technology, especially the FPGA SoC implementation of the ICER-3D-HW, into future space-borne instruments. Missions such as FLORA and PPFT [17][45] are including compression in their initial concept designs and are good candidates to use our final product. Fault tolerant designs for the FPGA system may be required by such missions and will be investigated. Issues related to transient faults arising from single event upsets (SEUs) and mitigation techniques for FPGA based designs will be addressed. Figure 6.1 illustrates a proposed fault tolerant block diagram of the FPGA SoC compression system for future space missions. The green shaded areas in the figure indicate fault tolerant hardware, and ABFT is algorithm based fault tolerance [51][60][16].

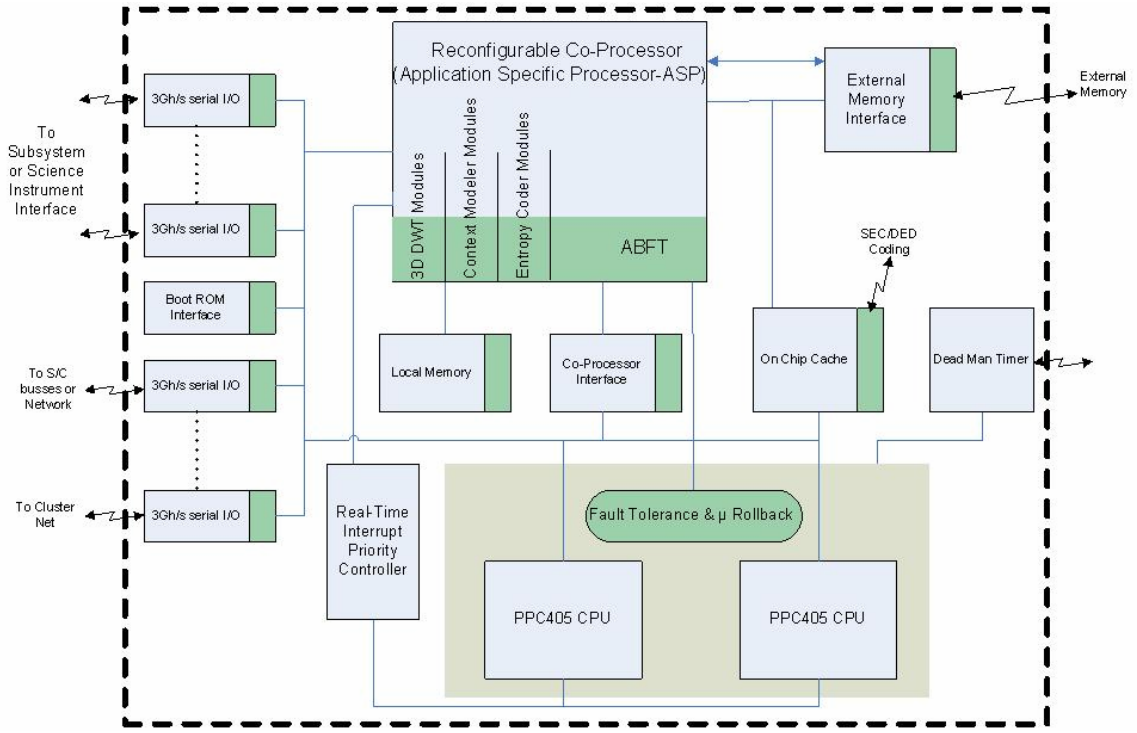


Figure 6.1: Fault Tolerant FPGA based Compression System

The new ICER-3D algorithm uses an optimized spectral context modeler for three dimensional data and produces better compression efficiency than ICER-3D-HW. Future work will also migrate the new spectral context models to the hardware platform and extend the implementation to include region of interest compression, i.e. ROI-ICER-3D.

Bibliography

[1] M. Adams and F. Kossentini, “Reversible Integer-to-Integer Wavelet Transforms for Image Compression: Performance Evaluation and Analysis,” IEEE Transactions on Image Processing, vol. 9, no. 7, pp. 1010–1024, June 2000.

[2] M. Adams and R. Ward, “ Wavelet Transform in the JPEG-2000 Standard”, Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada, Aug. 2001, vol. 1, pp. 160-163.

[3] N. Akansu and R. A. Haddad, “Muinresolution Signal Decomposition", Transforms, Subbands and Wavelets, New York, Academic, 1992.

[4] B. Aiazzi, L. Alparone, A. Barducci, S. Baronti and I. Pippi. “Information-Theoretic Assessment of Sampled Hyperspectral Imagers”, IEEE Transactions on Geoscience and Remote Sensing vol. 39, No. 7, 2001.

[5] Amphion, CS6210 Discrete Wavelet Transform.

[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, “Image coding using the wavelet transform", IEEE Trans. Image Proc. 1, pp. 205-220, Dec. 1992.

[7] N. Aranki, A. Moopen, and R. Tawel, “Parallel FPGA Implementation of the Split and Merge Discrete Wavelet Transform,” Proc. of 12th International Conf., FPL2002, pp 740-749, Montpellier, France, Sep., 2002.

[8] N. Aranki, M. Klimesh, H. Xie, S. Dolinar, “Region-of-Interest (ROI) Compression of Hyperspectral Data”, JPL’s Interplanetary Network Directorate, Annual Review Report, Sep., 2004.

[9] N. Aranki, A. Kiely, M. Klimesh, H. Xie, “Advanced Hyperspectral Data Compression,” Proc. 2005 AVIRIS Earth Science and Applications Workshop, Pasadena, CA, May 24-27, 2005.

[10] N. Aranki, R. Some, J. Namkung, and C. Villalpando, “Hyperspectral Data Compression on Reconfigurable Platforms for Space”, to be submitted to Military and Aerospace FPGA and Applications (MAFA), to be held in Florida, Nov. 2007.

[11] N. Aranki, J. Namkung, C. Villalpando, A. Kiely, M. Klimesh, H. Xie “Hyperspectral Data Compression on Reconfigurable Platforms”, NASA Tech Briefs, 2007.

[12] N. Aranki, J. Namkung, C. Villalpando, “Hyperspectral Data Compression: FPGA Platform Development”, JPL’s Interplanetary Network Directorate, Annual Review Report, Oct. 18, 2005.

[13] N. Aranki, J. Namkung, C. Villalpando, “Hyperspectral Data Compression: FPGA Development”, JPL’s Interplanetary Network Directorate, Annual Review Report, Sep. 17, 2004.

[14] N. Aranki, J. Namkung “Hyperspectral Data Compression: Algorithm and Hardware Development”, JPL’s Interplanetary Network Directorate, Annual Review Report, Sep. 10-12, 2003.

[15] N. Aranki, W. Jiang and A. Ortega, “FPGA-Based Parallel Implementation for the Lifting Discrete Wavelet Transform”, Parallel and Distributed Methods for Image Processing IV - SPIE's 45th Annual Meeting, San Diego, CA, July 2000.

[16] N. Aranki and R. Some, “Algorithm Based Fault Tolerance For FPGAs”, NASA Tech Briefs, 2006.

[17] G. Asner, S. Ungar, R. Green, and R. Knox, “FLORA: Leaping from AVIRIS to high-fidelity spaceborne imaging spectroscopy,” Proc. 2005 AVIRIS Earth Science and Applications Workshop, Pasadena, CA, May 24-27, 2005.

[18] G. Asner, “Airborne imaging spectroscopy across diverse ecosystems: The Carnegie program in Hawaii. Proc. 2005 AVIRIS Earth Science and Applications Workshop, Pasadena, CA, May 24-27, 2005.

[19] H. Aumann and L. Strow, “AIRS, the first hyperspectral infrared sounder for operational weather forecasting,” in Proceedings of IEEE Aerospace Conference New York, 2001, pp. 1683-1692.

[20] AVIRIS Free Data Website: http://aviris.jpl.nasa.gov/html/aviris.freedata.html

[21] A. Benkrid, D. Crookes, K. Benkrid, “Design and Implementation of Generic 2-D Biorthogonal Discrete Wavelet Transform on and FPGA,” IEEE Symposium on Field Programmable Custom Computing Machines, pp 1 – 9, April 2001.

[22] G. Beylkin, R. R. Coifman and V. Rokhlin, “Wavelets in Numerical Analysis” in Wavelets and Their Applications, New York, Jones and Bartlett, 1992, pp.181-210.

[23] W. Bicknell, C. Digenis, S. Forman and D. Lencioni, "EO-1 Advanced Land Imager," SPIE Conference on Earth Observing Systems IV, Denver, Colorado, Proc. SPIE, Vol. 3750, pp. 80-88, July 1999.

[24] M. Boliek, M. Gormish, E. Schwartz and A. Keith, “Next Generation Image Compression And Manipulation Using CREW”, Proceedings of International Conference on Image Processing, Vol 3, Oct 1997, pp. 567-570.

[25] N. Bradley, and C. Brislawn, “SPECTRUM analysis of multispectral imagery in conjunction with wavelet/KLT data compression”, IEEE, 1993 Conf. Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, Vol. 1, 1993, pp. 26-30

[26] B. Brower and A. Lan, and J. McCabe “Hyperspectral Lossless Compression,” Proc. SPIE, 3753 1999, pp.247-257.

[27] K. Carlson, and G. Asner. 2005. Spaceborne imaging spectroscopy of tropical forest properties in Hawaii. Proc. 2005 AVIRIS Earth Science and Applications Workshop, Pasadena, CA, May 24-27, 2005.

[28] Cassini-Huygens Website: http://saturn.jpl.nasa.gov/home/index.cfm.

[29] Cassini VIMS Website: http://wwwvims.lpl.arizona.edu.

[30] R. Castano, D. Mazzoni, N. Tang, R. Greeley, T. Doggett, B. Cichy, S. Chien, and, A. Davies, “Onboard Classifiers for Science Event Detection on a Remote Sensing Spacecraft”, Proceedings of the 12th ACM SIGKDD international conf. on Knowledge discovery and data mining, pp. 845 – 851, Philadelphia, PA, 2006.

[31] C. Chakrabarti, M. Vishwanath and R. M. Owens, "Architectures for wavelet transforms" in Proc. IEEE VLSI Signal Processing Workshop, 1993, pp.507-515.

[32] C. Chakrabarti, M. Vishwanath, “Efficient Realization of the Discrete and Continuous Wavelet Transforms: From Single Chip Implementations to Mappings in SIMD Array Computers,” IEEE Transactions on Signal Processing, Vol. 43, pp 759 – 771, March 1995.

[33] K. Chen, C. Lian, H. Chen and L. Chen, “Analysis and architecture design of EBCOT in JPEG2000”, Proceedings, IEEE International ISCAS-01, Vol.1, May 2001 pp 765-768.

[34] J. Chiang, Y. Lin and C. Hsieh, “Efficient pass-parallel architecture for EBCOT in JPEG2000”, Proceedings, IEEE International ISCAS-02, Vol.1, May 2002 pp 773-776.

[35] C. Cortez and V. Vapnik, “Support vector networks”, Machine Learning, Vol. 20, pp. 273 – 279, 1995.

[36] I. Daubechies and W. Sweldens, “Factoring wavelet transforms into lifting steps", J. Fourier Anal. Appl. 4 (3), pp. 247-269, 1998.

[37] D. Ding, Q. Dai, F. Yang and Y. Yin,“A new region-of-interest image compression method based on Wyner-Ziv coding”, Proceedings of the SPIE, Volume 5960, pp. 849-856, 2005.

[38] G. Ding, Q. Dai, F. Yang and W. Xu, “Distributed source coding theorem based region of interest image compression method”, Electronics Letters, Vol 41, Issue 22, 27 Oct. 2005, pp. 1215– 1217

[39] S. Dolinar, A, Ortega, R. Manduchi et al., “Region of Interest Data Compression with Prioritized Buffer Management (III),” Proceedings of NASA Earth Science Technology Conference 2003, College Park, MD, June 2003.

[40] P. Dragotti, G. Poggi, and A. Ragozini, “Compression of Multispectral Images by Three-Dimensional SPIHT Algorithm”, IEEE Transactions on Geoscience and Remote Sensing, vol.38, No. 1, pp. 416-428, Jan. 2000.

[41] EO1/ Hyperian Website: http://eo1.gsfc.nasa.gov/Technology/Hyperion.html

[42] EOS Data and Information System (EOSDIS) Website: http://terra.nasa.gov/Brochure/Sect_5-1.html

[43] T. Fry, S. Hauck, "SPIHT Image Compression on FPGAs", IEEE Transactions on Circuits and Systems for Video Technology., Vol. 15, No. 9, Sep. 2005, pp 1138-1147.

[44] S. Fukuma,, T. Tanaka and M. Nawate, “Switching Wavelet Transform for ROI Image Coding”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 2005, Vol. E88-A, No. 7, pp. 1995-2006.

[45] R. Green, G. Asner, and F. Muller-Karger, “The Plant Physiology and Functional Type NASA Mission - Concept Study”, AVIRIS Annual JPL Airborne Earth Science Workshop. Pasadena, CA, May 2007.

[46] A. Goetz and M. Herring, “The high resolution imaging spectrometer (HIRIS) for EOS”, EEE Transactions on Geoscience and Remote Sensing (ISSN 0196-2892), vol. 27, March 1989, p. 136-144.

[47] M. Goldberg, L. Zhou, and W. Wolf, “Applications of Principal Component Analysis (PCA) to AIRS Data”, Proceedings of SPIE, Vol. 5655 January 2005, pp. 479-488

[48] A. Grzeszczak, Mandal, M.K., S. Panchanathan, S. and T. Yeap, “VLSI implementation of discrete wavelet transform” IEEE Transactions on VLSI Systems, Volume 4, Dec. 1996, pp 421 –433

[49] J. Gruninger, R. L. Sundberg, M. J. Fox, R. Levine, W. F. Mundkowsky, M. S. Salisbury and A. H. Ratcliff, “Automated Optimal Channel Selection for Spectral Imaging Sensors”, Proceedings SPIE 4381, Algorithms for Multi-spectral and Hyper-spectral Imagery VII,[4381-07], Orlando April 2001.

[50] S. Hou, “Modern Techniques in Sounder Data Compression”, The Aerospace Corporation, May 2003

[51] K. Huang and J. Abraham, “Algorithm-based fault tolerance for matrix operations,” IEEE Trans. Comput., vol. 33, no. 6, pp. 518-528, 1984.

[52] “Image Data Compression”, Recommendation for Space Data Systems Standard, The Consultative Committee for Space Data Systems(CCSDS), Blue Book, Issue 1. Nov. 2005 (CCSDS 122.0-B-1).

[53] W. Jiang and A. Ortega, "Efficient Discrete Wavelet Transform Architectures Based on Filterbank Factorizations" in Intl. Conf. on Image Processing, Kobe, Japan, Oct. 1999.

[54] W. Jiang and A. Ortega, "Parallel Architecture for the Discrete Wavelet Transform based on the Lifting Factorization" in Proc of SPIE in Parallel and Distributed Methods for Image Processing III, Denver, CO., USA, July 1999

[55] W. Jiang, “ Contribution to transform coding system implementation”, Ph.D. Thesis, USC, May 2000.

[56] W. Jiang and A. Ortega, “Discrete Wavelet Transform System – Architecture Design Using Filter Bankc”, USC Special Report, May 1999

[57] JPEG2000 Image Coding System, ISO/IEC FCD15444-1, March 2000.

[58] JPEG2000 3D (Part 10 – JP3D): http://www.jpeg.org/jpeg2000/j2kpart10.html

[59] JPEG2000 Encoder Core, Cast Inc., Oct. 2002

[60] J. Jou and J. Abraham, “Fault-tolerance matrix arithmetic and signal processing on highly computing structures,” Proc. IEEE, vol.74, no.5, pp, 732-741, 1984

[61] Y. Kang, "Low-power design of wavelet processors" in Proc. SPIE, vol.2308, 1993, pp.1800-1806.

[62] A. Kiely and M. Klimesh, “The ICER Progressive Wavelet Image Compressor”, IPN progress report, JPL, Nov. 2003.

[63] A. Kiely, H. Xie, M. Klimesh, N. Aranki, “ICER-3D: A progressive wavelet-based compressor for hyperspectral images,” IPN Progress Report, vol. 42-163, November 15, 2005.

[64] A. Kiely, M. Klimesh, “Preliminary Image Compression Results from the Mars Exploration Rovers,” IPN Progress Report, vol. 42-156, February 15, 2004, pp. 1–8.

[65] A. Kiely, M. Klimesh, and J. Maki, “ICER on Mars: Wavelet-Based Image Compression for the Mars Exploration Rovers,” IND Technology and Science News, Issue 15, to appear, 2002.

[66] A. Kiely, N. Aranki, M. Klimesh, H. Xie, “Hyperspectral Data Compression: Algorithm and Software Development”, JPL’s Interplanetary Network Directorate, Annual Review Report, Sep, 2004.

[67] A. Kiely, M. Klimesh, N. Aranki, H. Xie, “A progressive 3D wavelet-based compressor for hyperspectral images”, NASA Tech Briefs, 2007.

[68] A. Kiely, H. Xie, M. Klimesh, N. Aranki, “Hyperspectral Data Compression (ICER-3D): Enhanced Context Modeler”, JPL’s Interplanetary Network Directorate, Annual Review Report, Oct. 18, 2005.

[69] B.-J. Kim, Z. Xiong, and W. A. Pearlman, “Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT),” IEEE Transactions on Circuits and Systems for Video Technology, vol. 10, no. 8, pp. 1374–1387, December 2000.

[70] H. King-Ch, H Yu-Jung, T. Trieu-Kien, and W. Chia-Ming, “FPGA implementation for 2D discrete wavelet transform”, Electronics Letters, 1998 Vol. 34, pp 639 –640.

[71] M. Klimesh, “Low-Complexity Lossless Compression of Hyperspectral Imagery via Adaptive Filtering,” The Interplanetary Network Progress Report, vol. 42-163, Jet Propulsion Laboratory, Pasadena, California, pp. 1–10, November 15, 2005

[72] M. Klimesh, A. Kiely, H. Xie, N. Aranki, “Spectral Ringing Artifacts in Hyperspectral Image Data Compression,” Book chapter in “Hyperspectral Data Compression”, G. Motta, F. Rizzo, J. Storer, editors, Springer, October 2005.

[73] M. Klimesh, A. Kiely, H. Xie, N. Aranki, “Spectral Ringing Artifacts in Hyperspectral Image Data Compression,” IPN Progress Report, vol. 42-160, pp. 1–17, February 15, 2006.

[74] M. Klimesh, A. Kiely, N. Aranki, H. Xie “Techniques for Improving the Effectiveness of 3D Wavelet-Based Compression of Hyperspectral Images”, NASA Tech Briefs, 2007.

[75] G. Knowels, “VLSI architecture for the discrete wavelet transform”, Electron. Lett., 1990, Vol 26, pp.1184-1185.

[76] A. Lewis and G. Knowels, “VLSI architecture for 2D Daubechies wavelet transform without multipliers”, Electron. Lett., 1991, Vol 27, pp. 17l-173.

[77] Y. Long, C. Zhang and F. Kurdahi, “A high-performance parallel mode EBCOT encoder architecture design for JPEG2000”, Proceedings, IEEE International SOC Conference, Sep. 2004, pp 213- 216.

[78] S. Mallat, “A theory for multiresolution signal decomposition: The wavelet representation", IEEE Trans. on Patt. Anal. and Mach. Intell.11 (7), pp. 674-693, 1989.

[79] S. Mallat, "Multifrequncy channel decompositions of images and wavelet models," IEEE Trans. Acoust., Speech, and Signal Processing, vol.37, pp.2091-2110, Dec.1989.

[80] R. Martin, G. Asner. 2005. New insight to carbon and nutrient cycles from airborne imaging spectroscopy. Proc. 2005 AVIRIS Earth Science and Applications Workshop, Pasadena, CA, May 24-27, 2005.

[81] B. Masschelein, B. Vanhoof, L. Nachtergaele, J. Bormans, and I. Bolsens, "Implementation Driven Selection of Wavelet Filters for Still Image Coding Based on Bitrange Expansion," IEEE International Workshop on Multimedia Signal Processing, Copenhagen, Denmark, pp. 371–376, September 13–15, 1999.

[82] A. Miguel, A. Askew, A. Chang, S. Hauck, R. Ladner, and E. Riskin, "Reduced Complexity Wavelet-Based Predictive Coding of Hyperspectral Images for FPGA Implementation", Data Compression Conference, Snowbird, UT, March 2004, pp. 469-478.

[83] MotionWavelets Website: http://www.aware.com/imaging/motionwavelets.htm.

[84] G. Motta, F. Rizzo, and J. Storer, "Compression of Hyperspectral Imagery", Data Compression Conference, Snowbird, UT, March 2003, pp. 333-342.

[85] A. Rao, and S. Bhargava, "Multispectral Data Compression using Bidirectional Interband Prediction", IEEE Trans. On Geoscience and Remote Sensing, 34 (2):385-396, March 1996.

[86] A. Reza, R. Turney, "FPGA implementation of 2d wavelet transform", Signals, Systems, and Computers, Conference Record of the Thirty-Third Asilomar , 1999, Vol. 1, pp 584 –588.

[87] J. Ritter, P. Molitor, "A Pipelined Architecture for Partitioned DWT Based Lossy Image Compression using FPGAs," ACM/SIGDA Ninth International Symposium on Field Programmable Gate Arrays, pp 201 – 206, February 2001.

[88] F. Rizzo and B. Carpentieri, "High Performance Compression of Hyperspectral Imagery with Reduced Search Complexity in the Compressed Domain", Data Compression Conference, Snowbird, UT, March 2004, pp. 479-488.

[89] J. Saghri, A. Tescher, and J. Reagan, "Practical Transform coding of Multispectral Imagery", IEEE Signal Processing Magazine, Jan. 1995, pp. 32-43.

[90] A. Said and W. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp. 243–250, June 1993.

[91] A. Secker and D. Taubman, “Motion-compensated highly scalable video compression using an adaptive 3D wavelet transform based on lifting”, Proc. of International Conference on Image Processing, Vol 2, Oct 2001, pp.1029-1032.

[92] L. Senhadji, G. Carrault, and J. J. Bellanger, "Interictal EEG spike detection: A new framework based on wavelet transform," in Proc. IEEE-SP mt. Symp. Time-Frequency Time-Scale Anal, pp. 548-55I, Philadelphia, PA, Oct 1994.

[93] A. Stocker and A. Shaum, “Application of Stochastic Mixing Models to Hyperspectral Detection Problems”, Proceedings SPIE 3071, Algorithms for Multispectral and Hyperspectral Imagery III,47-60, 1997.

[94] W. Sweldens, “The lifting scheme: A new philosophy in biorthogonal wavelet constructions" in Wavelet Applications in Signal and Image Processing III, A. F. Laine and M. Unser, eds., pp. 68-79, Proc. SPIE 2569, 1995.

[95] X. Tang, W. Pearlman and J. W. Modestino, “Hyperspectral Image Compression Using Three-Dimensional Wavelet Coding”, SPIE/IS&T Electronic Imaging 2003, Proceedings of SPIE Vol. 5022, Jan. 2003.

[96] X. Tang, S. Cho, and W. A. Pearlman, “3D set partitioning coding methods in hyperspectral image compression,” in Proc. 2003 International Conference on Image Processing, vol. II, 14–17 Sept. 2003, pp. II–239–II–242.

[97] X. Tang and W. Pearlman, “Three-Dimensional Wavelet-Based Compression of Hyperspectral Images”, Book chapter in “Hyperspectral Data Compression”, G. Motta, F. Rizzo, J. Storer, editors, Springer, October 2005.

[98] D. Taubman, “High Performance Scalable Image Compression with EBCOT,” IEEE Transactions on Image Processing, vol. 9, no. 7, pp. 1158–1170, July 2000.

[99] D. Taubman and M. Marcellin, “JPEG2000: Image Compression Fundamentals, Standards and Practice,” Kluwer Academic Publishers, 2002.

[100] R. Tawel and N. Aranki, “IMAS – Multispectral Image Coding”, Technical Report, Jet Propulsion Laboratory , Pasadena, CA, Jan. 1998.

[101] J. Tham, S. Ranganath, and A. Kassim, “Highly scalable wavelet-based video codec for very low bit-rate environment,” IEEE Journal on Selected Areas in Communications, vol. 16, no. 1, pp. 12–27, January 1998.

[102] A. Uhl, “A parallel approach for compressing satellite data with wavelets and wavelet packets using PVM," in Workshop Paragraph '94, RISC - Linz Report Series No. 94-17, 1994.

[103] M. Unser, "Texture classification and segmentation using wavelet frames." IEEE Trans. Image Processing. vol.4. pp. 1549-1560, Nov. 1995.

[104]P. P. Vaidyanathan and P.-Q. Hoang, “Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks," IEEE Trans. Acoust., Speech, Signal Processing 36, pp. 81-94, Jan. 1988.

[105]P. P. Vaidyanathan, “Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial", Proceedings of The IEEE 78 , pp. 56-93, Jan. 1990.

[106]G. Vane, et. al., “The Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)”, Remote Sens. Environ. 44, pp 127-143, 1993.

[107]J. Venbrux, J. Gambles, D. Wiseman, G. Zweigle, W. H. Miller, and P.-S. Yeh, “A VLSI Chip Set Development for Lossless Data Compression,” Ninth AIAA Computing in Aerospace Conference, San Diego, California, October 19–21, 1993.

[108]M. Vishwanath, “Discrete wavelet transform in VLSI”, in Proc. IEEE Int. Conf. Appl. Specfic Array Processors, 1992, pp. 218-229.

[109]M. Vishwanath, M. Michael, R.M., and M.J. Irwin, “VLSI architecture for the discrete wavelet transform”, IEEE Trans. Circuits Syst. -II Anal Dig. Signal Process., 1995, Vol. 42, pp.305-316.

[110]Y. Wang, J. Rucker, and J. Fowler, “Three-dimensional tarp coding for the compression of hyperspectral images,” IEEE Geoscience and Remote Sensing Letters, vol. 1, no. 2, pp. 136–140, April 2004.

[111]M. Weinberger, G. Seroussi, and G. Sapiro, “The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS,” IEEE Transactions on Image Processing, vol. 9, no. 8, pp. 1309–1324, August 2000.

[112]H. Xie, A. Kiely, M. Klimesh, N. Aranki, “Context Modeler for Compression of Wavelet-Transformed Hyperspectral Data”, NASA Tech Briefs, 2007

[113]Xilinx Virtex II and Virtex II pro data books, 2006.

[114]Z. Xiong, X. Wu, S. Cheng, and J. Hua, “Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms,” IEEE Transactions on Medical Imaging, vol. 22, no. 3, pp. 459–470, March 2003.

[115]K. Youn-Hong, J. Kyong-il, and R. Kang-Hyeon, “FPGA implementation of subband image encoder using discrete wavelet transform”, TENCON 99. Proceedings of the IEEE Region 10 Conference, 1999, Vol. 2, pp 1335 -1338.

[116]C. Zhang, Y. Long, S. Oum and F. Kurdahi, “ Software-Pipelines 2D Discrete Wavelet Transform with VLSI Hierarchical Implementation”, Proceedings, IEEE Inter. Conference on Robotics, Intelligent Systems and Signal Processing, Vol. 1, pp 148-153, Oct. 2003.

Appendices

Appendix A: An Alternative Approach to Hyperspectral Data Compression

An approach with lower complexity than ICER-3D-HW is to use a 2D DWT decomposition in the spatial dimension and predictive coding among spectral bands (see Figure A.1), which provides a less computationally intensive approach. Most predictive techniques (such as DPCM)[4][26][85]) operate on both spatial and spectral dimensions and are not scalable. The DWT produces coding gains by exploiting data correlations in the spatial dimension. Prediction in the wavelet domain provides a good lossless/virtually lossless approach that allows for faster adaptation of the local statistics between DWT transformed spectral bands. It requires less computation than another 1D wavelet transform, scalable, however, the algorithm is not progressive.

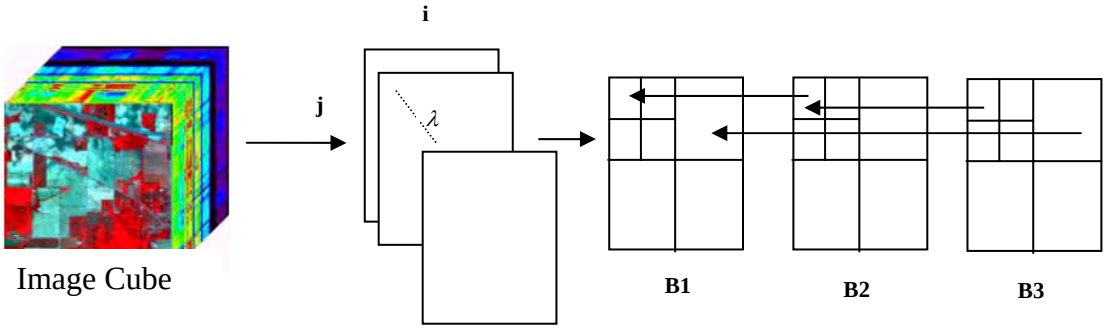


Figure A.1: Spectral data is arranged as 2D images (spectral bands), integer DWT applied to 2D images followed by inter-band prediction

Our alternative algorithm applies first a 2D wavelet decomposition to each spectral band (spatial domain), then exploits spectral correlations through interband predictive coding. Quantization (if lossy) and entropy coding are applied to the error (difference) images to complete the compression process.

Users aiming for lower complexity and high bit rates may choose the 2D DWT along with spectral prediction, while users looking for a progressive lossy/lossless compression may choose the 3D DWT algorithm, ICER-3D-HW.

A.1 2D DWT Compression with Prediction

We investigated various prediction schemes. Prediction depth of more than 2 or three planes in the wavelet transform domain did not produce better results and came at a high computational cost. The following 3 adaptive predictors were designed for this part of the algorithm

Predicted value $\underline{X}$:

$$\underline{X} = X_{i,j,\lambda-1} + a\left(X_{i,j,\lambda-1} - X_{i,j,\lambda-1}\right) \qquad (A.1)$$

$$\underline{X} = a + bX_{i,j,\lambda-1} + cX_{i,j,\lambda-2} \qquad (A.2)$$

$$\underline{X} = a + bX_{i,j,\lambda} + cX_{i,j,\lambda-1} - dX_{i,j,\lambda-2} \qquad (A.3)$$

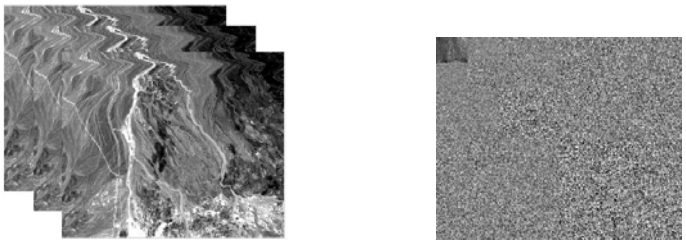
where i and j are the special coordinates of a pixel, λ is the spectral band being predicted, and the coefficients a, b, c, and d are computed with least square

minimization. Representative samples of calibrated ’97 AVIRIS data downloaded from the AVIRIS website were used for training [20]. On-board adaptive training is also possible, but comes at a higher computational cost. A practical solution can update the coefficients less frequently.

The residual $\underline{r}$ is given by:

$$\underline{r} = X_{i,j,\lambda} - \underline{X} \qquad (A.4)$$

Figure A.2 shows an example of 3 consecutive spectral bands of an AVIRIS data set and the resulting residual image using the first predictor



(a) (b)

Figure A.2: 2D wavelet decomposition with spectral predictive coding. (a) Three consecutive spectral bands of AVIRIS Cuprite scene. (b) The resulting residual image

A.2 Quantization and Entropy Coding

While bit plane encoding with prioritized DWT sub-bands might be the ideal approach for post transform processing, our first baseline approach was to use scalar quantizers (when lossy compression is required) and entropy encoding to minimize algorithm complexity.

Quantization aims to reduce data entropy by decreasing the data precision. A quantization scheme maps a large number of input values into a smaller set of output values. This implies that some information is lost during the quantization process. A quantization strategy design must balance the compression achievement and information loss. One of the criteria for optimal quantization is minimizing the mean square error (MSE) given a quantization scale. Our wavelet coefficients were compressed using a uniform scalar quantizer and the Lloyd-Max optimal scalar quantization scheme. Residual errors resulting from the 2D DWT with spectral prediction scheme were quantized with a uniform quantizer operating on each sub-band error separately. Sub-band blocks resulting from the 3D DWT decomposition consist of two types of data, the high energy LLL block, which preserves most of the energy; and the other high-resolution data blocks, which contain the sharp edge information. Small quantization scale was used for the LLL block and a relatively large scale for the other blocks.

The last step in the compression process is the entropy coding. Huffman coding is a variable length scheme that is a minimum redundancy coding. It assigns fewer bits to the values with a higher frequency of occurrence and more bits to the values with a lesser frequency of occurrence. Based on the occurrence frequency of each quantization level, a hierarchical binary coding tree structure brings by sequentially

finding the lowest two frequencies as tree branches and adding each low frequency pair as a new node for the next level. Since each data block is quantized into different numbers of quantization levels, the coding process was performed for each data block separately resulting in a separate code book for each sub-band.

A.3 Experimental Results

Our algorithm was tested using the (2,6) DWT transform with AVIRIS '97 data sets. Figures A.3, A.4 and A.5 demonstrate the lossless and lossy results accomplished with this approach and in comparison to 2D DWT compression and a baseline 3D DWT using the same type of encoding. The spectral prediction used in these runs is based on the simple predictor shown in equation (A.1).

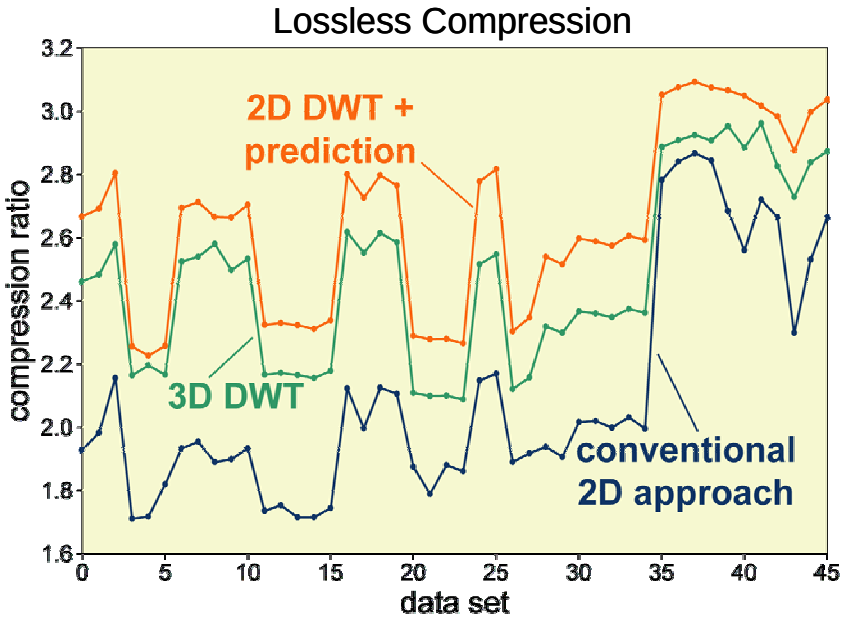


Figure A.3: Lossless compression - Comparison of 2D DWT with prediction compressor to 3D and 2D DWT compressors

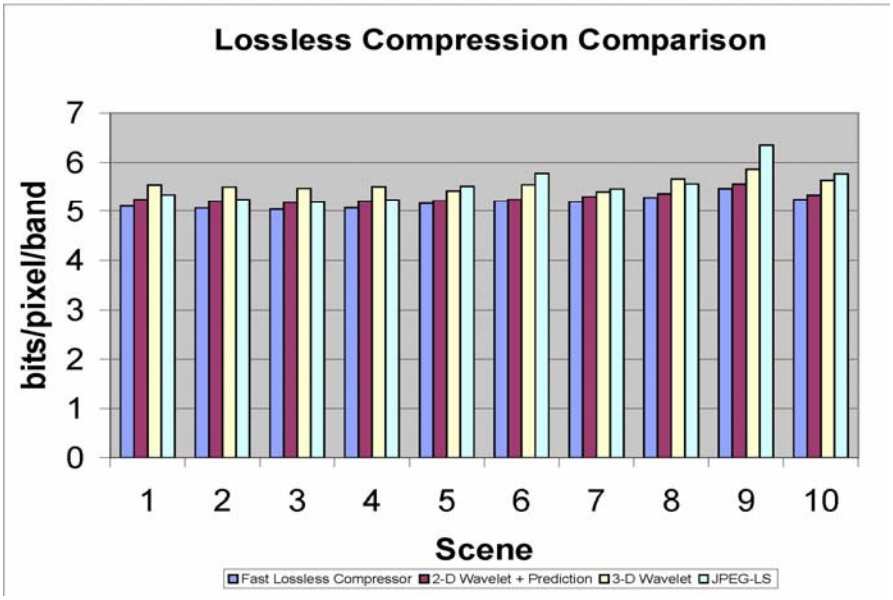


Figure A.4: Lossless Results – Comparisons of 2D DWT with prediction compressor to fast lossless, 3D DWT, and JPEG-LS lossless compressors

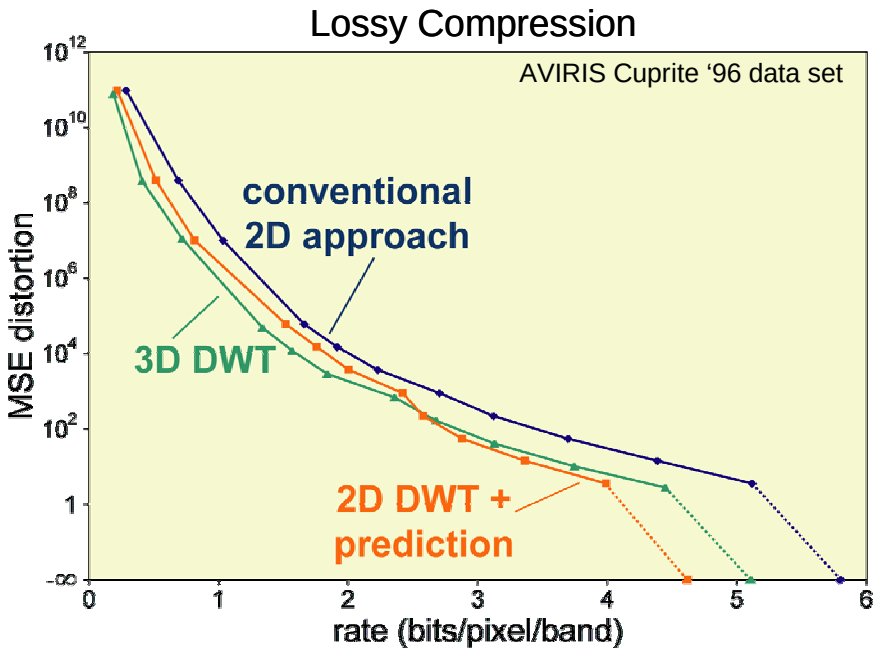


Figure A.5: Lossy compression – Comparisons of 2D DWT with prediction compressor to 3D and 2D DWT compressors

As can be seen from these results, the proposed DWT with prediction algorithm is not optimal for all types of hyperspectral data, but produced good results with low computational complexity. Enhanced prediction scheme, bit-plane encoding, and entropy coding can be pursued with further research and can be expected to improve the compression performance.