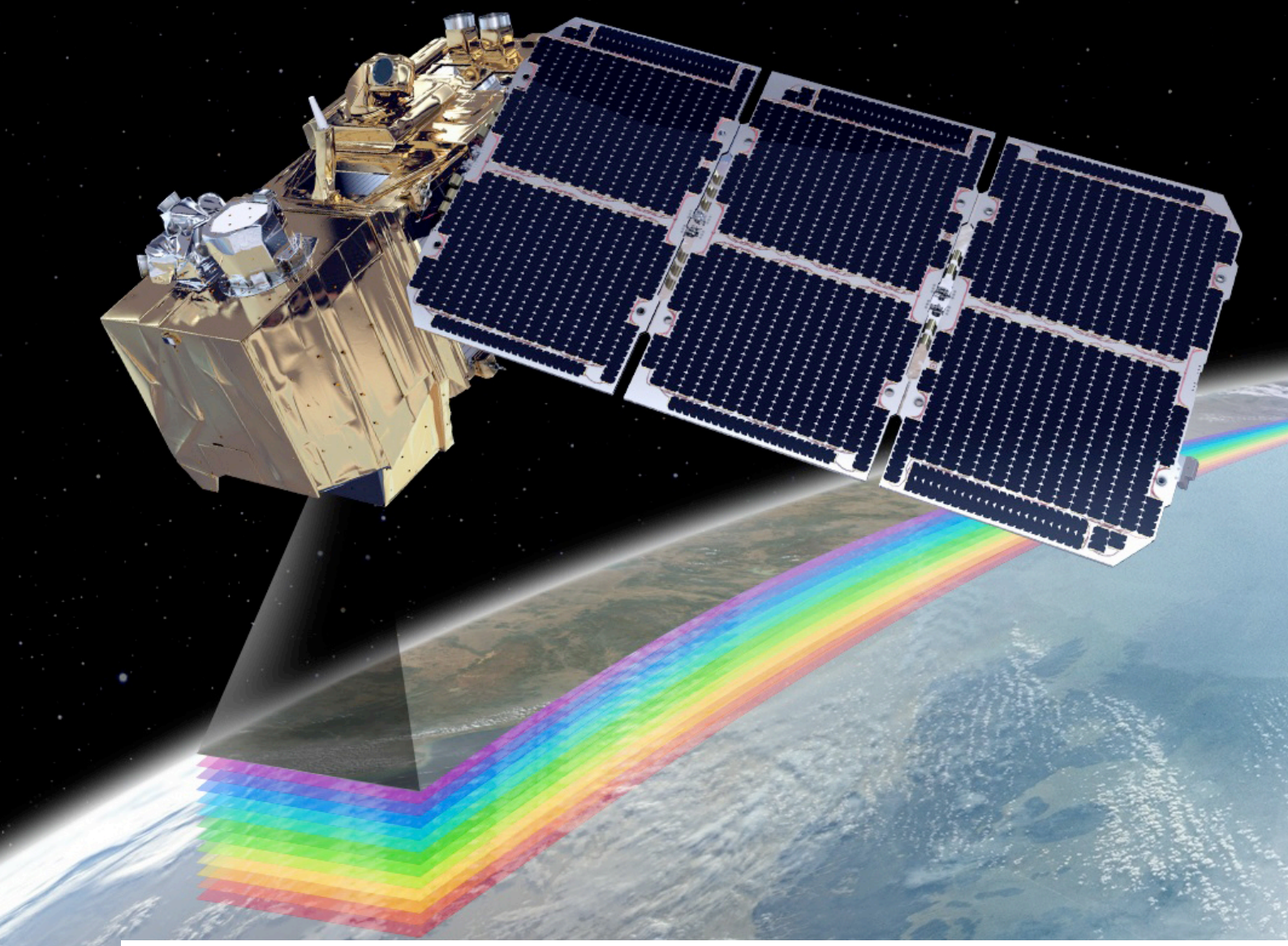
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada

Doctorado en Tecnologías de Telecomunicación



Implementaciones hardware usando nuevas técnicas de diseño de sistemas de compresión con pérdidas de imágenes hiperespectrales para futuras misiones espaciales

Aday García del Toro
Las Palmas de Gran Canaria, junio 2017

D. PEDRO PÉREZ CARBALLO, SECRETARIO DEL INSTITUTO UNIVERSITARIO DE MICROELECTRÓNICA APLICADA DE LA UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA,

CERTIFICA,

Que la Comisión Académica del Instituto Universitario de Microelectrónica Aplicada (IUMA), en su sesión de fecha **nueve de junio de 2017**, tomó el acuerdo de dar el consentimiento para su tramitación, a la tesis doctoral titulada **"Implementaciones hardware usando nuevas técnicas de diseño de sistemas de compresión con pérdidas de imágenes hiperespectrales para futuras misiones espaciales"** presentada por el doctorando D. Aday García del Toro y dirigida por los Doctores D. Roberto Sarmiento Rodríguez, D. José Francisco López Feliciano y Dña. María Lucana Santos Falcón.

Y para que así conste, y a efectos de lo previsto en el Artº 6 del Reglamento para la elaboración, defensa, tribunal y evaluación de tesis doctorales de la Universidad de Las Palmas de Gran Canaria, firmo la presente en Las Palmas de Gran Canaria, a **nueve de junio de dos mil diecisiete**.

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

Instituto: INSTITUTO UNIVERSITARIO DE MICROELECTRÓNICA APLICADA

Programa de doctorado: TECNOLOGÍAS DE TELECOMUNICACIÓN

Título de la Tesis

IMPLEMENTACIONES HARDWARE USANDO NUEVAS TÉCNICAS DE DISEÑO DE SISTEMAS DE COMPRESIÓN CON PÉRDIDAS DE IMÁGENES HIPERESPECTRALES PARA FUTURAS MISIONES ESPACIALES

Tesis Doctoral presentada por D. ADAY GARCÍA DEL TORO

Dirigida por el Dr. D. ROBERTO SARMIENTO RODRÍGUEZ

Codirigida por el Dr. D. JOSÉ FCO. LÓPEZ FELICIANO

Codirigida por la Dra. Dña. MARÍA LUCANA SANTOS FALCÓN

El Director/a, (firma)

El Codirector/a (firma)

El Codirector/a (firma)

El Doctorando, (firma)

Las Palmas de Gran Canaria, a 12 de junio de 2017

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

DIVISIÓN DE DISEÑO DE SISTEMAS INTEGRADOS

Tesis Doctoral

Implementaciones hardware usando nuevas técnicas de diseño de sistemas de compresión con pérdidas de imágenes hiperespectrales para futuras misiones espaciales

Aday García del Toro

Las Palmas de Gran Canaria, junio 2017

Dedicado a Laura Ojeda

Agradecimientos

Recordaré siempre el día en el que, tras una reunión en la que “no habíamos ido a perder el tiempo”, mis directores de tesis me plantearon la posibilidad de realizar la tesis doctoral. Mi reacción inmediata fue: ¿y para qué? Pues bien, aquella idea que en principio pareció peregrina, fue madurando y tomando forma hasta llegar a este momento. Sigo sin lograr dar una respuesta concreta a aquella pregunta, pero su respuesta debe estar irremediablemente relacionada con esta sensación de gratitud y orgullo tras el duro trabajo que hemos realizado. Empleo el plural de manera consciente ya que ha sido un esfuerzo que llegue hasta aquí y quiero agradecéroslo.

Me gustaría comenzar agradeciendo a mis directores de tesis todas las horas que me han dedicado y todos los esfuerzos que han realizado para poder juntarnos siempre que se podía. Roberto, José, gracias por vuestra dedicación y apoyo constante, no sólo en esta tesis doctoral, sino por todos los más de 10 años en los que habéis estado siempre ahí. Lucana, gracias por tu rigurosidad y por dedicarme el tiempo que sé no tenías y hacías el esfuerzo por atenderme, has sido fundamental en esta tesis doctoral.

Quiero agradecer especialmente a mi prometida, Laura, todos los esfuerzos que ha tenido que realizar para permitirme compatibilizar el trabajo realizado en esta tesis doctoral con mi actividad profesional. Han sido muchas las horas que nos he secuestrado para poder desarrollar este trabajo, durante todo este tiempo siempre has estado apoyándome y animándome para que acabara lo que ya había empezado. Por todo, este trabajo también es tuyo, y por eso te lo dedico, eres la persona que más se lo merece.

Termino dando las gracias a mi familia. Gema, Sergio y Santiago gracias por vuestra fuerza y apoyo incondicional en todas las decisiones de mi vida. Olga, Antonio y Cristina, os estaré eternamente agradecido por tanto apoyo, entrega y cariño que me habéis dado. Me siento afortunado de teneros conmigo.

Resumen

Los sensores a bordo de los satélites de observación de la tierra generan una gran cantidad información que ha de ser transmitida a tierra para poder ser empleada. Debido al limitado, y en ocasiones insuficiente, ancho de banda de comunicaciones y capacidades de almacenamiento a bordo, existe un esfuerzo creciente en proporcionar algoritmos de compresión que optimicen tanto el canal de comunicaciones como el almacenamiento disponible. En este sentido, las imágenes hiperespectrales generan una gran cantidad de información a bordo que ha de ser comprimida. Los algoritmos de compresión en el entorno espacial han de presentar una alta eficiencia de compresión, así como, una implementación viable sobre dispositivos calificados para operar en el espacio.

En esta tesis doctoral se describen las contribuciones realizadas en el campo de la compresión con pérdidas de imágenes hiperespectrales a bordo de futuras misiones espaciales. El objetivo de estas contribuciones es proporcionar implementaciones hardware sobre dispositivos de tipo FPGA capaces de operar a bordo. Asimismo, se evalúa la aptitud de las metodologías de síntesis de alto nivel para la implementación hardware de algoritmos de compresión de imágenes hiperespectrales en el sector espacial.

En primer lugar, se presenta la implementación hardware realizada del algoritmo predictivo LCE diseñado para la misión ExoMars. A continuación, se detalla la implementación realizada de la transformada POT, que se postula como extensión del estándar CCSDS 122.0 en la compresión de imágenes hiperespectrales. Posteriormente, se describe la implementación realizada de un algoritmo de compresión predictivo con pérdidas y con control de la tasa de bits, cuyo objetivo es ampliar el estándar de compresión sin pérdidas de imágenes hiperespectrales CCSDS 123.0 proporcionando una extensión con pérdidas y con control de la tasa de bits. El trabajo desarrollado en esta tesis doctoral se completa con la presentación del análisis comparativo de los resultados obtenidos con respecto a los disponibles en el estado del arte.

Abstract

Sensors on-board Earth observation satellites generate huge amount of information that has to be transmitted to earth to be useful. Due to the limited, and sometimes insufficient, communication bandwidth and on-board storage, there is an increasing effort to provide compression algorithms that optimize both the communication channel and available storage. In this sense, hyperspectral images generate a great amount of information on board that has to be compressed. Compression algorithms in the space environment have to provide high compression efficiency, as well as, an implementation which is able to operate on space qualified devices.

This thesis describes the contributions made in the field of lossy compression of hyperspectral images on future space missions. The purpose of these contributions is to provide hardware implementations on FPGAs that are able to operate on-board. Likewise, the suitability of the high level synthesis methodologies for the hardware implementation of lossy compression algorithms of hyperspectral images in the space sector is evaluated.

First, we present the hardware implementation of the LCE predictive algorithm designed for the ExoMars mission. Next, we detail the implementation of the POT transform. This transform is a candidate to be included as an extension of the CCSDS 122.0 standard in the compression of hyperspectral images. Then, we present the implementation of a predictive lossy compression algorithm which includes rate control capabilities. The objective of this algorithm is to extend CCSDS 123.0 lossless compression standard of hyperspectral images, providing a lossy extension with rate control capabilities. Finally, the work performed in this thesis is completed with a comparative analysis of the results obtained with respect to those available in the state of the art.

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de acrónimos

2D	Dos (**2**) **D**imensiones
3D	Tres (**3**) **D**imensiones
ASIC	**A**pplication-**S**pecific **I**ntegrated **C**ircuit
AVIRIS	**A**irbone **VI**sible/Infra**R**ed **I**maging **S**pectrometer
BI	**B**and **I**nterleaved
BIL	**B**and **I**nterleaved by **L**ine
BIP	**B**and **I**nterleaved by **P**ixel
BJT	**B**ipolar **J**unction **T**ransistor
BPE	**B**it **P**lane **E**ncoder
bpp	**b**its **p**or **p**ixel
bps	**b**its **p**or **s**egundo
BSQ	**B**and **S**e**Q**uential
CALIC	**C**ontext-**A**daptive **L**ossless **I**mage **C**oding
CCSDS	**C**onsultative **C**ommittee for **S**pace **D**ata **S**ystems
C-DPCM	**C**lustered **D**ifferential **P**ulse **C**ode **M**odulation
CLB	**C**onfigurable **L**ogic **B**lock

CNES	**C**entre **N**ational d'**É**tudes **S**patiales
CPU	**C**entral **P**rocessing **U**nit
CWICOM	CCSDS **W**avelet **I**mage **C**ompression **M**odule
dB	**d**eci**B**elio
DCT	**D**iscrete **C**osine **T**ransform
DFF	**D**ifferential **F**lip-**F**lop
DPCM	**D**ifferential **P**ulse **C**ode **M**odulation
DSP	**D**igital **S**ignal **P**rocessor
DWT	**D**iscrete **W**avelet **T**ransform
ECC	**E**rror **C**orrecting **C**odes
ECSS	**E**uropean **C**ooperation for **S**pace **S**tandardization
EDAC	**E**rror **D**etection **A**nd **C**orrection
EDRS	**E**uropean **D**ata **R**elay **S**ystem
EEPROM	**E**lectrically **E**rasable/**P**rogrammable **R**ead-**O**nly **M**emory
ESA	**E**uropean **S**pace **A**gency
FAPEC	**F**ully **A**daptive **P**rediction **E**rror **C**oder
FET	**F**ield-**E**ffect **T**ransistor
FIFO	**F**irst **I**n, **F**irst **O**ut
FL	**F**ast **L**ossless
FLEX	**F**ast **L**ossless **EX**tended
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
FSM	**F**inite **S**tate **M**achine
Gbps	**G**iga **b**its **p**or **s**egundo
GCR	**G**alactic **C**osmic **R**ay
GPP	**G**eneral **P**urpose **P**rocessor

GPU	**G**raphics **P**rocessing **U**nit
HDL	**H**ardware **D**escription **L**anguage
HLS	**H**igh **L**evel **S**ynthesis
HyLoC	**Hy**perspectral **Lo**ssless **C**ompressor
ISO	**I**nternational **O**rganization for **S**tandardization
IUMA	**I**nstituto **U**niversitario de **M**icroelectrónica **A**plicada
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
JPL	**J**et **P**ropulsion **L**aboratory
JUICE	**JU**piter **IC**y moons **E**xplorer
KLT	**K**arhunen-**L**oève **T**ransform
LAIS-LUT	**L**ocally **A**veraged **I**nterband **S**caling - **L**ook**U**p **T**able
LCE	**L**ossy **C**ompression for **E**xo**M**ars
LMS	**L**east **M**ean **S**quare
LOCO-I	**LO**w **CO**mplexity Lossless Compression for **I**mages
LUT	**L**ook**U**p **T**able
LUT-NN	**L**ook**U**p **T**able - **N**earest **N**eighbor
Mbps	**M**ega **b**its **p**or **s**egundo
MBU	**M**ultiple **B**it **U**pset
MCT	**M**ultiple **C**omponent **T**ransformation
M-CALIC	**M**ultiband **C**ontext-**A**daptive **L**ossless **I**mage **C**oding
MRCPB	**M**ulti-**R**ésolution par **C**odage de **P**lans **B**inaires
MOSFET	**M**etal-**O**xide-**S**emiconductor **F**ield-**E**ffect **T**ransistor
MSE	**M**ean **S**quared **E**rror
MSI	**M**ulti**S**pectral **I**nstrument
MSS	**M**ulti**S**pectral **S**canner

NASA	**N**ational **A**eronautics and **S**pace **A**dministration
OLI	**O**perational **L**and **I**mager
OTP	**O**ne **T**ime **P**rogrammable
PLL	**P**hase-**L**ocked **L**oop
POT	**P**airwise **O**rthogonal **T**ransform
PSNR	**P**eak **S**ignal-to-**N**oise **R**atio
RAM	**R**andom **A**ccess **M**emory
ROM	**R**ead **O**nly **M**emory
RKLT	**R**eversible **K**arhunen-**L**oève **T**ransform
RTL	**R**egister **T**ransfer **L**evel
SEB	**S**ingle **E**vent **B**urnout
SEE	**S**ingle **E**vent **E**ffect
SEGR	**S**ingle **E**vent **G**ate **R**upture
SEL	**S**ingle **E**vent **L**atchup
SET	**S**ingle **E**vent **T**ransient
SEU	**S**ingle **E**vent **U**pset
SLSQ	**S**pectrum-oriented **L**east **SQ**uare**s**
SNR	**S**ignal to **N**oise **R**atio
SPIHT	**S**et **P**artitioning **I**n **H**ierarchical **T**rees
SRAM	**S**tatic **R**andom **A**ccess **M**emory
SSDP	**S**calable **S**ensor **D**ata **P**rocessor
SWIR	**S**hort **W**ave **I**nfra**R**ed
TCLS	**T**riple-**C**ore **L**ock **S**tep
TID	**T**otal **I**onizing **D**ose
TMR	**T**riple **M**odular **R**edundancy

UAB	**U**niversidad **A**utónoma de **B**arcelona
USQ	**U**niform **S**calar **Q**uantizer
UTQ	**U**niform-**T**hreshold **Q**uantizer
VHDL	**V**HSIC **H**ardware **D**escription **L**anguage
VHSIC	**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit
VNIR	**V**isible and **N**ear **I**nfra**R**ed
WICOM	**W**avelet **I**mage **COM**pression

Capítulo 1

Introducción

En este capítulo se presenta el planteamiento del problema al que los trabajos desarrollados pretenden dar respuesta. En este sentido, se detallan las motivaciones que han llevado a la realización de esta tesis doctoral, asimismo, se exponen los objetivos a alcanzar en el desarrollo de este trabajo.

Con tal fin, se define el concepto de imagen hiperespectral y se contextualiza la necesidad de la compresión de este tipo de imágenes a bordo de satélites en futuras misiones espaciales.

1.1 Planteamiento del problema

Antes del comienzo de la era espacial, la humanidad no había podido tener una imagen de la esfera terrestre en un único vistazo. De hecho, jamás había tenido una visión global del planeta en el que vive. El primer vehículo espacial de observación de la tierra, a pesar de que su concepción no fuera para tal propósito, fue el satélite ruso Sputnik-1 lanzado en octubre de 1957 [1]. El satélite Sputnik-1 (Figura 1.1) realizó mediciones a lo largo de su misión que permitieron obtener las primeras estimaciones de la densidad atmosférica terrestre.

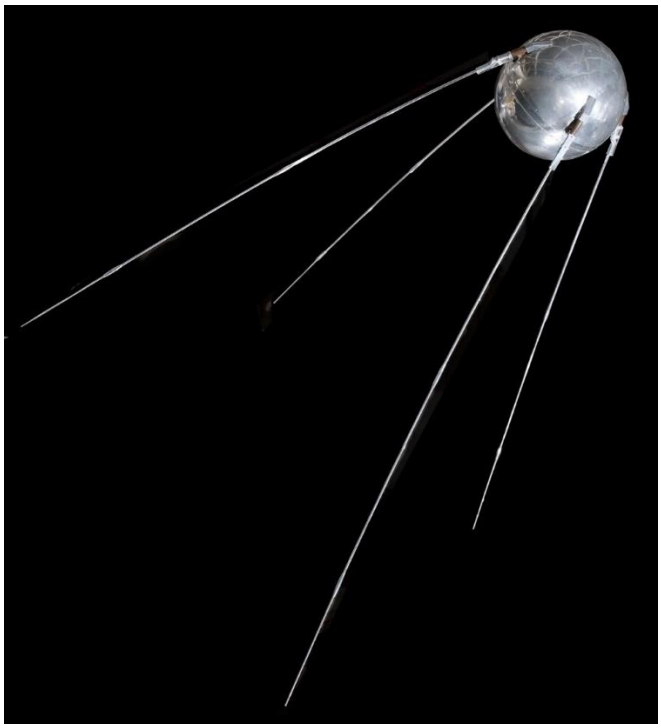


Figura 1.1 Satélite Sputnik-1 (imagen: ROSCOSMOS)

Durante las siguientes décadas hasta la actualidad, las misiones de observación de la tierra desde plataformas satelitales han sido llevadas a cabo por las distintas agencias espaciales. En ellas, la observación de la tierra mediante instrumentos ópticos ha recibido históricamente una atención especial por la gran variedad de aplicaciones en las que pueden emplearse las imágenes capturadas a bordo. Desde el principio se constató que la información en otras longitudes de ondas, diferentes a las que representan el

espectro visible, proporcionaban una información muy valiosa. Este hecho queda reflejado en el elevado número de satélites lanzados con sensores multiespectrales a bordo, y más recientemente con sensores hiperespectrales.

El programa Landsat de la NASA (del inglés, *National Aeronautics and Space Administration*) se inició con el lanzamiento de seis satélites en el año 1972 [2]. Dentro de los instrumentos a bordo de cada satélite de observación de la tierra destacan, el escáner multiespectral MSS (del inglés, *MultiSpectral Scanner*), que genera información de cuatro bandas espectrales VNIR (del inglés, *Visible and Near Infrared*) y una SWIR (del inglés, *Short Wave Infrared*) y el instrumento OLI (del inglés, *Operational Land Imager*) que añade una banda espectral en la región azul dentro del infrarrojo visible.

El programa de observación de la tierra SPOT fue iniciado en 1978 por Francia a través del CNES (del francés, *Centre National d'Études Spatiales*), en colaboración con Bélgica y Suecia convirtiéndose en la primera iniciativa a nivel europeo de observación de la tierra [3]. Fruto de esta colaboración el primer satélite, el SPOT-1, fue lanzado en el año 1986 mientras que el último lanzamiento del satélite denominado SPOT-5 fue en el año 2002. El sensor multiespectral a bordo contiene cuatro bandas espectrales, tres en el infrarrojo visible y cercano VNIR y uno en el infrarrojo de onda corta SWIR.

El primer instrumento con capacidad hiperespectral fue el sensor AVIRIS (del inglés, *Airbone VIsible/InfraRed Imaging Spectrometer*) desarrollado por el JPL (del inglés, *Jet Propulsion Laboratory*) para la observación de la tierra [4]. Se compone de un sensor óptico capaz de operar con una resolución espectral de 224 bandas contiguas dentro de las longitudes de onda desde 400 nm hasta 2500 nm. AVIRIS ha sido empleado en la observación de la tierra desde aeronaves, no siendo empleado en vuelos orbitales. Sin embargo, las imágenes generadas por este sensor son generalmente empleadas en el desarrollo de algoritmos que tienen como objetivo este tipo de imágenes.

Actualmente, el programa Copérnico de observación de la tierra, liderado por la Comisión Europea en colaboración con la ESA (del inglés,

European Space Agency), asegura la continuidad de las misiones de observación de la tierra. Seis familias de satélites se han concebido para los diferentes dominios de aplicación para los que pueden ser empleados [5]. El satélite Sentinel-2 (así son denominados secuencialmente los satélites pertenecientes al programa Copérnico) presenta dentro de su carga útil el instrumento MSI (del inglés, *MultiSpectral Instrument*) capaz de obtener 13 bandas espectrales en el rango VNIR y SWIR desde 443 nm hasta 2190 nm.

La resolución espectral y espacial de los instrumentos a bordo de satélites de observación ha causado que el procesado de la información generada suponga un reto [6]. Se espera que el volumen de datos procesado a bordo alcance los 10 TBytes/día en el momento en el que todos los satélites Sentinel pertenecientes al programa Copérnico de la ESA se encuentren en funcionamiento [7]. La Figura 1.2 muestra el volumen total estimado de datos generados anualmente del programa Copérnico comparado con los datos generados por los satélites Landsat y MODIS [8].

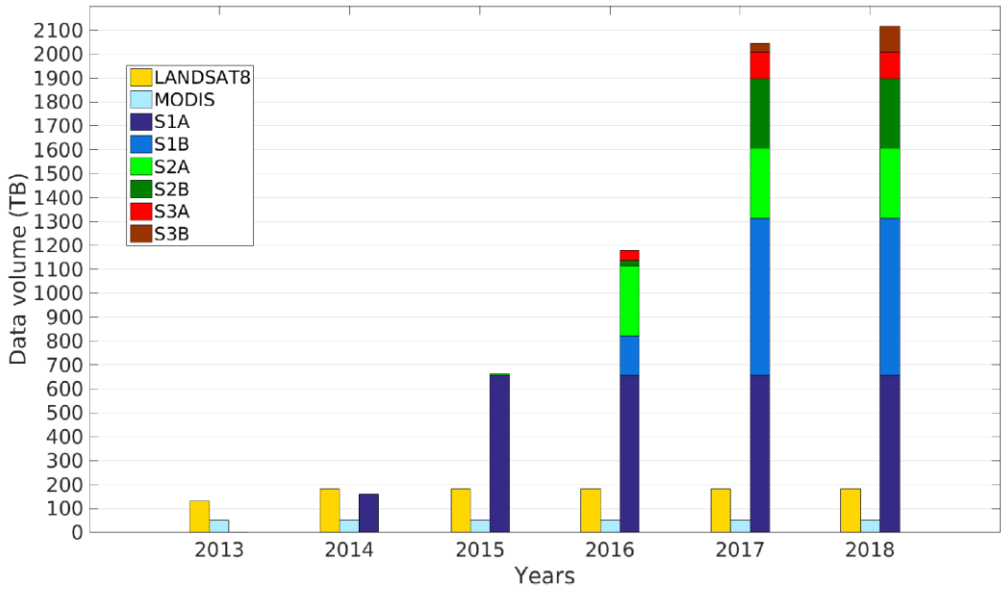


Figura 1.2 Volumen de datos generados anualmente por los satélites Sentinel comparados con los satélites Landsat8 y MODIS (imagen: [8])

Los datos adquiridos por los sensores a bordo de satélites han de ser transmitidos al segmento terreno para poder ser empleados. El ancho de banda de transmisión en una misión espacial se encuentra limitado por una

combinación de factores tales como la órbita del vehículo, el tamaño de la antena de telemetría, la potencia de transmisión y la disponibilidad de las estaciones terrenas. La Figura 1.3 muestra la evolución de la tasa de datos en bits por segundo (bps) de las comunicaciones hacia tierra desde el inicio de la era espacial. Se incluye su proyección a corto plazo en base a las misiones que actualmente se encuentran planificadas [9].

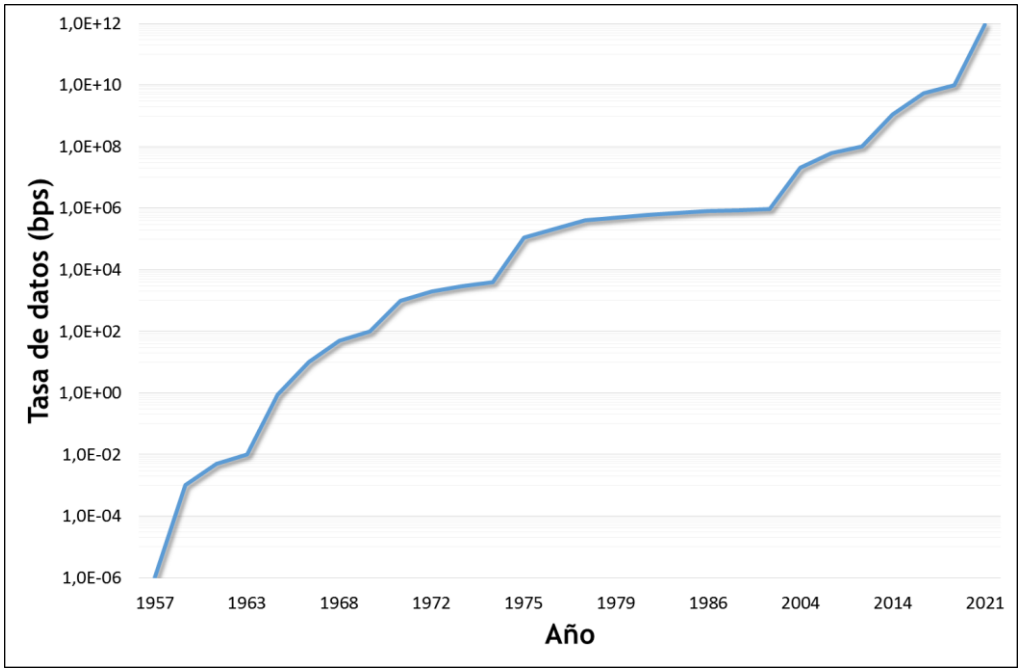


Figura 1.3 Evolución de la tasa de datos de las comunicaciones a tierra (imagen adaptada: [9])

Se aprecia que durante el periodo de 40 años comprendido entre finales de la década de 1970 hasta finales de la década de los años 2000 el ancho de banda se mantuvo estable en valores cercanos a 1 Mbps. Actualmente, el ancho de banda de comunicaciones se sitúa en torno a 1 Gbps que para la transmisión de la cantidad de datos generados resulta una limitación que refuerza la necesidad de mejorar el ancho de banda disponible. Con el fin de incrementar el ancho de banda de transmisión, la actual iniciativa EDRS (del inglés, *European Data Relay System*) proporcionará una red de satélites geoestacionarios para la transmisión de grandes volúmenes de datos [10]. Una vez se encuentre operativa esta red de satélites proporcionará una capacidad de transmisión de 50 TBytes/día. Esta evolución futura en el

ancho de banda irá acompañada con la aparición de nuevos sensores que generen tasas de datos aún mayores. Por lo tanto, optimizar el canal de comunicaciones resulta fundamental en cualquier misión espacial, siendo particularmente necesario en las misiones de observación de la tierra.

Dentro del sistema de procesado de datos, los algoritmos de compresión aumentan la eficiencia del canal de comunicaciones eliminando las redundancias de la información a transmitir. Las siguientes generaciones de sensores incrementarán la cantidad de datos a transmitir requiriendo nuevos algoritmos de compresión capaces de reducir el volumen de información. Asimismo, las futuras misiones espaciales requerirán del envío de los datos en tiempo real hasta la estación terrena, hecho que impactará en las necesidades de los algoritmos de compresión necesitando altas tasas de datos de tal modo que sea viable la operación paralela del algoritmo de compresión y el sistema de adquisición de datos [11].

1.1.1 Imagen hiperespectral

Los sensores ópticos de observación de la tierra adquieren la reflectancia de la superficie terrestre con el fin de obtener la firma espectral característica de cada material [12]. Una imagen hiperespectral es una imagen de una misma localización espacial tomada por un sensor hiperespectral a diferentes longitudes de onda en la que cada píxel representa la reflectancia de los materiales contenidos en ella. Por consiguiente, si se toman distintas imágenes a diferentes longitudes de onda, cada píxel se encontrará representado por un vector de valores espectrales con la contribución de la luz detectada en ese punto para cada banda en el espectro. De esta forma, una imagen hiperespectral estará formada por un número finito de bandas que puede ir desde las decenas hasta los varios centenares de bandas dependiendo del sensor empleado. Si bien no existe una definición concreta en la literatura, generalmente, una imagen con decenas de bandas es denominada imagen multiespectral, mientras que a una imagen con cientos de bandas se le denomina imagen hiperespectral y una imagen con miles de bandas se conoce como imagen ultraespectral. Por claridad en la nomenclatura en este documento, se emplea la denominación de imagen

hiperespectral para hacer referencia a imágenes multiespectrales, hiperespectrales y ultraespectrales, exceptuando los casos concretos donde la explicación exija una diferenciación explícita.

A la hora de representar las imágenes hiperespectrales se emplea comúnmente el concepto de hipercubo o cubo hiperespectral. La Figura 1.4, muestra la notación empleada en la formación de una imagen hiperespectral. Las dos primeras dimensiones, denominadas x e y respectivamente, representan la ubicación en el espacio de un píxel determinado de la imagen y la tercera dimensión, denominada z, representa la longitud de onda a la que ha sido tomada la imagen. De este modo, una imagen hiperespectral contiene N_x columnas de píxeles, N_y filas de píxeles y N_z bandas espectrales.

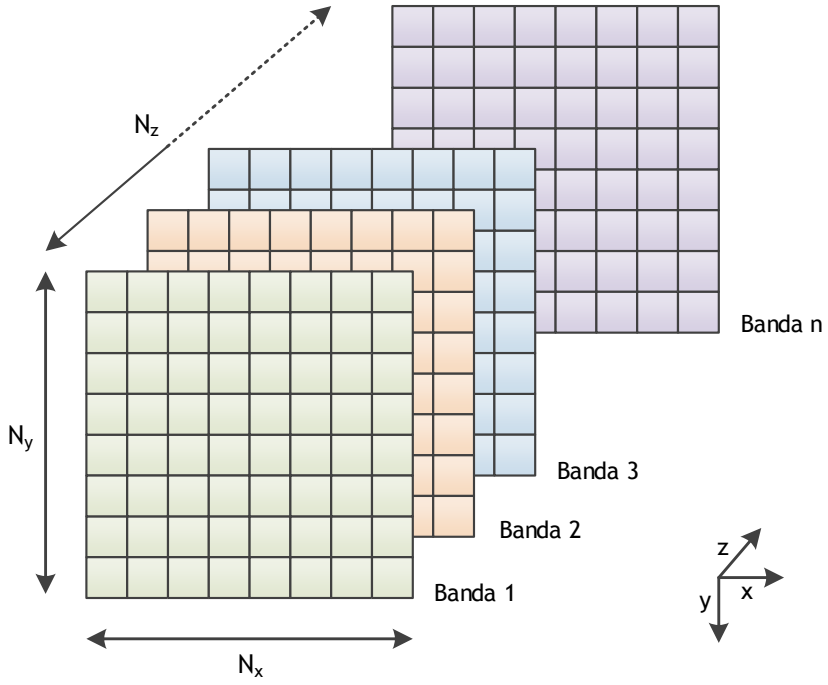


Figura 1.4 Imagen hiperespectral

Un fenómeno característico de este tipo de imágenes es la presencia de distintos tipos de elementos en el mismo píxel, lo que requiere en muchas ocasiones estudios a nivel sub-píxel. También, existen píxeles llamados píxeles puros, que representan un único material, pero en general la mayoría son píxeles mezcla que contienen distintos tipos de materiales [13]. Este fenómeno es producido por la resolución espacial insuficiente de los sensores

hiperespectrales. Es también corriente encontrar en una imagen píxeles mezcla, en los que, independientemente de la escala que se considere, la mezcla se produce a nivel microscópico [14]. El análisis de las imágenes hiperespectrales es de gran interés científico y supone un campo de investigación importante [15] [16] .

1.1.2 Compresión de imágenes hiperespectrales en aplicaciones espaciales

Las imágenes hiperespectrales generan una gran cantidad de información que ha de ser comprimida con el fin de reducir el volumen de datos y aumentar la eficiencia de la transmisión hacia el segmento terreno [17]. Los diferentes tipos de imágenes espectrales difieren en la cantidad de datos disponibles bien en la dimensión espacial o en la dimensión espectral, requiriendo potencialmente diferentes técnicas de compresión. Por ejemplo, las imágenes multiespectrales presentan una resolución espacial con alta granularidad mientras que la granularidad en resolución espectral no es tan alta. Por consiguiente, su compresión explota la correlación espacial. El caso opuesto ocurre con las imágenes de tipo hiperespectrales y ultraespectrales, en las que la correlación espectral se presenta como factor dominante [18].

Existen diferentes técnicas de compresión disponibles. En los algoritmos de compresión sin pérdidas, la imagen reconstruida es idéntica a la original [19] [20]. Los algoritmos de compresión con pérdidas, proporcionan una imagen reconstruida similar a la imagen original, empleándose el valor del error cuadrático medio para una tasa de bits objetivo [21]. Un caso particular de la compresión con pérdidas es la compresión casi-sin pérdidas, en ella, la diferencia máxima absoluta entre la imagen reconstruida y la imagen original no debe exceder un valor máximo predefinido [22].

Los algoritmos de compresión de imágenes hiperespectrales se encuentran limitados por las características de los dispositivos calificados para operar en el entorno espacial, éstos dispositivos, por ejemplo, han presentar estrategias para mitigar los efectos de la radiación ionizante presente en el espacio. Por este motivo, al margen de las características intrínsecas que

poseen las imágenes hiperespectrales, estos algoritmos de compresión han de cumplir con una serie de requisitos adicionales con los que, los algoritmos de compresión desarrollados en otros sectores no se enfrentan.

El conjunto de dispositivos sobre los que pueden ser desplegados los algoritmos de compresión de imágenes hiperespectrales se encuentra limitado. Normalmente, los dispositivos basados en lógica reprogramable de tipo FPGA son la opción preferida para la implementación hardware de los algoritmos de compresión gracias a su bajo consumo, capacidad de procesamiento paralelo y capacidad de reconfiguración. No obstante, inicialmente los algoritmos se encuentran descritos en lenguajes de alto nivel (por ejemplo: C/C++, Java, etc.) capaces de ser ejecutados en dispositivos como procesadores de propósito general con el fin de evaluar su comportamiento [23]. Los dispositivos de tipo ASIC son empleados para la ejecución de algoritmos de compresión de imágenes hiperespectrales cuando se requiere de altas prestaciones y muy bajo consumo. La popularidad de los ASICs en cuanto a su utilización en el sector espacial es menor si se comparan con las FPGAs, ya que presentan costes de producción elevados y no poseen capacidades de reconfiguración. Finalmente, los dispositivos de tipo GPU han demostrado su idoneidad para la ejecución de algoritmos de compresión de imágenes hiperespectrales gracias a su capacidad masiva de procesamiento paralelo. Sin embargo, actualmente no existen dispositivos capaces de operar en el entorno espacial, limitándose de este modo su uso en misiones reales [24].

En este sentido, resulta igualmente importante tanto la eficiencia del algoritmo de compresión propuesto como el de disponer implementaciones sobre los dispositivos calificados para operar en el espacio. Los algoritmos de compresión sin pérdidas han sido empleados mayoritariamente en la compresión de imágenes hiperespectrales, presentando implementaciones capaces de ser desplegadas a bordo de satélites [25] [26] [27]. Actualmente, existe una tendencia marcada por los algoritmos de compresión con pérdidas de imágenes hiperespectrales fruto del incesante incremento de información generada a bordo por sensores hiperespectrales [28]. Por lo tanto, proporcionar implementaciones que demuestren la viabilidad del empleo de

este tipo de algoritmos con pérdidas resulta parte fundamental para completar su desarrollo y que se postulen como alternativas reales para ser operados a bordo. El estudio de las técnicas de compresión con pérdidas de imágenes hiperespectrales y su implementación sobre dispositivos FPGA calificados para operar en el entorno espacial es el tema principal de esta tesis doctoral.

El algoritmo de compresión con pérdidas de imágenes hiperespectrales LCE (del inglés, *Lossy Compression for ExoMars*) tiene como objetivo principal su despliegue a bordo de la misión ExoMars alcanzando altas ratios de compresión [29]. Este algoritmo, de naturaleza predictiva, resulta una alternativa atractiva para la compresión a bordo ya que emplea importantes consideraciones en el diseño para facilitar su desempeño en el entorno espacial. La implementación de este algoritmo sobre un dispositivo de tipo FPGA calificado para operar en el espacio se aborda en esta tesis doctoral estudiando la viabilidad de su inclusión en futuras misiones espaciales.

En [30], se presenta el algoritmo predictivo con control de la tasa de bits que ha sido propuesto como extensión con pérdidas del estándar CCSDS 123.0-B-1 [25] bajo las premisas de baja complejidad y bajos requisitos de almacenamiento. Este algoritmo aporta un nuevo mecanismo de control de la tasa de bit que puede resultar viable para su empleo a bordo. En esta tesis doctoral se evalúa el desempeño de este algoritmo, en el que, además, se han incluido una serie de aportaciones encaminadas a reducir su complejidad. Asimismo, se realiza su implementación con el objetivo de estudiar la viabilidad de su operación a bordo de futuros satélites de observación de la tierra.

Dentro de los algoritmos basados en transformada, la transformada ortogonal por parejas, en adelante POT, destaca como decorrelador espectral en algoritmos de compresión de imágenes hiperespectrales basados en transformada [31] [32]. Generalmente, la principal desventaja de los algoritmos basados en transformadas es su mayor complejidad para ser implementados a bordo. La transformada POT, gracias a su reducida complejidad, se presenta como una alternativa atractiva a ser implementada

como decorrelador espectral en los sistemas de compresión de imágenes hiperespectrales a bordo de satélites de observación de la tierra. En esta tesis doctoral se estudia en detalle la complejidad de este algoritmo y se realiza su implementación sobre dispositivos de tipo FPGA calificados para espacio con el fin de evaluar su viabilidad para ser desplegado a bordo de un satélite.

1.2 Motivación de la Tesis Doctoral

La compresión de imágenes hiperespectrales se ha convertido en un foco de investigación a lo largo de los últimos años debido, principalmente, a la necesidad de adecuar la capacidad de adquisición de los sensores hiperespectrales con las capacidades de comunicación de los enlaces a tierra. En este sentido, se han propuesto numerosos algoritmos de compresión de imágenes hiperespectrales empleando diferentes técnicas dentro del estado del arte. La mayor parte de estos algoritmos han sido descartados para su empleo a bordo de satélites de observación o bien por sus limitaciones en cuanto a prestaciones de compresión, es decir, la ratio entre la imagen original y la imagen comprimida, o por su alta complejidad computacional, no pudiendo ser desplegados en los dispositivos calificados para operar en el entorno espacial.

El Comité Consultivo para los Sistemas de Datos Espaciales, en adelante CCSDS (del inglés, *Consultative Committee for Space Data Systems*), tiene dentro de sus objetivos la publicación de estándares, recomendaciones e informes con el fin de promocionar la interoperabilidad y soporte entre las diferentes agencias espaciales, mejorar la colaboración y promover nuevas capacidades en el sector espacial. El CCSDS proporciona una solución para la compresión sin pérdidas de imágenes multiespectrales e hiperespectrales a bordo de satélites, el estándar CCSDS 123.0 [25]. Es notorio que, la implementación de algoritmos de compresión de imágenes hiperespectrales sin pérdidas ha sido tradicionalmente preferida para su despliegue a bordo, ya que se preserva la integridad de toda la información de la imagen original. Sin embargo, dado que el ancho de banda de transmisión hacia tierra es limitado y, que además, existen restricciones en la cantidad de almacenamiento disponible a bordo para imágenes hiperespectrales, la compresión con pérdidas de este tipo de imágenes resulta necesaria con el fin de reducir el volumen de datos a almacenar y/o transmitir.

En la actualidad, no existe un algoritmo estandarizado para la compresión con pérdidas de imágenes hiperespectrales, si bien el comité CCSDS está trabajando en un nuevo estándar. En este sentido, existen dos

iniciativas paralelas que se pretenden llevar a cabo. En primer lugar, se plantea extender el estándar existente de compresión de imágenes CCSDS 122.0-B-1 [33] para su empleo en imágenes hiperespectrales. El estándar CCSDS 122.0 puede ser empleado en la compresión tanto sin pérdidas como con pérdidas de imágenes en dos dimensiones (2D). Si se emplea este estándar en conjunto con una transformada espectral, se puede hacer uso de la correlación espectral presente en las imágenes hiperespectrales para conseguir una compresión eficiente. En segundo lugar, se propone realizar una extensión con pérdidas del estándar CCSDS 123.0 con mecanismos para el control de la tasa de bits.

Por lo tanto, el hecho de que el CCSDS se encuentre trabajando actualmente hacia la definición de diferentes alternativas que se conviertan en estándares de compresión con pérdidas, refuerza la necesidad de proporcionar implementaciones que confirmen la viabilidad de los algoritmos propuestos para ser desplegados en dispositivos calificados para operar a bordo de un satélite. En esta tesis doctoral se realiza la implementación sobre dispositivos de tipo FPGA calificados para el espacio de dos de las alternativas estudiadas en seno del CCSDS para formar parte de futuros estándares de compresión de imágenes hiperespectrales: la transformada POT y el algoritmo predictivo con control de la tasa de bits. Estas implementaciones ayudarán en la evaluación de la viabilidad de este tipo de algoritmos con pérdidas como alternativas reales para ser operados a bordo.

La implementación hardware sobre dispositivos de tipo FPGA de los algoritmos de compresión de imágenes hiperespectrales resulta un reto dada la complejidad que actualmente presentan estos algoritmos. La aproximación tradicional a la implementación hardware mediante la descripción del circuito a nivel RTL (del inglés, *Register Transfer Level*) requiere del conocimiento y experiencia en el uso de lenguajes de descripción hardware, tales como VHDL o Verilog, y de un flujo de diseño específico. Este flujo proporciona la implementación del diseño una vez realizada tanto la etapa de síntesis lógica como la de emplazamiento y rutado. Asimismo, es necesaria la verificación del diseño con el objetivo de garantizar que el algoritmo continúa funcionando correctamente tras las diferentes etapas del flujo de diseño.

El tiempo de desarrollo del flujo de diseño RTL puede acortarse mediante el empleo de metodologías y herramientas para la síntesis de alto nivel, en adelante HLS (del inglés, *High Level Synthesis*). Estas metodologías de diseño no alcanzan los resultados de implementación que pueden alcanzarse mediante las implementaciones realizadas a nivel RTL. Sin embargo, suponen un paso adelante en cuanto a la implementación de algoritmos complejos aumentando el nivel de abstracción desde el que se realiza la implementación.

Las herramientas de síntesis de alto nivel permiten la generación de código a nivel RTL (VHDL o Verilog) desde lenguajes como C/C++, SystemC o Matlab entre otros. En la actualidad, existen una gran cantidad de herramientas de síntesis de alto nivel proporcionadas por distintas empresas [34]. En esta tesis doctoral se ha escogido la herramienta *CatapultC High-Level Synthesis* ya que los algoritmos de compresión implementados se encuentran escritos en C/C++ y esta herramienta posibilita la generación de código RTL desde este lenguaje [35]. Por lo tanto, la implementación de algoritmos de compresión capaces de ser empleados a bordo de satélites mediante la síntesis de alto nivel es un desafío abordado en esta tesis doctoral.

1.3 Objetivos de la Tesis Doctoral

Como ha quedado descrito, existe la necesidad de aumentar la eficiencia de los algoritmos de compresión de imágenes hiperespectrales a bordo de satélites de observación con el objetivo de optimizar el ancho de banda de transmisión a tierra, así como, para alcanzar mayores ratios de compresión mediante la introducción de pérdidas en el proceso de compresión. Para lograr este objetivo, resulta de igual importancia tanto la definición del propio algoritmo de compresión como la viabilidad de su implementación para ser desplegado a bordo.

Por lo tanto, los objetivos de este trabajo de tesis doctoral son los siguientes:

- Estudio de viabilidad de la inclusión en un sistema de compresión a bordo de los algoritmos de compresión de imágenes hiperespectrales con pérdidas actuales así como de los estándares recomendados.
- Aportar implementaciones hardware sobre dispositivos calificados para espacio de algoritmos de compresión con pérdidas de imágenes hiperespectrales que demuestren su viabilidad para ser desplegados a bordo de futuras misiones espaciales. Se plantea la implementación de los siguientes tres algoritmos de compresión:
    - El algoritmo de compresión con pérdidas para la misión ExoMars, denominado LCE.
    - La transformada Ortogonal por Parejas (POT).
    - Algoritmo ligero de control de la tasa de bits propuesto como extensión con pérdidas del estándar CCSDS 123.0.
- Empleo de metodologías de síntesis de alto nivel para la generación del código RTL de los algoritmos de compresión implementados.
- Análisis comparativo de los resultados obtenidos con los disponibles en el estado del arte, mostrando la bondad de las implementaciones realizadas.

1.4 Estructura del documento

La estructura de esta Tesis Doctoral se compone de seis capítulos que dan forma a los trabajos desarrollados y cuyos contenidos se describen a continuación:

Capítulo 1. Introducción

Se realiza el planteamiento del problema al que los trabajos desarrollados pretenden dar respuesta. Se presentan las motivaciones que han derivado en la realización de esta tesis doctoral, presentando asimismo los objetivos a alcanzar en la tesis doctoral. Con tal fin, se define el concepto de imagen hiperespectral y se contextualiza la necesidad de su compresión a bordo de satélites.

Capítulo 2. Estado del arte

En este capítulo se presentan los algoritmos de compresión de imágenes hiperespectrales tanto sin pérdidas como con pérdidas disponibles en la literatura. Para ello, se establecen las principales características que han de tener estos algoritmos para poder operar a bordo. Además, se presentan las principales aportaciones en términos de implementación sobre dispositivos calificados para operar en el espacio con el fin de establecer, posteriormente, comparativas entre los distintos resultados obtenidos en esta tesis doctoral.

Capítulo 3. Implementación del algoritmo de compresión con pérdidas para la misión ExoMars

La implementación del algoritmo de compresión con pérdidas para la misión ExoMars, denominado LCE, se aborda en este capítulo. Se detalla, en primer lugar, el funcionamiento del algoritmo para, posteriormente, presentar la implementación realizada. Asimismo, se concluye con los resultados obtenidos tras la implementación.

Capítulo 4. Implementación Hardware de las Operaciones Aritméticas de la Transformada Ortogonal por Parejas

A lo largo de este capítulo se describe la transformada ortogonal por parejas resaltando su empleo como etapa de eliminación de la correlación espectral de una imagen hiperespectral. Para ello, se describe el algoritmo que compone la transformada ortogonal por parejas y las simplificaciones realizadas para su posible implementación hardware. Finalmente, se presentan las consideraciones de diseño y los resultados obtenidos en la implementación.

Capítulo 5. Implementación del estándar CCSDS 123.0 con pérdidas y control de la tasa de bits

En este capítulo, se presenta la implementación de un algoritmo de compresión basado en el predictor descrito en el estándar CCSDS 123.0-B-1 en el que se incluye una etapa dedicada para el control de la tasa de bits y diferentes alternativas en cuanto al codificador entrópico. Además, se presentan los resultados de implementación obtenidos y se analizan comparativamente tanto con los resultados disponibles en el estado del arte como los obtenidos en los capítulos anteriores.

Capítulo 6. Conclusiones y líneas futuras

Este capítulo se dedica a la presentación de las conclusiones y contribuciones propuestas a partir del trabajo desarrollado en esta tesis doctoral. Asimismo, se presentan las líneas futuras de investigación que dan continuidad a estos trabajos.

Capítulo 2

Estado del Arte

La compresión de imágenes hiperespectrales en aplicaciones espaciales supone un reto dentro del campo de estudio de la compresión de datos. Estas imágenes presentan características singulares que incrementan la complejidad de los algoritmos de compresión a emplear. Asimismo, su desempeño a bordo de satélites ha de considerar requisitos adicionales determinados por las características propias del entorno espacial y de la misión para la cual esté diseñada.

En este capítulo se recogen las principales necesidades de los algoritmos de compresión de imágenes hiperespectrales para ser desplegados a bordo de satélites. Con tal fin, se abordan las principales características que han de presentar los algoritmos, se analizan los principales algoritmos existentes y se describen los dispositivos cualificados para operar en el entorno espacial que pueden ser empleados para su implementación.

2.1 Introducción

La detección a distancia o teledetección (en inglés, *Remote Sensing*) es actualmente un campo de investigación activo en donde la comunidad científica genera una cantidad creciente de información permitiendo el desarrollo de un nuevo nicho de aplicaciones. En este sentido, la variedad y cantidad de sensores de alta resolución desplegados en los satélites de observación, generando grandes volúmenes de datos, suponen un reto a la hora de transmitir esa información hacia el segmento terreno. Como se presenta en el capítulo 1, el canal de comunicaciones para transmitir la información desde los sensores a bordo de un satélite hacia el segmento terreno es limitado, por lo que, se requieren de nuevos protocolos de transmisión para usar de manera eficiente el canal de comunicaciones disponible.

La compresión de la información generada por los sensores resulta una estrategia eficiente en el incremento de la tasa de datos de transmisión sobre un mismo canal de comunicaciones. La compresión de imágenes hiperespectrales no permanece al margen de otros tipos de datos generados por sensores de observación a bordo de satélites, siendo necesario la investigación e implementación de nuevos algoritmos dentro del sistema de procesado de datos del satélite. Se pueden establecer diferentes tipos de clasificaciones en cuanto a algoritmos de compresión de datos en el sector espacial, que resultan de igual aplicabilidad en el caso particular de la compresión de imágenes hiperespectrales [11]. De este modo, los algoritmos pueden ser clasificados teniendo en cuenta la introducción o no de pérdidas en la imagen reconstruida. Mediante este criterio, se disponen de algoritmos sin pérdidas, con pérdidas y casi-sin pérdidas, siendo este último un caso particular de los algoritmos con pérdidas. Otro tipo de clasificación se basa en el tipo de técnica de compresión empleada. En este caso se pueden encontrar algoritmos de tipo predictivo, algoritmos basados en transformadas y algoritmos basados en cuantificación vectorial. Además, dentro de una misión espacial en donde la interoperabilidad entre las distintas agencias espaciales es fundamental, resulta necesario conocer si un algoritmo se

encuentra basado en un estándar aprobado para operar en el entorno espacial, si se trata de un algoritmo desarrollado *ad hoc* para las necesidades concretas de una misión o si se trata de una algoritmo desarrollado en otro sector y aplicado en el entorno espacial (por ejemplo, JPEG 2000 [36]). La clasificación seguida en este capítulo para cada uno de los algoritmos de compresión de imágenes hiperespectrales analizados considerará, en primer lugar, si se introducen (o no) pérdidas en la imagen reconstruida para luego detallar la técnica de compresión empleada y si forman parte (o no) de un estándar de compresión publicado previamente.

2.2 Algoritmos de compresión de imágenes hiperespectrales

La compresión de datos es una de las tareas de mayor importancia dentro de las funciones del sistema de procesado de datos realizadas a bordo de un satélite [11]. Por ello, antes de presentar en detalle los algoritmos más relevantes empleados para la compresión de imágenes hiperespectrales, resulta imprescindible describir las características que han de presentar para ser desplegados a bordo.

2.2.1 Principales características

El sistema de procesado de datos a bordo se encuentra limitado, generalmente, en términos de capacidad computacional, ya que el consumo de potencia, dimensiones y masa disponible para el mismo está asimismo limitado. Además, los dispositivos que se encuentran calificados para operar en el entorno espacial, y que resultan adecuados para ser empleados en el sistema de procesado de datos, presentan técnicas para la mitigación de los efectos de la radiación presente en el entorno espacial que reducen sus capacidades computacionales. Por este motivo, los algoritmos de compresión de imágenes hiperespectrales han de presentar una baja complejidad computacional para ser desplegados a bordo.

Los algoritmos de compresión de imágenes hiperespectrales se encuentran dentro de la plataforma satelital a la salida del sensor hiperespectral cuyos datos han de comprimir. Por lo tanto, es necesario que los algoritmos de compresión sean capaces de procesar datos en crudo (en inglés, *Raw Data*) o no calibrados provenientes del sensor hiperespectral. Esto implica que los algoritmos han de ser capaces de procesar imágenes hiperespectrales que presenten imperfecciones, por ejemplo, ruido en la imagen o aparición de rayas (en inglés, *striping*) entre otros, que provocan una reducción en la eficiencia de compresión del algoritmo.

La transmisión de datos desde una plataforma satelital de observación hacia el segmento terreno sufre errores o incluso pérdidas de paquetes en el

proceso de comunicación. Por este motivo, los algoritmos de compresión de imágenes hiperespectrales han de proporcionar mecanismos para proporcionar tolerancia a errores, es decir, se ha de reconstruir en el segmento terreno la mayor parte de información disponible sin errores. En este sentido, una aproximación común es la de reiniciar el algoritmo de compresión una vez alcanzado cierto número de unidades de procesado básicas (por ejemplo: líneas de píxeles o bloques de píxeles) de tal forma que si se produce un error, este afecte únicamente a las unidades de procesados comprimidas entre reinicios [37].

En el momento en el que se introducen pérdidas en la compresión de los datos a bordo, resulta necesario cuantificar la calidad de la imagen reconstruida y si, esta imagen reconstruida sigue siendo válida para el propósito científico para el que fue diseñado [38]. Los criterios de calidad de la imagen pueden relacionarse, por lo tanto, con las pérdidas de información causadas por la compresión de los datos. En el caso de imágenes en dos dimensiones (2D), los criterios de calidad han de reflejar la percepción visual del observador humano. En el caso de imágenes hiperespectrales, los criterios de calidad se encuentran asociados a la aplicación final de los datos capturados. Entre los distintos criterios disponibles, el indicador de calidad de imagen empleado es la relación señal a ruido de pico, PSNR (del inglés, *Peak Signal-to-Noise Ratio*). El PSNR, también denominado ratio tasa de bits – distorsión, se obtiene mediante el cociente entre error cuadrático medio, MSE (del inglés, *Mean Squared Error*) y el valor máximo que pueda alcanzar un píxel en la imagen a comprimir, es decir, 2^b, en donde b es igual al valor de bits por píxel definido en cada imagen. El cálculo de los valores de MSE y PSNR se describe en las ecuaciones (2.1) y (2.2), en donde $s_{z,y,x}$ es el valor de la muestra actual, $\hat{s}_{z,y,x}$ es el valor de la muestra reconstruida, N_z representa el número de bandas de la imagen, N_y es el número de filas de la imagen y N_x representa el número de columnas de la imagen.

$$MSE = \frac{1}{N_z N_y N_x} \sum_{z=0}^{N_z-1} \sum_{y=0}^{N_y-1} \sum_{x=0}^{N_x-1} \left| s_{z,y,x} - \hat{s}_{z,y,x} \right|^2 \qquad (2.1)$$

$$PSNR = 10 \cdot log_{10}\left(\frac{2^b - 1}{MSE}\right) \quad (2.2)$$

2.2.2 Algoritmos de compresión sin pérdidas

En los algoritmos de compresión sin pérdidas, la imagen reconstruida es idéntica a la original [19] [20]. Tradicionalmente, la compresión de imágenes hiperespectrales, ha sido realizada mediante algoritmos de compresión sin pérdidas. Estos algoritmos posibilitan que, tras la compresión, los datos sean reconstruidos sin pérdida alguna de información, es decir, la imagen reconstruida es idéntica a la original. La compresión sin pérdidas resulta la deseada ya que se preserva la integridad de toda la información contenida en la imagen. Sin embargo, los ratios de compresión que se obtienen mediante la aplicación de estos algoritmos son limitados. Los mejores algoritmos de compresión sin pérdidas proporcionan ratios de compresión típicamente del orden de 2:1 a 3:1 [39] [40] [41].

Independientemente del algoritmo de compresión sin pérdidas empleado, la compresión de la información contenida presenta el límite definido por el teorema de Entropía de Shannon [42]. Este teorema establece que, dada una secuencia de eventos $e_1, e_2, e_3, \ldots, e_n$ y la distribución de probabilidad P de ocurrencia de dichos eventos, el menor número de bits necesarios para su codificación viene determinado por la ecuación (2.3). En donde, $p\{e_k\}$ representa la probabilidad de que el evento $\{e_k\}$ ocurra.

$$H(P) = \sum_{k=1}^{n} -p\{e_k\} \log_2 p\{e_k\} \quad (2.3)$$

La compresión de imágenes hiperespectrales sin pérdidas ha estado basada, generalmente, en el paradigma de la codificación predictiva, en donde cada pixel es predicho mediante datos anteriormente procesados y en donde el error de predicción se codifica entrópicamente [19]. Se emplea por tanto la codificación DPCM (del inglés, *Differential Pulse Code Modulation*) como base de este tipo de codificadores [43].

Existen métodos de predicción relativamente simples como el denominado LUT-NN (del inglés, *Look-Up Table - Nearest Neighbour*), que realizan la predicción del pixel actual tomando el valor del vecino más cercano transmitido en la banda actual [44]. Este algoritmo alcanza ratios de compresión comparables a algoritmos predictivos con una mayor complejidad. Sin embargo, estas prestaciones se alcanzan únicamente en los casos en el que la imagen de entrada se encuentre calibrada, ya que en ese caso el proceso de calibración proporciona histogramas en los que la frecuencia de algunos valores de píxel se incrementa por encima de otros. Esta eficiencia de compresión artificial en cuanto a la compresión de imágenes hiperespectrales calibradas ha sido objeto de mayor investigación dando lugar al algoritmo predictivo LAIS-LUT (del inglés, *Locally Averaged Interband Scaling - LUT*) [45]. Este algoritmo emplea dos LUTs en las que se almacenan los dos valores más cercanos al valor del píxel actual dentro de la vecindad cercana. El algoritmo LAIS-LUT alcanza mejores prestaciones de codificación que el algoritmo LAIS-NN a expensas de un incremento en la complejidad computacional. Estos algoritmos basados en LUT han sido desarrollados *ad hoc* en un entorno de investigación no siendo estandarizados o empleados en misiones reales dadas sus limitaciones para el procesado de imágenes no calibradas.

En [46] se introduce el concepto de predicción difusa para la compresión de imágenes hiperespectrales. Este método emplea un predictor adaptativo espacial-espectral junto con clasificadores basados en contexto y un codificador aritmético como codificador entrópico para la codificación de los residuos. La introducción de la agrupación diferencial como base para la codificación, concepto denominado C-DPCM (del inglés, *Clustered Differential Pulse Code Modulation*), se presenta en [47]. En este algoritmo, la imagen es dividida en agrupaciones de vectores espectrales para cada banda, empleando un predictor lineal de mínimos cuadrados para cada agrupación de cada banda. Los algoritmos [46] y [47] pertenecen a tipos de algoritmos basados en la cuantificación vectorial no formando parte de ningún estándar de compresión en el sector espacial, pudiendo ser clasificados como algoritmos *ad hoc* dentro de las clasificaciones empleada en este documento.

Se han propuesto métodos de compresión de imágenes hiperespectrales basados en la codificación de imagen sin pérdidas adaptativa en contexto, en adelante CALIC (del inglés, *Context-Adaptive Lossless Image Coding*). En [48] se presenta el algoritmo 3-D CALIC el cual elige entre el modo de predicción intra-banda o inter-banda basándose en la correlación existente entre bandas consecutivas. En el método multi-banda, denominado M-CALIC [49], la predicción se realiza mediante el empleo de la pareja de píxeles de bandas anteriores que se encuentran situados en la misma posición espacial del pixel actual. Los coeficientes empleados durante la predicción son calculados a priori sobre un conjunto de datos de entrenamiento. En [50] se presenta la técnica de predicción adaptativa basada en mínimos cuadrados denominada SLSQ (del inglés, *Spectrum-oriented Least SQuares*). En ella, se emplea un predictor optimizado para cada píxel en cada banda usando la vecindad del píxel actual. De la misma forma que en los casos anteriores, estos algoritmos de compresión de tipo predictivo han sido desarrollados *ad hoc* y no han sido empleados como base para el desarrollo de estándares de compresión por su complejidad para ser implementado en los dispositivos disponibles a bordo de un satélite.

El primer estándar de compresión de datos sin pérdidas publicado por el comité CCSDS es el denominado CCSDS 121.0-B-1 [51]. Se trata de un estándar de compresión de datos genérico centrado en su aplicación en el sector espacial. Consta de un preprocesador y un codificador entrópico basado en la codificación Rice [52]. El objetivo del preprocesador es el de disminuir la entropía de los datos de entrada mediante la modificación del comportamiento estadístico de los datos. El estándar recomendado no establece el tipo de algoritmo que ha de emplearse en el preprocesador dándole libertad al usuario para particularizar la etapa de procesado a las características de los datos a comprimir. El compresor entrópico procesa bloques de J-muestras incorporando varias opciones de codificación: Golomb potencia de dos, identificación de bloques nulos y bloques sin compresión. Se ha demostrado que las prestaciones de este algoritmo decrecen en presencia de valores atípicos en los datos de origen. Con el fin de mejorar esta limitación, el algoritmo denominado FAPEC (del inglés, *Fully Adaptive*

Prediction Error Coder) [53] presenta una eficiencia de compresión mayor que el estándar CCSDS 121.0 junto con una baja complejidad computacional convirtiéndolo en una alternativa viable para ser desplegado a bordo de un satélite. Ninguno de estos dos algoritmos ha sido concebido para la compresión imágenes, por lo que no explotan las características propias de este tipo de datos.

El estándar de compresión CCSDS 122.0-B-1 [33], aborda la compresión tanto sin pérdidas como con pérdidas de imágenes en dos dimensiones generadas por sensores a bordo de satélites. Forma parte de los algoritmos de compresión basados en transformadas, empleando la transformada DWT (del inglés, *Discrete Wavelet Transform*) para eliminar la correlación de las muestras de la imagen seguido de un codificador BPE (del inglés, *Bit Plane Encoder*) para proporcionar la imagen comprimida. La aplicación de la transformada unidimensional DWT de forma sucesiva, resulta en una transformada 2D-DWT de tres niveles. Dentro del estándar, queda abierta la elección entre la aproximación en punto flotante o entera de la transformada DWT. Los coeficientes de salida son convertidos a valores enteros antes del codificador BPE. Este estándar emplea como base la transformada DWT, de la misma manera que el estándar JPEG2000. Sin embargo, la complejidad computacional del estándar CCSDS 122.0 es significativamente inferior permitiendo su implementación hardware en los dispositivos cualificados para operar en el entorno espacial [54].

En 2007 se estableció el grupo de trabajo de compresión de datos multiespectrales e hiperespectrales con el fin de desarrollar un estándar de compresión de imágenes multiespectrales e hiperespectrales que pueda ser desplegado a bordo de un satélite [55]. Se estableció como prioridad la naturaleza sin pérdidas del algoritmo a desarrollar. La solución de consenso elegida entre las distintas agencias espaciales fue tomar como punto de partida el algoritmo de compresión denominado FL (del inglés, *Fast Lossless*) [56] [57]. El algoritmo FL realiza la predicción a partir de las muestras previamente procesadas tanto de la banda actual como de tres bandas anteriores. Este modo de procesamiento, en donde únicamente se emplea información causal, proporciona un filtrado adaptativo de baja complejidad

en conjunto con unas altas prestaciones de compresión [55]. El resultado de este trabajo ha sido el estándar de compresión sin pérdidas de imágenes multiespectrales e hiperespectrales CCSDS 123.0-B-1 [25] que tiene como propósito principal el establecer un estándar recomendado para la compresión de imágenes tridimensionales generadas por los sensores hiperespectrales a bordo de satélites. Este estándar si bien se basa en el compresor FL, emplea únicamente aritmética entera. El estándar se concibió con el objetivo de ser desplegado a bordo de un satélite por lo que se han tenido en cuenta tanto el requisito de baja complejidad computacional como el de baja necesidad de almacenamiento interno que, dadas las características de los dispositivos hardware disponibles a bordo, resultan fundamentales.

El estándar de compresión de imágenes hiperespectrales CCSDS 123.0 ha sido comparado en la compresión de imágenes hiperespectrales con los estándares de referencia de compresión de imagen JPEG2000 y JPEG-LS [55]. El algoritmo JPEG-LS es un estándar ISO (del inglés, *International Organization for Standardization*) basado en el algoritmo denominado LOCO-I (del inglés, *Low Complexity Lossless Compression for Images*) [55]. El algoritmo de compresión LOCO-I, se compone de un predictor adaptativo de baja complejidad en donde la muestra actual se predice mediante las muestras procesadas con anterioridad, codificándose entrópicamente los residuos. El algoritmo JPEG-LS soporta la compresión de imágenes 3D, pudiendo ser empleado en la compresión de imágenes hiperespectrales. Los resultados muestran la menor eficiencia de compresión del estándar JPEG-LS si se compara con los resultados proporcionados por el estándar CCSDS 123.0-B-1 [55].

Por otra parte, el estándar de compresión de imágenes JPEG2000, basado en la transformada DWT, puede ser empleado para la compresión de imágenes 3D gracias a su extensión MCT (del inglés, *Multiple Component Transformation*). En primer lugar, se realiza una transformada espectral para, a continuación, realizar la compresión bidimensional de cada banda sin correlación espectral. El principal inconveniente de esta aproximación es la complejidad computacional asociada a la aplicación sucesiva de varias transformadas. Por el contrario, aporta la definición de áreas de interés sobre

la imagen, lo que permite la menor compresión de las regiones dentro de la imagen que tengan mayor valor a nivel científico. Los resultados obtenidos tras la compresión de imágenes hiperespectrales concluyen que, pese a los beneficios aportados por el estándar JPEG2000-MCT, su mayor carga computacional asociada y su menor eficiencia de compresión, hacen que el estándar CCSDS 123.0-B-1 sea la referencia para la compresión sin pérdidas de imágenes hiperespectrales a bordo de satélites [21].

2.2.3 Algoritmos de compresión con pérdidas

En la actualidad el CCSDS se encuentra en fase de definición de un estándar para la compresión de imágenes hiperespectrales con pérdidas. La compresión con pérdidas de imágenes hiperespectrales se ha convertido en objeto de estudio dada la necesidad actual de mayores ratios de compresión. Los algoritmos de compresión con pérdidas alcanzan mayores ratios de compresión que aquellos que son sin pérdidas a expensas de una pérdida de información durante el proceso. A pesar de la pérdida de calidad en la imagen reconstruida, este tipo de algoritmos son de gran utilidad especialmente cuando se requieren grandes ratios de compresión. En [17] se evalúa el efecto de dichas pérdidas en imágenes hiperespectrales para aplicaciones específicas tales como la detección o clasificación de objetivos, mostrando que se pueden alcanzar grandes tasas de compresión con impactos menores con respecto a la funcionalidad objeto. Por otra parte, la compresión casi-sin pérdidas, puede ser clasificada como un subconjunto de algoritmos con pérdidas en donde la diferencia máxima absoluta entre la imagen reconstruida y la imagen original no exceden un valor máximo predefinido [22].

Existen en la literatura numerosos algoritmos desarrollados para la compresión con pérdidas de imágenes hiperespectrales, muchos de ellos resultan generalizaciones de algoritmos existentes para la compresión de imágenes en 2D o algoritmos para la compresión de vídeo.

Los métodos de compresión de imagen basados en transformadas son conocidos por ser adecuados para su empleo en algoritmos de compresión con pérdidas. Del mismo modo, son conocidos por ser computacionalmente

complejos en términos del número de operaciones involucradas y en las necesidades de recursos de almacenamiento que requieren. El estándar JPEG2000-MCT presentado en [36] y descrito en la sección anterior, ha sido empleado en la compresión con pérdidas de imágenes hiperespectrales. Algoritmos como el denominado SPIHT (del inglés, *Set Partitioning in Hierarchical Trees*) y sus variaciones SPIHT-2D y SPIHT-3D, así como, algoritmos basados en la aplicación 3-D de la transformada DWT y la transformada DCT (del inglés, *Discrete Cosine Transform*) han sido evaluados, igualmente, en la compresión de imágenes hiperespectrales [58] [59] [60] [61]. El algoritmo ICER [62] basado en la transformada DWT para la compresión de imágenes en 2D, ha sido extendido para la compresión 3D de imágenes hiperespectrales dando lugar al algoritmo ICER-3D [63]. Este algoritmo proporciona buenos ratios de compresión, sin embargo, su principal desventada es que otros algoritmos con una complejidad menor, como por ejemplo el CCSDS 123.0, alcanzan similares ratios de compresión. En [64] se introduce la transformada KLT (del inglés, *Karhunen-Loève Transform*) para la compresión de imágenes hiperespectrales. Por otra parte, la descomposición de Tucker para los coeficientes se emplea en [65] en conjunto con la transformada DWT.

Asimismo, se ha estudiado el empleo de diferentes transformadas para la eliminación de la redundancia espectral y la redundancia espacial separadamente. La aproximación seguida es el empleo de la transformada KLT, la transformada DWT o la transformada DCT, en la eliminación de la correlación espectral y emplear el estándar de compresión JPEG2000 en la eliminación de la correlación espacial [66]. En [67] y [68] se emplea la transformada SPIHT-3D usando como transformada espectral la transformada DWT. Los resultados obtenidos en las aproximaciones descritas reflejan que estos algoritmos no resultan candidatos para su despliegue a bordo de un satélite dada su alta carga computacional, sin embargo, sí resultan de utilidad en la compresión en tierra de imágenes hiperespectrales [21].

En [59] se comparan varias técnicas tridimensionales (3-D) de compresión con pérdidas de imágenes hiperespectrales, mostrando que, cuando se emplea la transformada KLT, se mejoran las prestaciones de

compresión en términos de la relación entre la ratio de compresión y la degradación de la imagen reconstruida, especialmente en la región de baja tasa de bits, donde la compresión es más elevada. Por este motivo, ha sido empleada en múltiples aportaciones para la eliminación de la correlación espectral dentro de la compresión de imágenes hiperespectrales [69] [70] [71] [72] [73]. Sin embargo, la transformada KLT presenta importantes desventajas que dificultan su implementación hardware debido a su alta carga computacional y sus elevados requisitos de memoria [74] [75]. En [74] se analiza en profundidad la carga computacional de la transformada KLT concluyendo que resultan necesarias $N_x \, x \, N_y \, x \, N_z^2$ operaciones de multiplicación para obtener el producto matricial de los de los autovalores y los valores normalizados de las muestras de entrada, en donde N_x representa el número de píxeles contenidos en una fila de una imagen, N_y el número de filas de una imagen y N_z el número de bandas de la imagen.

Aportaciones recientes tratan de proporcionar aproximaciones que reduzcan la carga computacional de la transformada KLT. Este es el caso de la transformada ortogonal por parejas, POT (del inglés, del inglés, *Pairwise Orthogonal Transform*) [31] [32]. Esta transformada, se basa en la aplicación de la estrategia *divide-y-vencerás* a la transformada KLT. La transformada resultante es una composición de transformadas KLTs en la que cada una emplea, únicamente, dos componentes de la imagen como entrada. La transformada POT presenta un coste computacional sensiblemente menor que la transformada KLT, estando el número de operaciones determinado por la expresión $12 \, x \, N_z \, x \, N_y \, x \, N_x$ [31]. Este algoritmo nace con la idea de ser empleado como etapa de eliminación de la correlación espectral en la extensión del estándar CCSDS 122.0 en la que actualmente se encuentra trabajando dicho comité. En el capítulo 4, se presenta en detalle la transformada POT y se propone una implementación hardware capaz de ser desplegada a bordo de un satélite sobre los dispositivos calificados a operar en el entorno espacial.

Una visión diferente para la compresión de imágenes hiperespectrales se propone en [76], donde se aplican las técnicas empleadas en el estándar

de compresión de video H.264/AVC consiguiéndose resultados significativos en términos de tasa de bit frente a la distorsión introducida. Sin embargo, este tipo de algoritmos de compresión presentan unas necesidades en cuanto a memoria de almacenamiento y capacidad de cómputo disponible a bordo que resultan demasiado complejas para una misión de larga duración.

El concepto de cuantificación vectorial se emplea en [77] para la compresión sin pérdidas de imágenes hiperespectrales. Si bien puede ser empleado en la compresión sin pérdidas, es en la compresión con pérdidas y casi-sin pérdidas en donde mayor desarrollo ha experimentado [78] [79] [80]. La cuantificación vectorial consta de dos pasos: un primer paso de entrenamiento, en donde se crea el catálogo de palabras código, y un segundo paso en el que a cada vector se le asigna una palabra código. Los resultados en cuanto a eficiencia de compresión muestran ratios de compresión en torno a 10:1, llegando a máximos de 30:1 con distorsiones aceptables en la aplicación objetivo [78]. La carga computacional asociada en este tipo de algoritmos hace que resulten opciones de difícil implementación hardware a bordo de un satélite. Por este motivo, los principales esfuerzos de investigación se centran fundamentalmente en la reducción de la complejidad computacional de los algoritmos basados en cuantificación vectorial.

Recientemente, se ha formado un nuevo paradigma basado en algoritmos de baja complejidad para la compresión con pérdidas de imágenes hiperespectrales. Estos algoritmos se encuentran basados en el esquema DPCM característico de los algoritmos de compresión sin pérdidas predictivos [21]. De esta forma presentan una primera etapa de predicción, seguida de un cuantificador y finalizando con una etapa de codificación entrópica. Asimismo, este tipo de algoritmos pueden presentar características adicionales como la optimización del binomio tasa de bits de salida - distorsión en la imagen reconstruida, el control de la tasa de bits, la definición de regiones de interés o incluso la operación casi-sin pérdidas si se limita mediante un parámetro las pérdidas máximas a introducir en la imagen reconstruida. Este tipo de algoritmos aprovechan la simplicidad y las altas prestaciones de los esquemas de compresión basados en la predicción empleando pocos accesos a memoria. En [28] se presenta el algoritmo de

compresión de imágenes hiperespectrales FLEX (del inglés, *Fast Lossless EXtended*) que supone una evolución casi sin pérdidas del algoritmo de compresión sin pérdidas FL. El algoritmo FLEX aporta un esquema de codificación entrópica híbrida en donde se emplea o bien un codificador entrópico Golomb potencia de 2 para datos con valores de entropía altos o bien una etapa de codificación condicional de dos pasos para datos con valores de entropía bajos. Este algoritmo, como se presenta en la sección 2.3.2, presenta una implementación hardware viable para ser desplegado en los dispositivos calificados para espacio, siendo su principal limitación la tasa de datos alcanzada causada por la complejidad del esquema híbrido de codificación entrópica.

En [29] se presenta el algoritmo LCE (del inglés, *Lossy Compression for ExoMars*). El objetivo principal es el diseño de un algoritmo *ad hoc* para su despliegue a bordo de la misión ExoMars alcanzando altas ratios de compresión. Para ello emplea una etapa predictiva que, a su vez, aporta una etapa de optimización tasa de bits – distorsión. En el capítulo 3, se detalla de este algoritmo y se propone una implementación hardware para ser incluida a bordo de un satélite haciendo uso de dispositivos calificados para el espacio.

El control de la tasa de bits es considerado como un reto en los algoritmos de compresión predictivos [21]. Este hecho se fundamenta en la inexistencia de una relación matemática simple entre la tasa de bits y el residuo predicho cuantificado. En [81], se presenta un algoritmo de compresión de imágenes hiperespectrales con control de la tasa de bits bajo las premisas de baja complejidad y bajos requisitos de almacenamiento. El algoritmo, si bien puede ser adaptado para su desempeño con cualquiera de los predictores sin pérdidas presentados en la sección anterior, ha sido implementado en conjunto con el predictor incluido en el estándar de compresión CCSDS 123.0, de tal forma que puede considerarse una extensión con pérdidas y control de la tasa de bits de este estándar. Asimismo, se introduce la codificación de rango como alternativa a los codificadores entrópicos propuestos en el estándar. La principal desventaja de este algoritmo reside en su naturaleza serie, ya que presenta fuertes dependencias de datos en las etapas que lo componen. Además, el codificador de rango

propuesto incrementa la complejidad del algoritmo de compresión ya que requiere de cuatro modelos estadísticos independientes para su funcionamiento. Con el fin de abordar estas limitaciones, en [30] se presenta un nuevo algoritmo que disminuye la latencia de ejecución y simplifica el cálculo del paso de cuantificación. Para ello, clasifica los píxeles en función de su situación dentro de la imagen y emplea el cálculo de medianas como base de su proceso. De esta forma, se presenta como una opción comparativamente más atractiva para ser implementado a bordo de un satélite. En el capítulo 5, se presentan en detalle ambos algoritmos de compresión, además de la implementación hardware del algoritmo [30] demostrando su viabilidad para ser desplegado a bordo de un satélite.

2.3 Implementaciones de algoritmos de compresión de imágenes hiperespectrales a bordo de satélites

En la sección anterior se presentaron una serie de algoritmos de compresión de imágenes hiperespectrales junto con sus limitaciones a la hora de realizar el procesamiento a bordo de satélites. De forma similar, en esta sección se analizan las tecnologías y dispositivos cualificados para ser incluidos a bordo de satélites, y se evalúan sus características haciendo especial hincapié en todos aquellos aspectos relativos a la mitigación de los efectos de la radiación que hacen que sus prestaciones se vean reducidas.

2.3.1 Tecnologías disponibles en el sector espacial

El elemento diferencial que distingue el entorno espacial del resto de entornos es la presencia de la radiación. Principalmente, la radiación espacial se compone de protones, electrones e iones pesados. Los protones e iones pesados son originados por la actividad solar y por rayos cósmicos, conocidos como GCR (del inglés, *Galactic Cosmic Ray*), que son producidos generalmente por supernovas en el interior y exterior de nuestra galaxia. Por otra parte, los electrones son partículas que se encuentran en el interior de los cinturones de Van Allen [82]. La radiación espacial es en ocasiones ionizante, por lo que es capaz de originar la producción de carga en la materia con la que interactúa, causando modificaciones en la estructura interna del material.

La radiación espacial es la responsable de causar los dos efectos principales que los semiconductores sufren en el entorno espacial:

- **TID** (del inglés, *Total Ionizing Dose*). A lo largo de una misión, multitud de partículas ionizantes atraviesan el material de un semiconductor depositando una cantidad de carga tal que provoca un daño sobre este material afectando a su composición interna. Este daño se acumula en el tiempo sobre el material del semiconductor y es conocido como TID. El TID es medido en unidades de J/Kg o gray (Gy), de acuerdo con el

sistema internacional. Asimismo, se puede encontrar en la literatura expresado en unidades de rad, siendo la conversión 1 rad = 0,01 Gy.

- **SEE** (del inglés, *Single Event Effect*). Los SEEs ocurren cuando una partícula ionizante suficientemente cargada, atraviesa un material semiconductor depositando la energía suficiente como para causar error instantáneo en el dispositivo. Existen distintos tipos de SEE y su efecto puede ser permanente o temporal. Dentro de los efectos no destructivos, los denominados SEU (del inglés, *Single Event Upset*) y MBU (del inglés, *Multiple Bit Upset*), son los principales, ya que pueden llegar a causar cambios no intencionados del valor lógico de uno (SEU) o varios (MBU) bits dentro de un circuito integrado. Sin embargo, otro tipo de SEE resultan destructivos, interrumpiendo la funcionalidad del dispositivo, siendo los principales, por su nivel de ocurrencia, los denominados SEL, SET, SEB y SEGR. Los eventos de tipo SEL (del inglés, *Single Event Latchup*) ocurren cuando una partícula ionizante crea un camino de corriente no intencionado dentro del material semiconductor. Un SET (del inglés, *Single Event Transient*) resulta en una rotura temporal de la salida de un dispositivo, ya sea analógico o digital, causado por una partícula ionizante. Finalmente, SEB (del inglés, *Single Event Burnout*) y SEGR (del inglés, *Single Event Gate Rupture*) crean un paso de corriente en el interior de transistores, ya sean de tipo BJT, FET o MOSFET, interrumpiendo su funcionamiento nominal.

Las técnicas de mitigación de los efectos de la radiación ionizante dependen del tipo de efecto a mitigar. Así pues, los efectos acumulativos TID son mitigados incrementando el apantallamiento del equipo en el que se encuentra el dispositivo y aplicando técnicas conservativas de diseño. En el caso de las técnicas de mitigación de SEEs, pese a existir dispositivos inmunes por diseño, la mitigación de eventos SEL se realiza mediante el diseño de circuitos específicos de detección de manera externa al dispositivo a proteger.

Los eventos SEU o MBU, pueden ser mitigados mediante el uso de redundancia, en general mediante TMR (del inglés, *Triple Modular Redundancy*), en conjunto con mecanismos de votación, y de códigos de detección y corrección de errores ECC (del inglés, *Error Correcting Codes*), por ejemplo mediante el empleo de bits de paridad o códigos Hamming [42]. La evaluación del impacto de los eventos transitorios de tipo SET resulta más compleja dada la corta duración del fenómeno y a que su efecto se propaga externamente hacia el resto de dispositivos en el diseño. Por ello, la tendencia actual es el empleo en el entorno espacial de dispositivos que son inmunes por diseño ante este tipo de eventos [83].

Los dispositivos empleados en el entorno espacial han de estar cualificados para operar en este entorno y presentar estrategias de mitigación frente a los eventos expuestos con el fin de operar ininterrumpidamente a lo largo de la misión para la que son diseñados. Sin embargo, estas estrategias de mitigación reducen las prestaciones de los dispositivos calificados para el espacio si son comparados con dispositivos comerciales.

Los principales dispositivos disponibles para operar en el entorno espacial que son empleados en los sistemas de compresión de imágenes hiperespectrales se presentan en las siguientes secciones.

2.3.1.1 Procesadores de propósito general

Los algoritmos de compresión de imágenes hiperespectrales, en su primer estadio, se implementan a nivel software, en lenguajes como C/C++, sobre procesadores de propósito general, GPP (del inglés, *General Purpose Processor*). Por lo tanto, resulta interesante la evaluación de este tipo de dispositivos para la implementación a bordo de los algoritmos de compresión. El desarrollo de algoritmos centrados en su implementación sobre GPPs de propósito general presentan una gran flexibilidad y tiempos cortos de desarrollo gracias al potencial tanto de las herramientas a emplear como del nivel de abstracción del lenguaje. Sin embargo, resultan ineficientes si se comparan con dispositivos de tipo DSP o FPGA, para la ejecución de algoritmos de compresión de imágenes hiperespectrales ya que estos presentan

características que favorecen su implementación paralela. Dentro del sector institucional europeo, la familia de procesadores basados en el LEON y sus evoluciones, dominan el mercado, siendo desplegados en la mayor parte de misiones. Este es el caso del LEON2-FT, LEON3-FT y actualmente el LEON4-FT [11]. Recientemente, destaca la iniciativa TCLS ARM (del inglés, *Triple-Core Lock Step*) en el marco del Horizonte 2020, con la intención de ser la primera implementación de un procesador basado en la tecnología ARM, en concreto el Cortex R-5, calificado para espacio [84].

2.3.1.2 Procesador Digital de Señal

Los dispositivos de procesado digital de señal, en adelante DSP (del inglés, *Digital Signal Processor*), presentan una alta tasa de datos combinada con un alto número de operaciones en punto fijo y punto flotante. Actualmente, la tendencia se sitúa en arquitecturas en donde existen varios núcleos de procesamiento paralelo proporcionando altas prestaciones en términos de capacidad de procesado. El dispositivo DSP 21020 del fabricante Analog Devices, empleado tradicionalmente en las misiones espaciales ESA, ha sido declarado obsoleto dado que las GPPs disponibles para operar en el espacio, como el LEON3-FT, mejoran sus prestaciones en cuanto a capacidad de cómputo. En este contexto, nace en la ESA la iniciativa para el desarrollo de un DSP europeo, denominado SSDP (del inglés, *Scalable Sensor Data Processor*) [11]. El SSDP consiste en un ASIC mixto analógico y digital tolerante a radiación, con el objetivo de ser empleado para el procesado de datos a bordo. Estos dispositivos representan una alternativa a las GPPs para la implementación de algoritmos de compresión de imágenes hiperespectrales gracias a su capacidad de procesado paralelo. Sin embargo, los dispositivos DSPs presentan menores tasas de bits si son comparados con dispositivos de tipo ASIC o FPGA.

2.3.1.3 Circuitos integrados de aplicación específica

Los ASICs (del inglés, *Application-Specific Integrated Circuit*) o circuitos integrados de aplicación específica, son empleados en aplicaciones con necesidades de procesamiento difícilmente alcanzable por GPPs o bien mediante DSPs. Los dispositivos tipo ASIC se presentan como alternativas viables para la compresión de imágenes hiperespectrales. Tradicionalmente, los algoritmos de compresión de imágenes han sido implementados en este tipo de dispositivos ya que resultan eficientes en cuanto a consumo de potencia y tasa de bits. Su mayor inconveniente reside en la imposibilidad de adaptarse a los cambios de un algoritmo de compresión una vez ha sido fabricado el dispositivo y al elevado coste económico de los mismos.

2.3.1.4 Dispositivos de lógica programable

Los dispositivos de lógica programable de tipo FPGA (del inglés, *Field Programmable Gate Arrays*) presentan mayor flexibilidad que los ASICs en la implementación de algoritmos de compresión. Además, son capaces de realizar un procesado paralelo de datos, incrementando de esta forma la tasa de datos de la implementación de un algoritmo. La Figura 2.1 muestra el árbol de tecnologías, fabricantes y familias de producto de las principales FPGAs calificadas para el espacio disponibles en el mercado. Se observa que, el mercado de dispositivos FPGA se encuentra compuesto principalmente por fabricantes europeos y americanos.

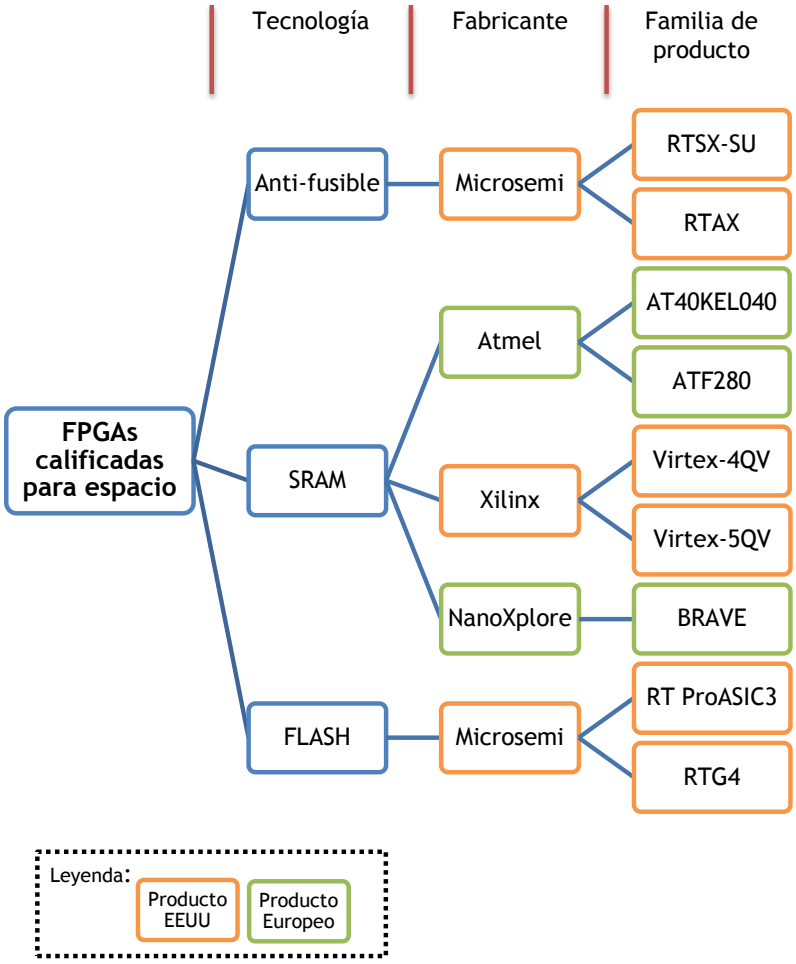


Figura 2.1 Principales dispositivos de tipo FPGA calificados para espacio

Las FPGAs basadas en tecnología anti-fusible han sido ampliamente usadas en misiones tanto ESA como NASA ya que implementan estrategias de mitigación frente a SEU/SEL y soportan altas dosis de TID [85]. Por el contrario, son dispositivos OTP (del inglés, *One Time Programmable*), por lo que una vez programados no es posible su modificación, perdiendo de esta manera la flexibilidad en fases avanzadas del diseño. Asimismo, los recursos disponibles en términos de área en este tipo de FPGAs son menores que los que presentan los dispositivos basados en tecnología SRAM o FLASH, incluso en el caso de los nuevos dispositivos RTAX4000S o RTAX4000S-DSP.

Los dispositivos FPGA basados en tecnología SRAM, al margen de ser reprogramables, presentan grandes capacidades en términos de área y de recursos dedicados. Por ejemplo, contienen bloques DSP dedicados o PLL (del

inglés, *Phase-Locked Loop*). Estas ventajas permiten la implementación de algoritmos de compresión complejos a bordo de satélites. La principal desventaja de este tipo de FPGAs reside en su mayor consumo de potencia y la necesidad de una memoria de configuración externa a la propia FPGA para su programación inicial. En el mercado europeo, los dispositivos de tipo FPGA empleados del fabricante Atmel presentan recursos limitados en términos de área resultando inviable la implementación de algoritmos de compresión de imágenes hiperespectrales. Actualmente, existe una iniciativa prometedora en el sector europeo para el desarrollo de una familia de FPGAs basada en tecnología SRAM denominada BRAVE. Esta iniciativa, liderada por NanoXplore, emplea un proceso de fabricación sobre tecnología tolerante a radiación de 65 nm del fabricante ST Microelectronics [86]. En el mercado americano, el fabricante Xilinx lidera la fabricación de dispositivos FPGA de tecnología SRAM. La familia de FPGAs Virtex-4QV no son tolerantes a radiación por diseño por lo que es necesario que el usuario emplee técnicas de mitigación en su diseño, complicando su proceso de desarrollo [87]. Por el contrario, la familia de FPGAs Virtex-5QV son tolerantes a la radiación, presentando una inmunidad comparable a las FPGA basadas en tecnología anti-fusible [88]. Este tipo de FPGAs han sido empleadas con éxito para la implementación a bordo de algoritmos de compresión de imágenes hiperespectrales [28].

Los dispositivos FPGA basados en tecnología FLASH comparten las mismas ventajas comentadas sobre los dispositivos basados en tecnología SRAM en términos de re-programación, área disponible y recursos dedicados. La principal diferencia reside en que los dispositivos basados en tecnología FLASH no necesitan memoria externa de configuración para su programación ya que ésta se almacena en el interior del dispositivo. La FPGA RTProASIC3 del fabricante Microsemi es la primera FPGA disponible para ser desplegada a bordo de un satélite [85]. Si bien no es tolerante a radiación, es posible implementar por parte del usuario técnicas de tipo TMR para mitigar los efectos de la radiación. Esta FPGA presenta pocos recursos disponibles en términos de área para poder implementar algoritmos con alta carga computacional como lo son los algoritmos de compresión de imágenes hiperespectrales. La FPGA RTG4, es tolerante a radiación, presenta una gran

área de recursos lógicos disponibles y, además, contiene multiplicadores dedicados para algoritmos de alta demanda computacional, sin embargo, carece de experiencia en vuelo por lo que no existen algoritmos desarrollados para este dispositivo que proporcionen métricas representativas [85].

2.3.2 Implementaciones disponibles

Las principales aportaciones en términos de implementación a bordo de algoritmos de compresión de imágenes hiperespectrales se presentan en esta sección realizando especial hincapié en los dispositivos sobre los que se han desplegados estas aportaciones. Se han excluido de este análisis las aportaciones existentes en la literatura de algoritmos de compresión de imágenes hiperespectrales implementados sobre dispositivos no cualificados para operar en el espacio. En este sentido, la implementación del algoritmo ICER-3D resulta un caso particular ya que pese a ser implementado sobre la FPGA no cualificada para espacio XC2VCP70 de la familia Virtex-2 Pro del fabricante Xilinx, ha sido empleado en la misión *Mars Exploration Rovers* [63]. Los resultados proporcionan una ocupación del área en el dispositivo del 60 % de los recursos y una tasa de datos del 4,5 Mmuestras/s, resultados mejorados por el resto de implementaciones abordadas en el resto de la sección.

Airbus Defence and Space ha desarrollado sobre un ASIC el compresor WICOM (del inglés, *Wavelet Image COMpression*) [11]. Este compresor implementa el algoritmo basado en la transformada DWT denominado MRCPB (del francés, *Multi-Résolution par Codage de Plans Binaires*) para la compresión de imágenes en 2D. El dispositivo alcanza una tasa de datos de 20 Mmuestras/s. La información detallada de este algoritmo no se encuentra disponible ya que tanto el ASIC como el propio algoritmo están patentados, limitando así su posible evolución hacia la compresión de imágenes hiperespectrales. El ASIC WICOM se ha empleado en misiones como PLEIADES, SENTINEL-2 o SEOSAT para la compresión de imágenes en 2D generadas por los sensores a bordo de estos satélites.

El ASIC CWICOM (del inglés, *CCSDS Wavelet Image COMpression*) implementa el estándar de compresión de imágenes 2D basado en

transformada CCSDS 122.0. Este ASIC permite la compresión sin pérdidas y con pérdidas de imágenes en 2D con una tasa de datos de hasta 60 Mmuestras/s. El dispositivo no requiere del empleo de almacenamiento externo ya que internamente contiene 5 Mbit de memoria. El ASIC CWICOM es parte fundamental del sistema de compresión empleado en el instrumento JANUS dentro de la misión JUICE cuyo lanzamiento se encuentra planificado para el año 2022 [89]. El ASIC CWICOM es un dispositivo interesante para ser empleado en la compresión 2D de imágenes a bordo gracias a sus características internas. Como se presenta en el capítulo 4, este dispositivo, junto con una etapa previa de eliminación de la correlación espectral, puede ser empleado a bordo como sistema de compresión de imágenes hiperespectrales.

La misión Proba-V tiene como objetivo el estudio de la evolución natural de la vegetación sobre la superficie terrestre mediante el empleo de imágenes multiespectrales. Como cualquier misión que emplee instrumentos de observación multiespectral, presenta limitaciones tanto en la capacidad de almacenamiento a bordo como en el canal de comunicaciones hacia la estación terrena. La solución adoptada ha sido la implementación del estándar de compresión CCSDS 122.0 sobre un dispositivo de tipo FPGA. La implementación se ha realizado sobre la FPGA de tecnología anti-fusible RTAX2000S del fabricante Microsemi. La transformada DWT se aplica en tres niveles en los que se realiza la descomposición en 10 sub-bandas de la imagen en 2D usando una aproximación entera no lineal del filtro 9/7. La FPGA no dispone de memoria interna suficiente para el almacenamiento de los resultados parciales de la DWT y para el almacenamiento de los coeficientes, por lo que resulta necesario que la implementación emplee memoria RAM externa. De manera similar, este estándar ha sido particularizado e implementado en una RTAX2000S en la misión gubernamental alemana EnMap. La Tabla 2.1 presenta los resultados de la implementación sobre la FPGA RTAX2000S del algoritmo CCSDS 122.0 dentro de la misión Proba-V y la misión EnMap [54].

	Registros	Comb.	Bloq. RAM	Frecuencia (MHz)
CCSDS 122.0 (Proba-V)	7125 (66%)	8455 (39%)	54 (84%)	64
CCSDS 122.0 (EnMap)	7620 (71%)	9772 (45%)	58 (91%)	50

Tabla 2.1 Resultados de implementación sobre la FPGA RTAX2000S del algoritmo CCSDS 122.0 en la misión Proba-V

El algoritmo FL, base del estándar de compresión sin pérdidas CCSDS 123.0, ha sido implementado en el seno del JPL (del inglés, *Jet Propulsion Laboratory*) sobre la FPGA de la familia Virtex-5 SX50T [26]. Esta FPGA comercial tiene su equivalente calificado para espacio en el dispositivo XQR5VFX130 del fabricante Xilinx, por lo tanto, los resultados de ocupación resultan representativos en términos de ocupación y tasa de datos. La tasa de datos alcanzada es de 41 Mmuestras/s mientras que, los resultados de ocupación, se sitúan en el 38 % de los recursos disponibles, tal y como se presenta en la Tabla 2.2.

	LUTs	CLBs	DFF o Latches	Bloq. RAM	Bloq. DSP	Frecuencia (MHz)
FL	12697 (38%)	898 (6%)	1586 (4%)	8 (6%)	3 (1%)	-

Tabla 2.2 Resultados de implementación sobre la FPGA Virtex-5 SX50T del algoritmo FL

En [27] se presenta una implementación del estándar de compresión sin pérdidas de imágenes hiperespectrales CCSDS 123.0 denominado HyLoC, demostrando la viabilidad de este estándar para ser implementado en dispositivos de tipo FPGA a bordo de satélites. La implementación se realiza sobre la FPGA basada en tecnología anti-fusible RTAX2000S del fabricante Microsemi y sobre el dispositivo XQR5VFX130 basado en tecnología SRAM del fabricante Xilinx. La implementación alcanza tasas de datos de 3,5 Mmuestras/s para la FPGA RTAX2000S y de 11,3 Mmuestras/s sobre la FPGA

XQR5VFX130. La Tabla 2.3 muestra los resultados obtenidos de su implementación sobre la FPGA RTAX2000S, mientras que, en la Tabla 2.4 se muestran los resultados de implementación sobre la FPGA XQR5VFX130.

	Registros	Comb.	Bloq. RAM	Frecuencia (MHz)
HyLoC	1438 (18%)	4282 (13%)	0 (0%)	41

Tabla 2.3 Resultados de implementación sobre la FPGA RTAX2000S del algoritmo HyLoC

	LUTs	CLBs	DFF o Latches	Bloq. RAM	Bloq. DSP	Frecuencia (MHz)
HyLoC	2342 (2%)	842 (4%)	1535 (1%)	0 (0%)	1 (0%)	134

Tabla 2.4 Resultados de implementación sobre la FPGA XQR5VFX130 del algoritmo HyLoC

En el campo de los algoritmos de baja complejidad para la compresión con pérdidas de imágenes hiperespectrales basados en el esquema DPCM, en [28] se presentan los resultados de implementación sobre la FPGA XQR5VFX130 del fabricante Xilinx del algoritmo FLEX. Los resultados muestran una tasa de datos de 3,4 Mmuestras/s y una ocupación máxima del 29 % de los recursos disponibles. La Tabla 2.5 presenta los resultados de implementación del algoritmo FLEX sobre la FPGA calificada para espacio XQR5VFX130 de la familia Virtex-5QV.

	LUTs	CLBs	DFF o Latches	Bloq. RAM	Bloq. DSP	Frecuencia (MHz)
FLEX	16185 (20%)	5911 (29%)	18648 (23%)	27 (9%)	40 (13%)	82,5

Tabla 2.5 Resultados de implementación sobre la FPGA XQR5VFX130 del algoritmo FLEX

En [90] se presenta la implementación del algoritmo de compresión con pérdidas y control de la tasa de bits de imágenes hiperespectrales basado en el predictor CCSDS 123.0 presentado en [81]. La implementación de este algoritmo ha sido denominada Hydra. Los resultados han sido obtenidos para la FPGA XQR5VFX130 de la familia Virtex-5QV del fabricante Xilinx. Esta implementación proporciona una tasa de datos de 20 Mmuestras/s. Los resultados obtenidos en la implementación del algoritmo sobre la FPGA XQR5VFX130 se muestran en la Tabla 2.6.

	LUTs	CLBs	DFF o Latches	Bloq. RAM	Bloq. DSP	Frecuencia (MHz)
Hydra	19957 (24%)	8496 (42%)	4296 (5%)	158 (53%)	71 (22%)	-

Tabla 2.6 Resultados de implementación sobre la FPGA XQR5VFX130 del algoritmo Hydra

2.4 Conclusiones

En este capítulo se ha presentado un estudio exhaustivo del estado del arte de algoritmos e implementaciones destinadas a la compresión de imágenes hiperespectrales para aplicaciones espaciales. La discusión se ha centrado en aquellos aspectos de especial relevancia teniendo en cuenta las particularidades que presenta el entorno espacial.

Se han presentado los principales algoritmos y estándares de compresión tanto sin pérdidas como con pérdidas, detallando su clasificación entre algoritmos basados en transformadas, algoritmos predictivos o algoritmos basados en cuantificación vectorial. De manera similar, se han identificado aquellos algoritmos diseñados *ad hoc* para una misión en concreto, aquellos que se encuentran basados en estándares multimedia y los algoritmos diseñados en el seno del sector espacial.

Las implementaciones disponibles de los algoritmos de compresión se han detallado a continuación, identificando en primer lugar las tecnologías disponibles a bordo de un satélite en los sistemas de compresión. En este sentido, los dispositivos de tipo FPGA destacan por su flexibilidad en el desarrollo del algoritmo y su posibilidad de procesado paralelo. El estudio de las implementaciones disponibles pone de manifiesto que existe un amplio abanico de algoritmos de compresión sin pérdidas de imágenes hiperespectrales disponibles para ser desplegados a bordo. Por el contrario, existe en la literatura un menor número de implementaciones de algoritmos de compresión con pérdidas disponibles. Este hecho refuerza la motivación del trabajo de investigación presentado en esta Tesis Doctoral, centrado en el desarrollo de implementaciones que confirmen la viabilidad de este tipo de algoritmos para ser desplegados a bordo de satélites en futuras misiones espaciales.

Capítulo 3

Implementación del algoritmo de compresión con pérdidas para la misión ExoMars (LCE)

El algoritmo de compresión con pérdidas LCE (del inglés, Lossy Compression for ExoMars) ha sido especialmente diseñado para ser desplegado en la misión ExoMars llevada a cabo por la Agencia Espacial Europea. En el diseño del algoritmo se han empleado estrategias para facilitar su implementación en dispositivos cualificados para operar en el entorno espacial.

A lo largo de este capítulo se aborda la implementación del algoritmo LCE sobre diferentes dispositivos FPGAs con el fin de demostrar su viabilidad en términos de uso de recursos en cada dispositivo siguiendo una metodología de síntesis de alto nivel, HLS (del inglés, High-Level Synthesis). Asimismo, se presenta una estrategia modular basada en la metodología HLS que proporciona mejores resultados que los obtenidos en otras implementaciones.

3.1 Introducción

El algoritmo de compresión con pérdidas para la misión ExoMars, en adelante LCE (del inglés, *Lossy Compression for ExoMars*), ha sido diseñado *ad hoc* para la misión ExoMars [29]. El LCE es un algoritmo con pérdidas que puede ser empleado como algoritmo casi-sin pérdidas gracias a que ofrece la posibilidad de limitar la distorsión máxima en la imagen reconstruida.

El algoritmo LCE presenta una complejidad computacional baja, lo que permite la compresión de imágenes hiperespectrales en tiempo real. Su funcionamiento se fundamenta en la compresión de bloques independientes de la imagen, hecho que transfiere al algoritmo la capacidad de tolerar errores a nivel de bit en los datos comprimidos causados por la radiación presente en el entorno espacial. Si se produce un error, éste afecta únicamente al bloque en el que se ha producido el error, permitiendo descomprimir correctamente el resto de bloques que componen la imagen. Por otra parte, los resultados al comprimir diversas imágenes con el algoritmo LCE muestran mejores resultados en cuanto a la ratio tasa de bits – distorsión de la imagen reconstruida para tasas de bits entre 1 bpp y 3 bpp si se compara con los resultados proporcionados por el algoritmo JPEG 2000 [29].

En este capítulo se presenta la implementación sobre distintos dispositivos FPGA del algoritmo LCE descrito en lenguaje ANSI C. En la implementación se ha empleado la herramienta de síntesis de alto nivel CatapultC para la generación de la descripción VHDL a nivel RTL (del inglés, *Register Transfer Level*) del algoritmo. Se ha escogido esta herramienta ya que permite la generación de código RTL desde descripciones en lenguaje ANSI C. La implementación se ha realizado sobre el dispositivo XQR5VFX130 de la familia Virtex-5QV del fabricante Xilinx y sobre la FPGA RTAX2000S del fabricante Microsemi. Los dispositivos XQR5VFX130 y RTAX2000S se encuentran cualificados para operar en el entorno espacial, por lo tanto, se han analizado con especial atención los resultados obtenidos sobre estos dispositivos en cuanto a ocupación de recursos, frecuencia máxima de operación y tasa de datos.

3.2 Algoritmo de compresión con pérdidas LCE

El algoritmo de compresión de imágenes hiperespectrales LCE ha sido desarrollado en el seno del Departamento de Electrónica y Telecomunicaciones de la Universidad Politécnica de Turín [29]. El objetivo principal en el diseño del algoritmo es su despliegue a bordo de la misión ExoMars alcanzando altas ratios de compresión. Para satisfacer tal demanda, el algoritmo ha de presentar las siguientes características: baja complejidad computacional, tolerancia a errores y, además, poseer una implementación viable en los dispositivos disponibles para operar a bordo de un satélite.

El algoritmo se compone de dos etapas funcionales: el predictor y el codificador entrópico Golomb potencia de 2. Ambas etapas funcionales se presentan en la Figura 3.1.

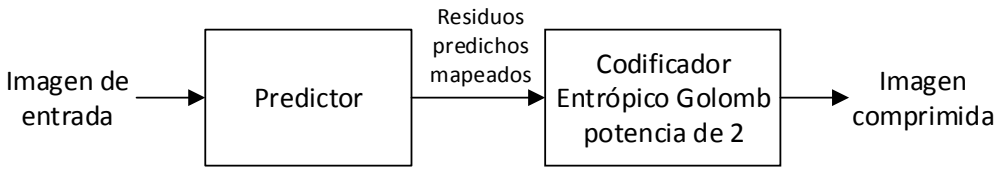


Figura 3.1 Estructura del algoritmo de compresión LCE

Con el fin de ofrecer robustez frente a errores, el algoritmo LCE codifica de manera independiente bloques de imagen de 16 x 16 píxeles. De esta forma, cada bloque es individualmente comprimido con lo que un error en un bloque dado no se propaga al resto de bloques. Asimismo, ha sido diseñado para que su implementación hardware explote el paralelismo entre el procesado de distintos bloques permitiendo mayores prestaciones en sensores con altas tasas de datos.

3.2.1 Predictor

El algoritmo LCE comprime bloques independientes no solapados de tamaño 16 x 16 píxeles para todas las bandas del cubo hiperespectral. Sea $x_{m,n,i}$ el píxel perteneciente a una imagen hiperespectral en la fila m, columna

n y banda i. La Figura 3.2, muestra la ubicación del píxel a codificar y de la vecindad que lo rodea.

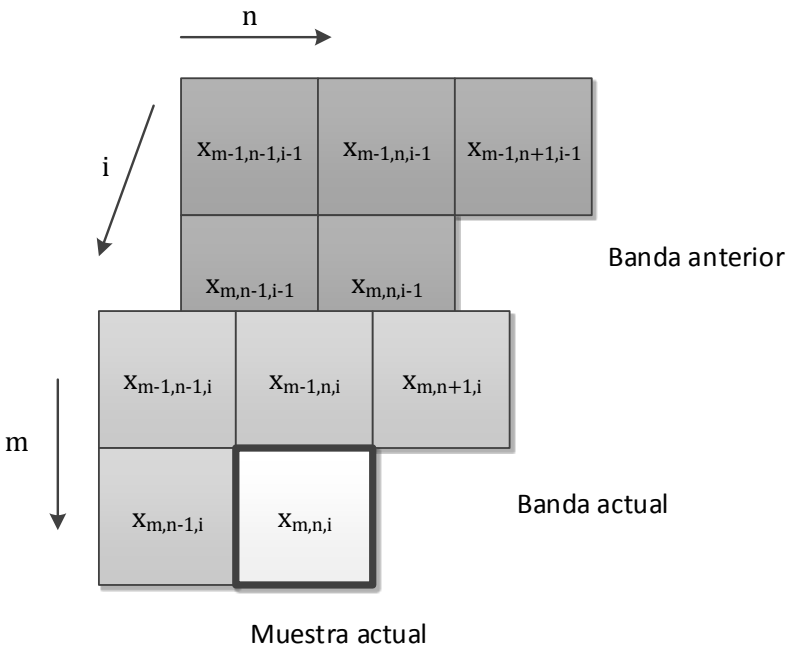


Figura 3.2 Localización de la muestra actual y su vecindad

En la primera banda ($i = 0$), se aplica una compresión denominada 2D (modo INTRA) en la que no se emplea información de ninguna otra banda. Las muestras son procesadas de izquierda a derecha y de arriba hacia abajo (*raster scan order*). En general, el predictor empleado para esta banda es el presentado en la expresión (3.1).

$$\tilde{x}_{m,n,0} = \left(\hat{x}_{m-1,n,0} + \hat{x}_{m.n-1,0}\right) \gg 1 \qquad (3.1)$$

Donde $\tilde{x}$ representa al predictor, $\hat{x}$ el valor del píxel reconstruido y $\gg$ representa el desplazamiento de un bit hacia la derecha. Por lo tanto, el predictor para la primera banda se basa en la media de los píxeles vecinos (encima e izquierda) del pixel a predecir. Nótese que el primer píxel no es predicho, los píxeles en la primera fila, es decir, $m = 0$, son predichos mediante la siguiente expresión $\tilde{x}_{0,n,0} = \hat{x}_{0,n-1,0}$ y los píxeles en la primera columna, $n = 0$, se predicen como $\tilde{x}_{m,0,0} = \hat{x}_{m-1,0,0}$.

Para el resto de bandas, $i \neq 0$, las muestras $x_{m,n,i}$ se predicen mediante las muestras decodificadas, $\hat{x}_{m,n,i-1}$, es decir, mediante las muestras del mismo bloque en la banda anterior. Se computa sobre el bloque un estimador LMS (del inglés, *Least Mean Square*) definido como $\propto = \frac{\propto_N}{\propto_D}$, siendo $\propto_N$ y $\propto_D$:

$$\propto_N = \sum_{m,n} (\hat{x}_{m,n,i-1} - m_{i-1})(x_{m,n,i} - m_i) \tag{3.2}$$

$$\propto_D = \sum_{m,n} (\hat{x}_{m,n,i-1} - m_{i-1})(x_{m,n,i-1} - m_{i-1}) \tag{3.3}$$

Donde, m_i y m_{i-1} son las medias, dentro del conjunto de números enteros, computadas de los bloques anteriores decodificados en las bandas i e $i-1$.

$$m_{i/i-1} = round\left(\frac{1}{P}\sum_{m,n} x_{m,n,i/i-1}\right) \tag{3.4}$$

En la ecuación anterior P se define como el número de muestras computadas en el bloque actual.

Las versiones cuantificadas de $\propto$ y m_i, denominadas $\hat{\propto}$ y $\hat{m}_i$, se generan mediante el empleo de un cuantificador escalar. Finalmente, los valores de las muestras predichas se calculan para todas las muestras pertenecientes a un bloque como:

$$\tilde{x}_{m,n,i} = \hat{m}_i + \hat{\propto} \left(\hat{x}_{m,n,i-1} - m_{i-1}\right) \tag{3.5}$$

Por consiguiente, el error de predicción se calcula mediante la siguiente expresión:

$$e_{m,n,i} = x_{m,n,i} - \tilde{x}_{m,n,i} \tag{3.6}$$

3.2.1.1 Optimización tasa de bits-distorsión

Antes de proceder a la cuantificación del error de las muestras predichas, se verifica si el valor de la predicción se aproxima al valor real del pixel, permitiendo de esta forma evitar la codificación de las muestras de

error. Si este es el caso, se fija al valor '1' lógico un *flag* que indica que el bloque actual de muestras de errores de predicción es cero. Esta condición se denomina, *zero_block condition*. Para la toma de esta decisión, se computa la energía del error de predicción mediante la siguiente expresión:

$$D_0 = \frac{1}{256}\sum_{m,n} e_{m,n,i}^2 \tag{3.7}$$

Si D_0 es menor que el umbral definido por el usuario D_T, es decir, $D_0 < D_T$, se aplica la condición *zero_block*.

3.2.1.2 Cuantificación y mapeo

Las muestras de los errores de predicción son cuantificadas en valores enteros, $eq_{m,n,i}$, para, posteriormente, ser decuantificadas a valores reconstruidos, $er_{m,n,i}$. Dentro de la primera banda, $i = 0$, este proceso se realiza pixel a pixel empleando un cuantificador escalar uniforme. Para el resto de bandas, $i \neq 0$, es posible escoger entre el cuantificador escalar uniforme y el cuantificador de umbral uniforme, UTQ (del inglés, *Uniform-Threshold Quantizer*) presentado en [91]. Finalmente, los valores reconstruidos son mapeados a números enteros no negativos empleando la siguiente aproximación.

$$s_{m,n,i} = \begin{cases} 2\left|eq_{m,n,i}\right| - 1, \text{si } eq_{m,n,i} > 0 \\ 2\left|eq_{m,n,i}\right| \quad\quad , \text{si } eq_{m,n,i} \leq 0 \end{cases} \tag{3.8}$$

3.2.2 Codificador entrópico

El bloque de 16 x 16 residuos se codifica (de izquierda a derecha y de arriba hacia abajo) empleando la codificación Golomb potencia de 2, exceptuando la primera muestra de cada bloque. Ésta, se codifica empleando un código Golomb exponencial potencia de 0. La primera muestra cuantificada de la primera banda no se codifica, sino que, se almacena en la imagen comprimida usando 16 bits.

El fichero comprimido a la salida del codificador entrópico se compone mediante la concatenación de bloques codificados. Éstos han de ser leídos de izquierda a derecha y de arriba hacia abajo en la dimensión espacial. Cada bloque es codificado para todas las bandas. En cuanto a la información de salida escrita en el archivo comprimido para todos los bloques se tiene:

a) los parámetros $\propto$ y m, excepto para la primera banda;
b) un bit correspondiente al indicador *zero_block condition*;
c) para aquellos bloques en los que no se cumpla la condición *zero_block*, las muestras de error de predicción cuantificadas de cada bloque escritas en *raster scan order*.

La Figura 3.3 muestra el formato del fichero que contiene la imagen comprimido que proporciona a su salida el algoritmo de compresión LCE.

Figura 3.3 Formato del fichero comprimido generado por el algoritmo LCE

3.3 Implementación del algoritmo LCE

En esta sección se detallan los trabajos realizados con el objeto de obtener una implementación sintetizable del algoritmo de compresión LCE. Se aborda, en primer lugar, el software de referencia del algoritmo LCE describiendo los módulos que lo componen, flujos de datos y consideraciones a la hora de separar en módulos independientes el código de referencia. Posteriormente, se presentan los trabajos realizados en la herramienta de síntesis de alto nivel CatapultC para la obtención del código VHDL a nivel RTL sintetizable. Finalmente, se presenta el módulo de control hardware diseñado en VHDL, implementado y verificado que permite la conexión de los módulos independientes del algoritmo.

3.3.1 Implementación del algoritmo LCE en el código de referencia

El código original tomado como referencia ha sido desarrollado en el seno de Universidad Politécnica de Turín. Este código ha sido proporcionado gracias al acuerdo de colaboración entre esta institución y el Instituto Universitario de Microelectrónica Aplicada (IUMA). La implementación original del algoritmo se encuentra en lenguaje ANSI C operando sobre cada bloque independiente de 16 x 16 píxeles de la imagen hiperespectral de entrada de forma secuencial, es decir, bloque a bloque [29]. Cada bloque se procesa conjuntamente con todas sus bandas espectrales, antes de que el siguiente bloque sea procesado.

La imagen hiperespectral de entrada es almacenada en una memoria temporal de datos de tipo entero. Durante el procesamiento de cada bloque, sus muestras son almacenadas localmente en una memoria de menor tamaño, en concreto de 256 posiciones (16 x 16 píxeles) de 16 bits, esta memoria local se denomina *curr_block*. El bloque usado como referencia a la hora de realizar la predicción se copia de igual forma una memoria local denominada *ref_block* (256 x 16 bits). Las palabras código de salida del algoritmo, se almacenan en una memoria de salida denominada *block_out* (256 x 32 bits).

Asimismo, se incluyen tres variables que proporcionan información relevante a la hora realizar las escrituras en la memoria de salida. La variable *pp* que almacena temporalmente los datos codificados y comprimidos. Cada vez que una nueva palabra de código es generada se almacena, temporalmente, en esta variable hasta que sus 32 bits sean escritos. Una vez se llene esta variable por la escritura de sus 32 bits, la palabra resultante se escribe en la memoria de salida *block_out*. El número real de bits escritos en la variable *pp* es controlado por la variable denominada *m*. Por otra parte, la variable *filecount* indica el número de palabras de 32 bits escritas en la memoria de salida. Durante la codificación de cada banda este valor es actualizado, fijándose nuevamente a su valor inicial de '0' lógico al inicio de cada banda.

El código de referencia del algoritmo LCE ha sido modificado en [24] aportando mejoras arquitecturales que facilitan su implementación a nivel hardware mediante la herramienta de síntesis de alto nivel CatapultC. Como paso inicial en [24] se extrae del código de referencia la parte del algoritmo que se desea implementar en hardware, quedando el resto como parte del banco de pruebas. Una vez realizado esta extracción, se introducen tipos de datos con precisión a nivel de bit. Para ello, se emplean datos de tipo *Algorithmic C*. De esta forma, la función denominada *pred1block()* contiene los módulos que posteriormente serán implementados mediante CatapultC. La Figura 3.4, muestra en pseudocódigo la estructura de la función *pred1block()*.

```
1: pred1block()
2:     if (i == 0)
3:         2D-prediction(INTRA-mode);
4:         entropy_coding();
5:     else
6:         spectral_prediction();
7:         RD-Optimization();
8:         quantization();
9:         entropy_coding();
10:    end if
11: end pred1block
```

Figura 3.4 Pseudocódigo de la función pred1block()

Dentro de las modificaciones realizadas en [24] se reducen el número de operaciones matemáticas realizadas mediante el empleo de operadores lógicos tales como, el desplazador lógico o funciones *and*/*or*, sin alterar el comportamiento del algoritmo. La división entera propuesta para el cálculo del parámetro de ganancia del predictor $\propto$, ha sido sustituida por una búsqueda dicotómica. En la misma línea, se minimizaron las operaciones de lectura/escritura en las memorias de entrada y de salida. Además, se optimizaron, con vistas a su implementación hardware, los bucles empleados por el algoritmo. Es importante remarcar que, como parte de los trabajos realizados en [24], se evaluó exhaustivamente el impacto de las modificaciones llevadas a cabo sobre el código original sin hallarse diferencias significativas entre el código inicial de referencia y el código a la salida.

En este trabajo, se ha empleado el código de referencia modificado en [24] para facilitar la implementación hardware del algoritmo LCE mediante la herramienta CatapultC.

3.3.2 Separación del código de referencia en módulos funcionalmente independientes

Como primer paso a la hora de abordar la implementación del algoritmo LCE a partir del código de referencia, se ha decidido dividir el mismo en módulos funcionalmente independientes. Para ello, se han identificado en el código de referencia aquellas funciones o sentencias de código asociadas a cada a cada una de las etapas del algoritmo LCE. Como salidas de esta tarea se dispone de dos ficheros para cada módulo independiente, un fichero con extensión *.cpp* que contiene el cuerpo de cada función y un fichero con extensión *.h* que contiene las cabeceras de las funciones empleadas.

El objetivo de esta primera etapa es el de obtener módulos funcionalmente independientes que sirvan de entrada a la herramienta CatapultC. La idea principal que subyace bajo esta aproximación es la de conocer dentro de cada módulo su ruta crítica para así poder conocer los módulos con que presentan una mayor carga computacional.

Los módulos funcionalmente independientes que componen el código final del algoritmo sobre el que se realiza la implementación hardware se describen a continuación:

- ***adgolomb.cpp* (*adgolomb.h*)**: En el interior de este módulo se incluyen las funciones: cuantificación del error de predicción, cómputo del bloque de referencia (valores reconstruidos, disponibles para la siguiente banda), implementación de la codificación entrópica del error de predicción y empaquetado a nivel de bit de las palabras código resultantes de la compresión.
- ***calc_perr.cpp* (*calc_perr.h*):** En este módulo, se computan tanto el predictor como el error de predicción teniendo en cuenta el valor del parámetro $\propto$ decuantificado. Asimismo, se retorna el estimador LMS ($D0$) con lo que a la salida de este módulo puede decidirse si se emplea la condición *zero_block*.
- ***estimationls.cpp* (*estimationls.h*):** El cálculo de los parámetros $\propto_N$ y $\propto_D$, además del propio $\propto$, se incluyen en este módulo. La resta del valor medio del bloque actual y el bloque de referencia también ha sido incluida en el módulo. Por otra parte, el cómputo del valor cuantificado de $\propto$ mediante la búsqueda dicotómica se añade como parte de este módulo. Finalmente, antes de retornar de la función, se calcula el valor decuantificado del parámetro $\propto$.
- ***init_output.cpp* (*init_output.h*):** Módulo auxiliar encargado de la inicialización a 0xFFFFFFFF de las primeras 128 posiciones de la memoria de palabras código de salida del algoritmo.
- ***mean.cpp* (*mean.h*):** El valor medio de las muestras pertenecientes a una banda se computa en este módulo. La función, retorna un valor entero de 22 bits.
- ***pred_2D.cpp* (*pred_2D.h*):** La predicción en dos dimensiones (2D), realizada para la primera banda, $i = 0$, se implementa en este módulo. Asimismo, la codificación entrópica y el cómputo del valor medio de las muestras del bloque actual para ser utilizadas en la siguiente banda son, igualmente, incluidos en el módulo.

- ***write_alphamu.cpp*** **(*write_alphamu.h*):** En este módulo se realiza la escritura en la memoria de palabras código de salida del valor cuantificado de $\propto$, la media del bloque actual y el indicador de la condición *zero_block*.
- ***zero_block.cpp*** **(*zero_block.h*):** Si se activa la condición *zero_block*, es decir, si el valor de la predicción se aproxima al valor real del pixel, se realiza la llamada a esta función, evitándose tanto la cuantificación, la codificación entrópica y el empaquetado de las palabras código.

Dentro del desarrollo de la tarea de separación en módulos funcionalmente independientes del código de referencia, se analizan cada una de las entradas/salidas resultantes de cada módulo con el fin de servir como entradas a las siguientes etapas del trabajo realizado. Resulta fundamental este análisis ya que las entradas o salidas que se definan, se convertirán posteriormente en puertos de entrada, de salida o bidireccionales en el lenguaje de descripción hardware VHDL.

En pseudocódigo, la función *pred1block()* presenta, una vez finalizada esta tarea, la implementación mostrada en la Figura 3.5. Se observa que, existen únicamente llamadas a funciones o decisiones a nivel de control, decidiéndose en ellas la llamada o no de cada función.

```
1: pred1block()
2:     if (i == 0)
3:         pred_2D();
4:     else
5:         mean();
6:         estimation_ls();
7:         calc_perr();
8:         write_alphamu ();
9:         if (k0 == 0);
10:            zero_block();
11:        else
12:            adgolomb();
13:        end if
14:    end if
15: end pred1block
```

Figura 3.5 Pseudocódigo de la función pred1block() tras la separación del código en módulos funcionalmente independientes

La Figura 3.6, presenta el flujo de datos y las dependencias entre los mismos resultado de la implementación de estas modificaciones en el código de referencia. Se observa que la primera banda, $i = 0$, es procesada de forma distinta al resto de bandas, tal y como se define a nivel teórico en el algoritmo LCE.

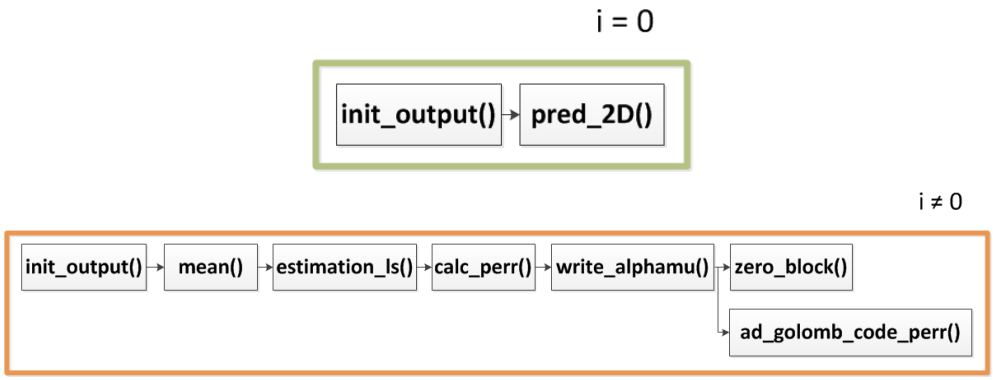


Figura 3.6 Flujo de datos y dependencias entre los distintos módulos que componen la implementación del algoritmo LCE

Cabe destacar que, durante la realización de esta tarea se ha verificado en todo momento que las modificaciones introducidas en el código de referencia no afectan al funcionamiento del algoritmo LCE, esto es, el código ANSI C obtenido como salida de esta tarea es funcionalmente

equivalente al código de referencia. Esta verificación se ha realizado mediante la comparación entre la imagen comprimida tras las modificaciones realizadas y una imagen comprimida con anterioridad mediante el uso del software de referencia con las modificaciones aportadas en [24].

3.3.3 Generación del lenguaje de descripción hardware VHDL

CatapultC [35] es una herramienta de síntesis de alto nivel, en adelante HLS (del inglés, *High-Level Synthesis*). Se trata de un software comercial propiedad de la compañía Mentor Graphics Corporation. Permite la utilización de código C/C++ y la generación RTL en código de descripción hardware VHDL y Verilog, entre otros, para la síntesis hardware de dispositivos FPGA y ASIC. De este modo, se reduce el tiempo de desarrollo empleado en la implementación de un algoritmo si se compara con el empleado mediante el flujo de diseño RTL clásico. CatapultC permite a los usuarios imponer las restricciones de tiempo y área indicando la frecuencia de reloj a utilizar y la tecnología de destino.

La utilización básica de CatapultC no resulta complicada y sólo se requiere de la inserción de los archivos *.cpp* para iniciar el proceso. Posteriormente, CatapultC comprueba que el código fuente es sintácticamente correcto. Una vez realizada esta comprobación, se continúa con la configuración del diseño, en donde se selecciona el dispositivo a sintetizar y la frecuencia de reloj que se quiere utilizar. En el siguiente paso, CatapultC genera las restricciones para la arquitectura seleccionada. El usuario tiene la libertad de tomar decisiones arquitecturales con el objetivo de generar una implementación acorde a sus necesidades. Destaca la posibilidad de selección del tipo de interfaz de cada variable en el diseño (entrada, salida, bidireccional, interfaz a memoria RAM/ROM o cola de tipo FIFO) y la evaluación de iteraciones de los bucles del diseño pudiendo ser desenrollados o ejecutados de forma paralela. Al finalizar este paso, se genera el lenguaje de descripción hardware RTL seleccionado. Asimismo, CatapultC proporciona acceso a la herramienta Precision RTL Synthesis. Esta

herramienta permite realizar los pasos necesarios para la síntesis del código RTL generado.

En la Figura 3.7 y Figura 3.8 se muestra el aspecto que presentan los programas CatapultC y Precision RTL Synthesis empleados.

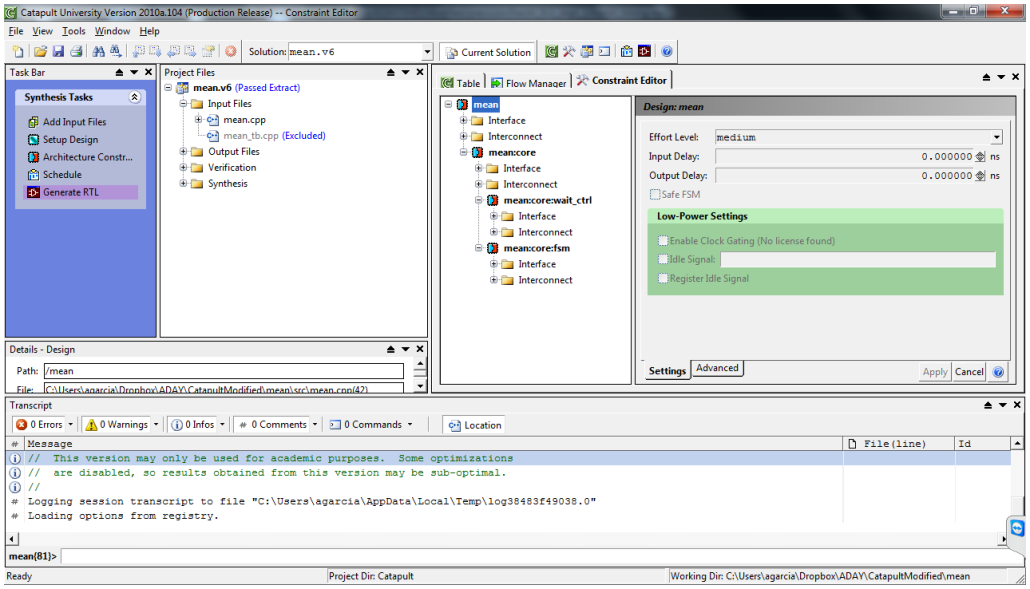


Figura 3.7 Interfaz de la herramienta CatapultC

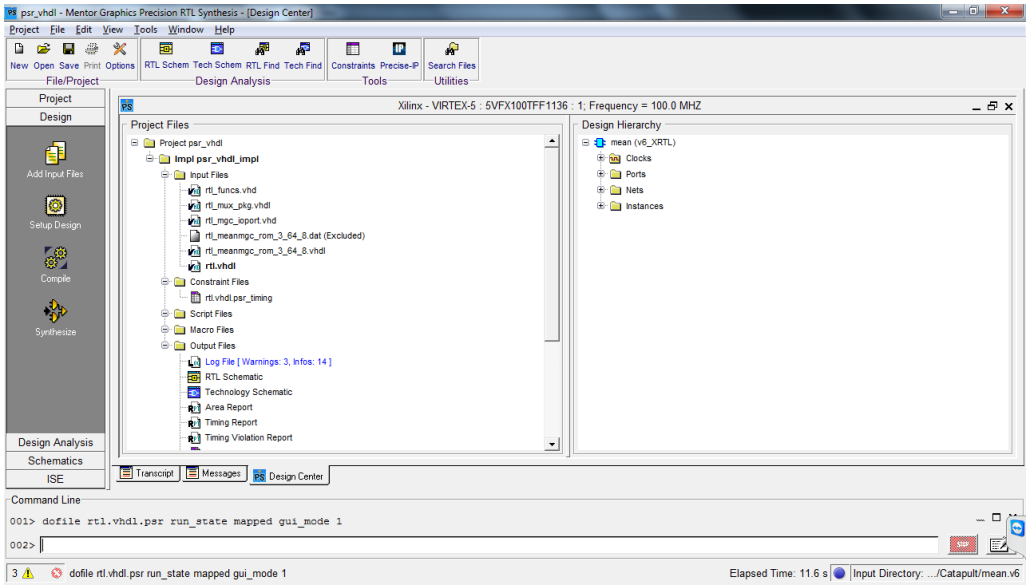


Figura 3.8 Interfaz de la herramienta Precision RTL Synthesis

3.3.3.1 Metodología de diseño en la herramienta CatapultC

En esta sección se describe la metodología de diseño que emplea la herramienta de síntesis de alto nivel CatapultC [35]. Conocer esta metodología, así como en qué medida afecta un cambio en los parámetros escogidos en el hardware generado, resulta fundamental para poder realizar su integración en tareas posteriores.

Paso 1. Escritura y verificación del código C/C++

El algoritmo escrito en lenguaje C/C++ es la entrada a la herramienta, afectando al hardware resultante en mayor medida que cualquier otro paso en la metodología. Por lo tanto, la implementación de alto nivel del algoritmo debe hacerse cuidadosamente.

Es probable que, cualquier código escrito en C/C++ e introducido en la herramienta CatapultC genere un hardware en código VHDL o Verilog sintetizable. Sin embargo, los resultados que se obtendrán, en términos de latencia, área y transferencia de datos, probablemente, no cumplirán con los requisitos impuestos y, lo que es peor, será complicado mejorar las prestaciones del sistema obtenido a partir de un código en C/C++ no escrito para ser usado en herramientas de síntesis de alto nivel.

Como ha quedado descrito, se parte de un código de referencia en ANSI C modificado para ser empleado en herramientas de síntesis de alto nivel. Asimismo, se han realizado las modificaciones necesarias en lenguaje ANSI C con el fin de obtener módulos funcionalmente independientes.

Paso 2. Configuración de las restricciones hardware a nivel global

En este paso, se especifica, por parte del usuario, la frecuencia de reloj, su polaridad y la presencia o ausencia de pines de control tales como *reset* y *enable*. Además, es en este paso donde se especifica la tecnología objeto de síntesis.

Por otra parte, se ha de especificar qué función constituye el módulo de más alto nivel (*top module*) de los comprendidos en la implementación. La herramienta, para el resto de funciones que sean llamadas desde este módulo, creará su implementación hardware. Mientras que, aquellas que queden fuera se considerarán parte de banco de pruebas con lo que no formarán parte del diseño hardware generado.

Paso 3. Especificación de las restricciones arquitecturales

CatapultC permite abstracciones de alto nivel proporcionando varios grados de libertad al usuario con el fin de decidir la manera en que el diseño será implementado a bajo nivel. En este paso, las principales características disponibles son:

- Posibilidad de mapeo de vectores a recursos de memoria internos.
- Decidir la forma en que se mapean las memorias en el diseño.
- Parámetros para la optimización de bucles en el diseño. Los más relevantes son: desenrollado de bucles (U), permitiendo la eliminación del bucle expandiéndolo completamente o dividiéndolo un número finito de veces, o la segmentación de bucles, permitiendo el comienzo de la segunda iteración antes de la finalización de la anterior, esto se fija mediante la variable intervalo de iniciación (II).
- Identificación de entradas, salidas y recursos a nivel global (por ejemplo: memorias ROMs, FIFO, etc.).
- Gestión de las interfaces de entrada/salida

Paso 4. Planificación del diseño

En este paso, CatapultC planifica el diseño hardware de acuerdo a las restricciones introducidas en los pasos 2 y 3. De forma gráfica proporciona un diagrama de Gantt en donde se representan las dependencias de datos existentes. De esta forma, el usuario obtiene una visión de cómo es planificado el diseño por la herramienta.

Asimismo, la herramienta proporciona información de utilidad con el fin de conocer cómo las restricciones impuestas en el paso 3 pueden ser optimizadas.

Por último, ya en este punto se obtienen estimaciones en cuanto a área, latencia y transferencia de datos.

Paso 5. Generación del código a nivel RTL

Es en este paso donde la herramienta genera los ficheros RTL del diseño. Se generan, además del código RTL en VHDL o Verilog, esquemáticos por lo que es posible identificar la ruta crítica del diseño.

En cuanto a los ficheros generados cabe destacar que se disponen de dos tipos de descripciones de bajo nivel del diseño:

- *cycle.vhdl*: contiene la descripción a nivel de comportamiento del diseño útil a la hora de realizar simulaciones.
- *rtl.vhdl*: descripción de diseño sintetizable. Empleada en la síntesis del mismo.

Tras la generación del código RTL, la herramienta proporciona un enlace directo a la herramienta Precision RTL Synthesis para realizar el emplazamiento y ruteo del diseño generado. A la salida de esta herramienta se proporcionan estimaciones con mayor detalle en términos de ocupación de recursos, latencia y transferencia de datos.

3.3.3.2 Flujo de verificación SCVerify

Complementariamente a la metodología de síntesis propuesta por CatapultC, existen una serie de flujos de verificación disponibles con el objetivo de garantizar el correcto comportamiento del código RTL generado.

De esta forma, el flujo de verificación SCVerify proporciona un entorno de trabajo para la validación de las salidas de CatapultC frente a la entrada original en C/C++ empleando un banco de pruebas proporcionado por el usuario en C/C++.

Se aplican los mismos estímulos tanto al diseño de alto nivel original como al código de bajo nivel generado y, mediante la comparación de las salidas de cada uno de los diseños, se valida que el comportamiento de ambos es idéntico.

Como soporte a las tareas realizadas, se ha creado, para cada uno de los módulos funcionalmente independientes, un banco de pruebas con valores aleatorios con el fin de garantizar a nivel de simulación que los módulos generados por la herramienta proporcionan los mismos resultados a estímulos idénticos a la entrada. La Figura 3.9, representa el último paso ya en la herramienta de simulación y depuración ModelSim. Se ha comprobado para cada módulo que número de errores obtenidos tras realizar la simulación es nulo. Este proceso ha sido repetido para todos y cada uno de los módulos que componen el algoritmo.

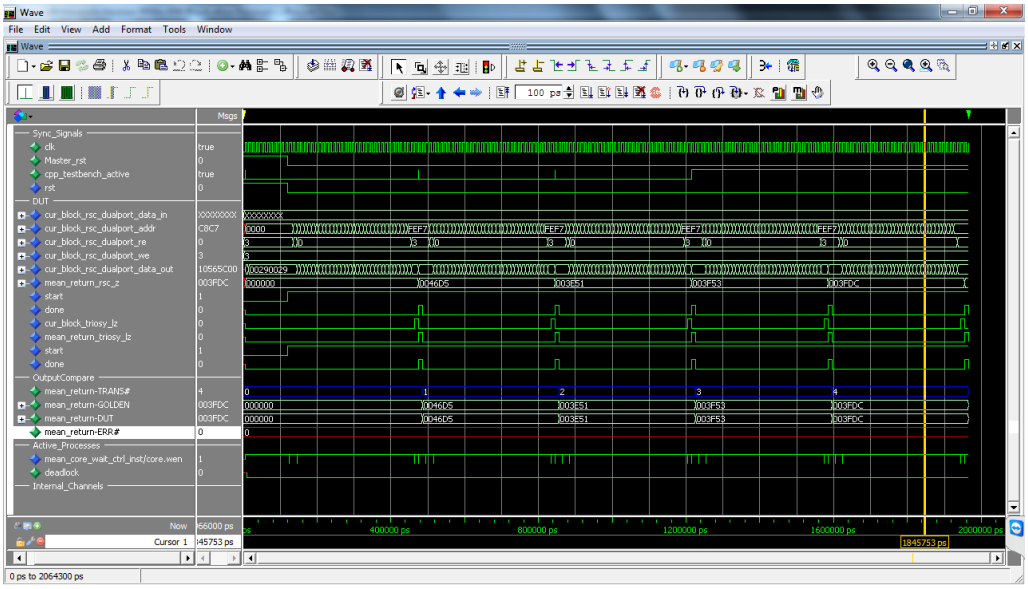


Figura 3.9 Interfaz de la herramienta ModelSim tras su llamada mediante CatapultC

Si bien esta verificación a nivel de bloque no resulta una tarea crítica en el diseño, ya que la herramienta garantiza en todo momento la coherencia del código generado, su realización permite ir con una mayor seguridad a tareas posteriores en las que se verifica con estímulos extraídos del software de referencia el correcto funcionamiento de la implementación del algoritmo LCE.

3.3.4 Implementación del algoritmo en lenguaje VHDL

A lo largo de esta sección se describe funcionalmente la arquitectura propuesta para la implementación del algoritmo de compresión de imágenes hiperespectrales LCE. La descripción se realizará siguiendo una estructura *top-down*, es decir, en primer lugar se presenta la arquitectura final, para luego bajar de nivel detallando cada uno de los módulos que la componen.

Para la implementación del algoritmo LCE se ha diseñado, implementado y verificado un módulo denominado controlador hardware, en adelante *hwctrl* (del inglés, *Hardware Controller*). Dentro de sus funciones principales destacan la conexión a bajo nivel de cada uno de los módulos obtenidos en la tarea anterior y gestionar el flujo de datos presente. Para ello ha de proporcionar adecuadamente los datos a la entrada de cada uno de dichos módulos así como almacenar sus valores de salida. Asimismo, ha de inferir internamente las memorias necesarias para el correcto funcionamiento de los módulos.

El esquema general del bloque funcional encargado de la implementación del algoritmo LCE se muestra en la Figura 3.10. En términos generales, presenta las siguientes características:

- El uso de memorias tanto a nivel de interfaces como interno es compartido por cada uno de los módulos. Esto se traduce en una mayor sencillez a la hora de abordar su implementación a costa de perder paralelismo en su ejecución.
- Empleo de señales de *start* y *done* en cada módulo. Esto facilita conocer en qué estado se encuentra cada módulo, es decir, si se encuentra a la espera de la activación de la señal de *start* para comenzar su procesado o si ha terminado con el mismo si la señal *done* es activada.
- Empleo de una señal de *reset* asíncrona para todos los módulos y registros.
- Señal de reloj, denominada *clk*, activa a nivel alto.

- Se han añadido señales para la validación de todos los datos disponibles a la salida del módulo, así como, una señal pulsada que indica la finalización del procesado de cada banda.
- Internamente se encuentra compuesto por ocho módulos que se corresponden con aquellos obtenidos tras su implementación mediante la herramienta CatapultC.

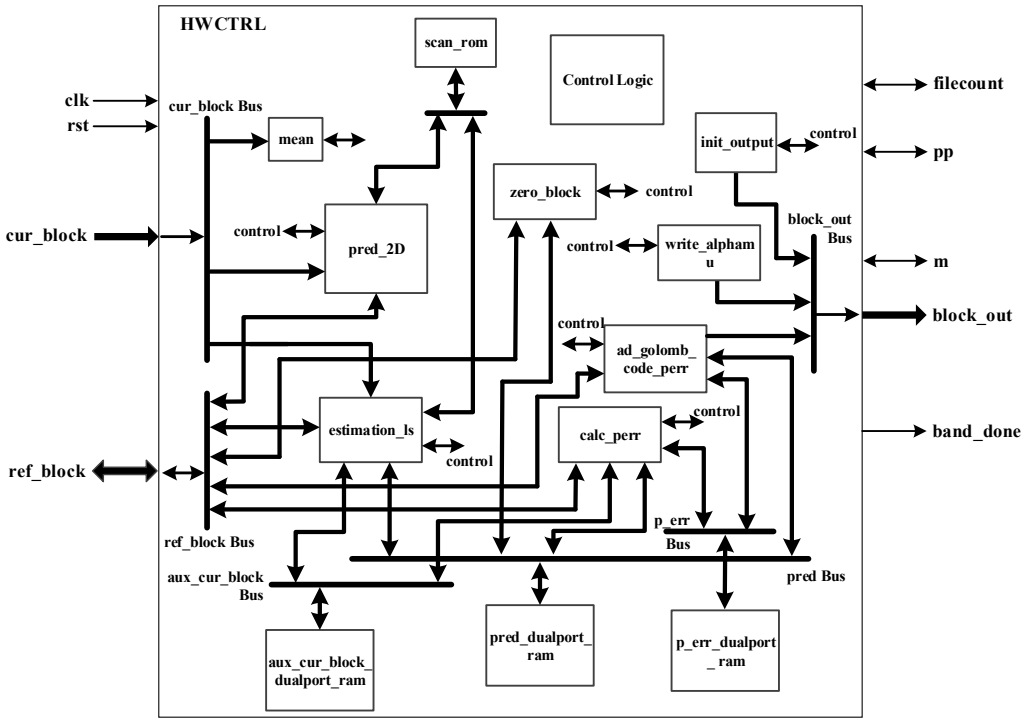


Figura 3.10 Diagrama de bloques funcional del algoritmo LCE implementado

A nivel de interfaces, visto el módulo como caja negra (*black-box*), el algoritmo LCE diseñado presenta la configuración mostrada en la Figura 3.11.

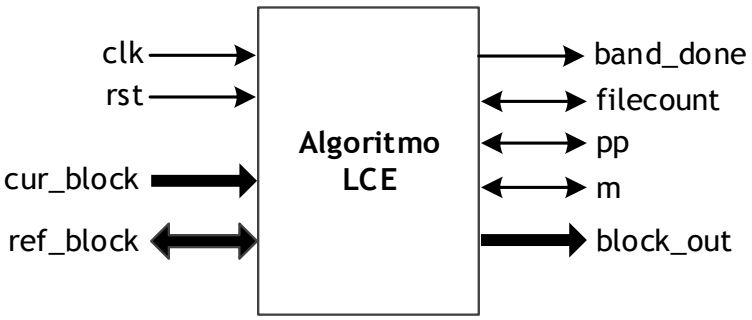


Figura 3.11 Interfaces del algoritmo LCE implementado

Su interfaz de comunicaciones presenta las señales mencionadas en la Tabla 3.1. Ésta, recoge, asimismo, la descripción de las señales de entrada/salida del algoritmo.

Nombre	Ancho	Tipo
clk	**1 bit**	**Entrada**
Reloj del sistema activo a nivel alto.		
rst	**1 bit**	**Entrada**
Reset asíncrono activo a nivel alto.		
band_done	**1 bit**	**Salida**
Señal activa a nivel alto que indica la finalización del procesado de una banda. Su duración es de un ciclo de la señal de reloj.		
cur_block	**32 bits**	**RAM doble puerto**
RAM de doble puerto que contiene el bloque actual.		
ref_block	**32 bits**	**RAM doble puerto**
RAM de doble puerto que contiene el bloque de referencia.		
filecount	**32 bits**	**Entrada / Salida**
Número de palabras de 32 bits escritas en la memoria de salida.		
pp	**32 bits**	**Entrada / Salida**
Memoria local intermedia de palabras código.		
m	**6 bits**	**Entrada / Salida**
Indicador de bits escritos en el buffer intermedio pp.		
block_out	**64 bits**	**Salida**
RAM de doble puerto que contiene el bloque de palabras código como resultado de la aplicación del algoritmo.		

Tabla 3.1 Descripción de las interfaces del algoritmo LCE implementado

Definida la funcionalidad de la arquitectura, el paso siguiente es su implementación. La implementación realizada se basa en su descripción en código HDL (del inglés, *Hardware Description Language*) para su modelado hardware. En esta descripción se ha definido la funcionalidad del circuito al nivel de abstracción RTL, con el fin de garantizar que el código resultante sea completamente sintetizable. El lenguaje de descripción VHDL [92] ha sido el utilizado para la implementación de este diseño. VHDL es un lenguaje de

descripción hardware ampliamente usado para describir sistemas electrónicos. Este lenguaje soporta el diseño, prueba e implementación de circuitos analógicos, digitales y mixtos a diferentes niveles de abstracción.

La integración se ha llevado a cabo siguiendo la siguiente estrategia. Primero, se ha realizado la integración obviando los controles con las interfaces de entrada y salida. Al resultado de esta integración se le añade el control de las interfaces del módulo. Posteriormente, se añaden al diseño las memorias necesarias para el funcionamiento del diseño. La integración se ha desarrollado de esta forma ya que así se plantea un diseño fácilmente adaptable a otro tipo de tecnologías.

Tanto las memorias RAM internas del módulo denominadas *pred_dualport_ram*, *p_err_dualport_ram* y *aux_cur_block_dualport_ram* como la memoria ROM denominada *scan_rom*, necesarias para la implementación del del algoritmo LCE han sido generadas a partir de las librerías disponibles en la herramienta CatapultC para memorias RAM de doble puerto y memorias ROM. Para esta generación, es necesario especificar el ancho del bus de cada memoria, la cantidad de palabras, el número de puertos, sus características (entrada/salida, sólo entrada, sólo salida, etc.) y definir las señales de control de cada una (*enable*, *reset*, *write enable*, etc.). El resultado es un código VHDL sintetizable del que se dispone de la interfaz para poder interactuar con la macrocélula correspondiente a la memoria de la FPGA, siendo únicamente necesario instanciar el código generado en la descripción del diseño que utilizará dicha memoria. Las características que necesitan especificarse de cada memoria se presentan en la Tabla 3.2. Todas las memorias empleadas registran el bus de direcciones.

Tipo de memoria	Ancho de bus (bits)	Num. de palabras	Puertos	Señales de control
RAM (*pred_dualport_ram*)	18	256	2 dual	reloj, reset, habilitación escritura/lectura
RAM (*p_err_dualport_ram*)	17	256	2 dual	reloj, reset, habilitación escritura/lectura
RAM (*aux_cur_block_dualport_ram*)	16	256	2 dual	reloj, reset, habilitación escritura/lectura
ROM (*scan_rom*)	8	64	1	-

Tabla 3.2 Especificación de las memorias internas de la implementación del algoritmo LCE

3.3.4.1 Módulo controlador de hardware

El módulo encargado de la correcta conexión de los módulos obtenidos mediante la herramienta CatapultC es el denominado *hwctrl*. Asimismo, a nivel de flujo de datos desempeña un papel fundamental ya que es este módulo el que controla dicho flujo durante la ejecución del algoritmo LCE. El controlador de hardware realiza la conexión de las instancias de cada uno de los módulos funcionalmente independientes.

Para la implementación de estas tareas se ha diseñado una máquina de estados finitos, en adelante FSM (del inglés, *Finite State Machine*). El diagrama de estados de la FSM perteneciente a este módulo es el que se representa en la Figura 3.12, y consta de nueve estados. El detalle de la funcionalidad presente en cada estado se describe en la Tabla 3.3.

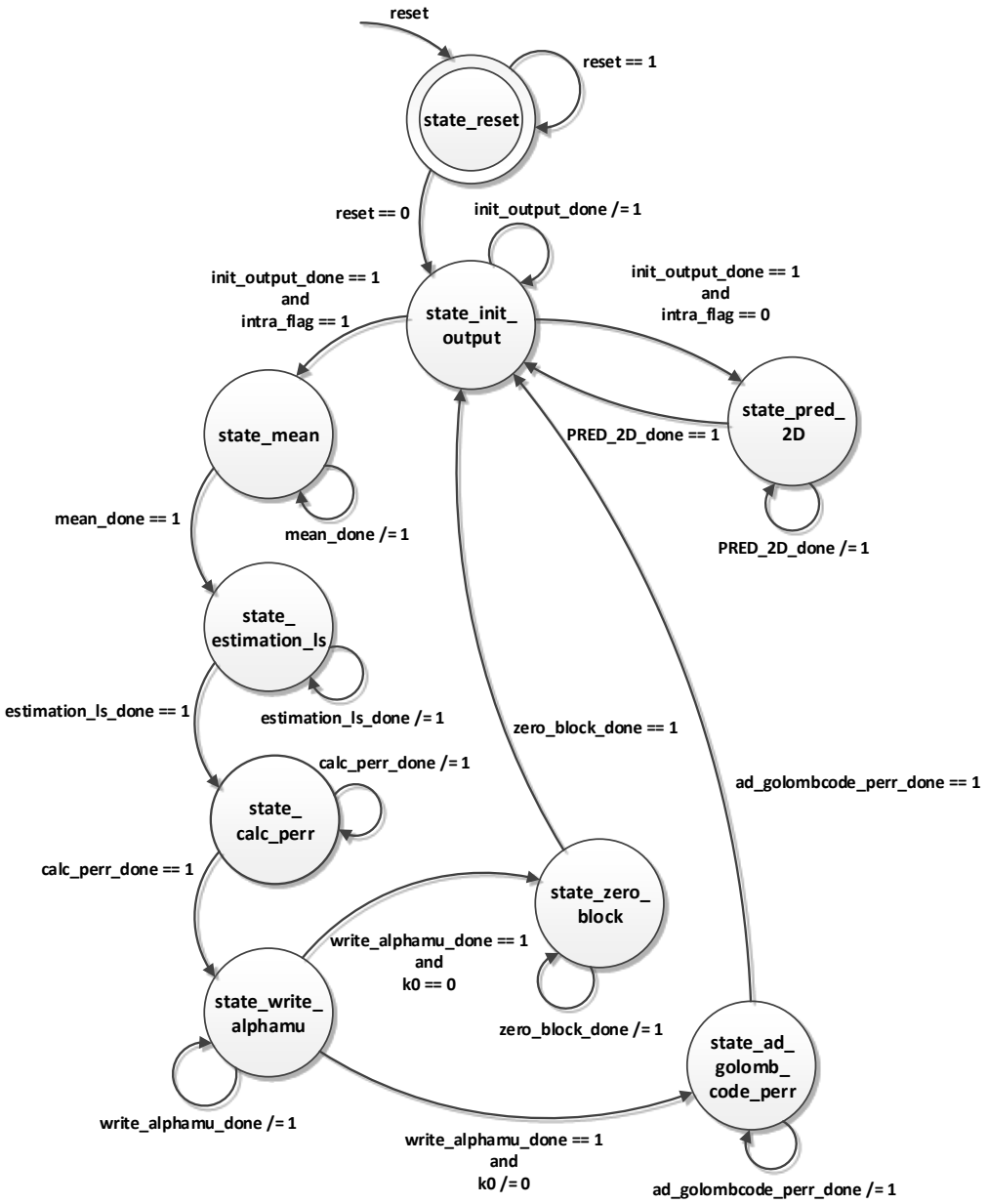


Figura 3.12 Diagrama de estados de la FSM perteneciente al bloque funcional hwctrl

Estado	Descripción
state_reset	Estado inicial. Se entra en él después de un reset. Se permanece en este estado hasta que la señal de reset se sitúa a nivel bajo, '0' lógico.
state_init_output	Estado de inicialización de la memoria de salida. En este estado se activa el módulo *init_output*. Se permanece en este estado hasta que la señal *init_output_done* se active durante un ciclo de la señal de reloj.
state_pred_2D	Predicción en 2 dimensiones o INTRA. En este estado se activa el módulo *pred_2D*. Se permanece en este estado hasta que la señal *pred_2D_done* sea activada durante un ciclo de la señal de reloj.
state_mean	Cálculo de la media del bloque actual. En este estado se activa el módulo *mean*. Se permanece en este estado hasta que la señal *mean_done* sea activada durante un ciclo de la señal de reloj.
state_estimation_ls	Cálculo de $\propto$. En este estado se activa el módulo *estimation_ls*. Se permanece en este estado hasta que la señal *estimation_ls_done* sea activada durante un ciclo de la señal de reloj.
state_calc_perr	Cálculo del error de predicción. En este estado se activa el módulo *calc_perr*. Se permanece en este estado hasta que la señal *calc_perr_done* sea activada durante un ciclo de la señal de reloj.

Estado	Descripción
state_write_alphamu	Escritura de $\propto$, media del bloque actual e indicador *zero_block*. En este estado se activa el módulo *write_alphamu*. Se permanece en este estado hasta que la señal *write_alphamu_done* sea activada durante un ciclo de la señal de reloj.
state_zero_block	Módulo de condición *zero_block*. En este estado se activa el módulo *zero_block*. Se permanece en este estado hasta que la señal *zero_block_done* sea activada durante un ciclo de la señal de reloj.
state_ad_golomb_code_perr	Escritura de palabras código. En este estado se activa el módulo *ad_golomb_code_perr*. Se permanece en este estado hasta que la señal *ad_golomb_code_perr_done* sea activada durante un ciclo de la señal de reloj.

Tabla 3.3 Descripción de los estados de la FSM del módulo hwctrl

3.3.4.2 Verificación funcional de la implementación hardware del algoritmo LCE

Durante la verificación funcional se toma como referencia la especificación previamente realizada para el diseño que es comparada con la implementación que se desea validar, esperando que ambas describan exactamente el mismo sistema. En nuestro caso, para hacer la verificación se sustituirán las especificaciones dadas en el algoritmo, por el software de referencia descrito en la sección 3.2 y que constituirá el modelo de referencia (*golden-reference*) en nuestra verificación. Este modelo se utilizará fundamentalmente para la verificación a nivel de bloque funcional.

La verificación de la implementación hardware del algoritmo LCE se ha realizado a nivel de sistema tomando como *golden-reference* la implementación en lenguaje ANSI C del algoritmo LCE. Para la correcta

verificación del módulo se han extraído, del software de referencia, los bloques de 16 x 16 píxeles de la imagen hiperespectral de entrada. Se han empleado las imágenes hiperespectrales del sensor AVIRIS (del inglés, *Airborne Visible/Infrared Imaging Spectrometer*), Indian Pines y World Trade Center como entradas en la verificación funcional.

En cuanto a los módulos funcionalmente independientes extraídos del software de referencia y presentados en la sección 3.3.2, cabe destacar que han sido previamente verificados haciendo uso del flujo de verificación SCVerify integrado en la herramienta CatapultC. Así pues, se garantiza su correcto desempeño durante su instanciación en el módulo *hwctrl*.

De este modo, la metodología seguida se basa los siguientes pasos:

1. Simulación mediante el flujo de verificación SCVerify de los módulos extraídos del software de referencia.
2. Simulación del software de referencia con el fin de obtener los estímulos que se aplican a la descripción RTL del bloque funcional y los resultados correctos.
3. Someter al módulo diseñado a los mismos estímulos que se han obtenido a partir del código de referencia.
4. En el caso de obtener las mismas salidas se podrá asegurar que la implementación del diseño coincide con lo establecido en el algoritmo LCE, si no es así se itera depurando el diseño implementado para, posteriormente volver al paso 3.

A continuación, se muestran una serie de capturas de la herramienta ModelSim empleada para la verificación y depuración de la implementación del algoritmo LCE. En la Figura 3.13, se muestra el inicio de la ejecución del algoritmo presentando los primeros estados donde se realizan las transiciones hasta llegar al estado donde se obtiene predicción en dos dimensiones para la primera banda $i = 0$. La Figura 3.14 presenta las distintas fases del algoritmo para bandas posteriores, es decir, $i \neq 0$. Por último, la Figura 3.15 muestra los valores de las señales internas más representativas del algoritmo durante su ejecución.

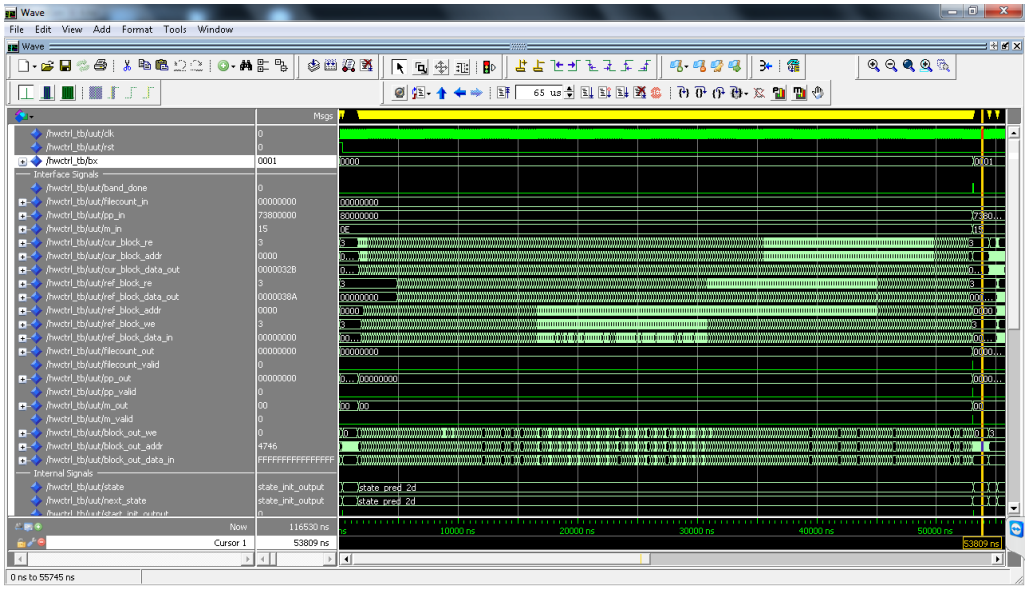


Figura 3.13 Verificación funcional - banda 0, $i = 0$

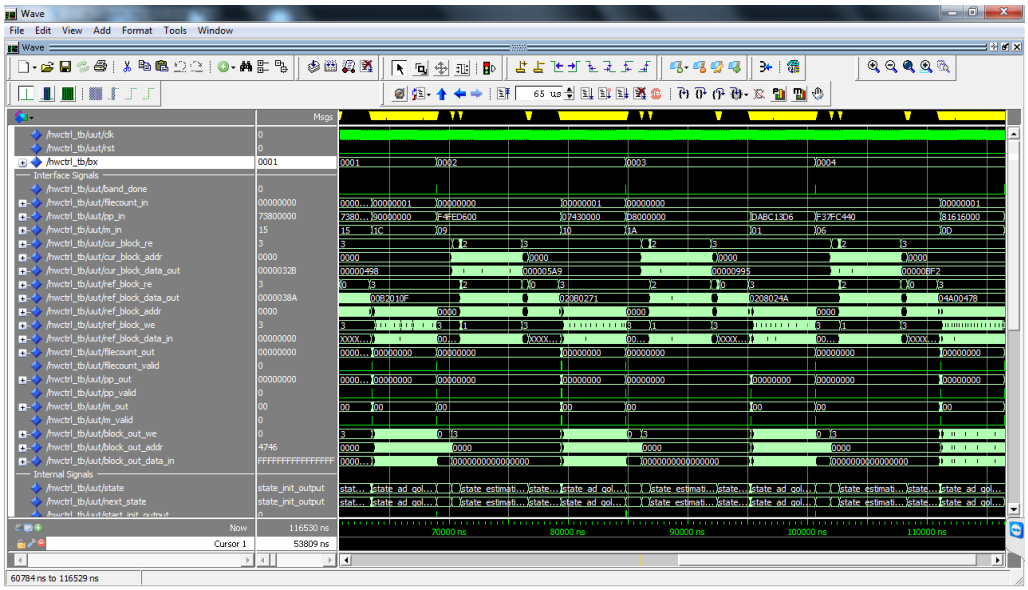


Figura 3.14 Verificación funcional - resto de bandas, $i \neq 0$

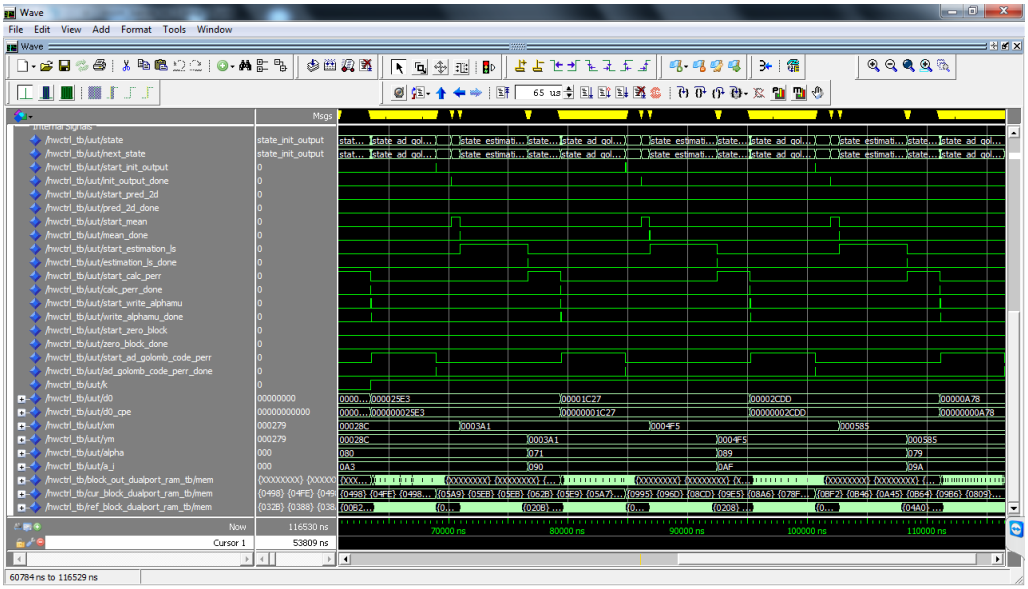


Figura 3.15 Verificación funcional - señales internas

3.4 Resultados obtenidos

En esta sección se presentan los resultados obtenidos para la síntesis de los módulos funcionalmente independientes así como para el algoritmo de compresión LCE en su conjunto. El algoritmo tomado como base para la implementación es el descrito, a nivel de pseudocódigo, en la Figura 3.5.

La síntesis se ha realizado sobre los dispositivos cualificados para operar en el entorno espacial XQR5VFX130 de la familia Virtex-5QV del fabricante Xilinx y sobre la FPGA RTAX2000S del fabricante Microsemi presentándose los resultados en esta sección.

La Tabla 3.4 muestra los recursos disponibles en el dispositivo XQR5VFX130 del fabricante Xilinx, mientras que, en la Tabla 3.5 se presentan los principales recursos de la FPGA RTAX2000S del fabricante Microsemi.

	Dispositivo
	XQR5VFX130
Celdas lógicas (LUTs)	81.920
Bloques de lógica configurable (CLBs)	20.480
D-Flip-Flops (DFF) o Latches	81.920
Bloques RAM (36 kb)	298
Bloques DSP	320
Bancos de Entrada/Salida	24
Entradas/Salidas disponibles para el usuario (máximo)	836

Tabla 3.4 Recursos disponibles en la FPGA Xilinx Virtex-5QV XQR5VFX130

	Dispositivo
	RTAX2000S
Capacidad	
Puertas lógicas equivalentes	2.000.000
Puertas ASIC equivalentes	250.000
Módulos	
Registros	10.752
Celdas combinacionales	21.504
Almacenamiento	
Bloques RAM	64 (36kB)
Procesado Digital de Señales	
Bloques DSP	0
Distribución de reloj	
Hardware dedicado	4
Rutado	4
Número de Entradas/Salidas	
Bancos de Entrada/Salida	8
Entradas/Salidas disponibles para el usuario (máximo)	684

Tabla 3.5 Recursos disponibles en la FPGA RTAX2000S

3.4.1 Resultados de la implementación a nivel de módulo mediante CatapultC

Una vez presentada, tanta la metodología de diseño empleada en la herramienta CatapultC, como el flujo de verificación usado, se introduce la implementación de los distintos módulos funcionalmente independientes obtenidos tras la ejecución de las tareas descritas con anterioridad.

La Tabla 3.6 presenta, para cada módulo, los bucles contenidos y su número de iteraciones además de la estrategia de optimización de bucles empleada en cada uno de ellos, es decir, los valores de desenrollado (U) e intervalo de iniciación (II). Resulta relevante remarcar que, como parte del proceso de optimización de bucles, se han aplicado diferentes opciones en cada uno de ellos presentándose únicamente en la Tabla 3.6 aquellas que proporcionan mejores resultados en términos de ocupación de recursos,

frecuencia de funcionamiento y tasa de datos. Asimismo, se muestra el margen con respecto al periodo de 13,33 ns (75 MHz) definido como objetivo.

Módulo	Número de Iteraciones	II U	Tasa de datos (muestras por ciclo)	Área (Puertas lógicas)	Margen (ns)
adgolomb.cpp	256	II=1 U=0	1	4146,73	1,42
calc_perr.cpp	256	II=1 U=2	1	3125,60	2,11
estimationls.cpp	256	II=3 U=0	1/3	4861,29	1,08
init_output.cpp	128	II=1 U=2	1	98,83	6,56
mean.cpp	64	II=1 U=2	1	436,32	1,80
pred_2D.cpp	256	II=10 U=0	1/10	4466,15	0,14
write_alphamu.cpp(*)	-	-	1	688,12	4,61
zero_block.cpp	256	II=1 U=2	1	213,97	4,39

(*) Módulo sin bucles asociados

Tabla 3.6 Optimización de bucles realizada para cada uno de los módulos

La tecnología objeto de la implementación a nivel de módulo realizada con la herramienta es una FPGA de la familia Virtex-5QV del fabricante Xilinx. En concreto, se ha empleado el dispositivo XQR5VFX130. Asimismo, se ha definido como objetivo, conseguir una frecuencia de funcionamiento por encima de los 75 MHz para cada uno de los módulos.

En la Tabla 3.7 se presentan los resultados de síntesis y frecuencia máxima de operación para cada uno de los módulos implementados con las optimizaciones a nivel de bucle realizadas. Se comprueba en ella que, la frecuencia máxima de operación, resulta superior a los 75 MHz fijados como objetivo en todos los casos.

Módulo	LUTs	CLBs	DFF o Latches	Bloq. RAM	Bloq. DSP	Frec. (MHz)
adgolomb.cpp	2502 (3,05%)	626 (3,06%)	1253 (1,53%)	1 (0,34%)	5 (1,56%)	85,02
calc_perr.cpp	893 (1,40%)	224 (1,40%)	450 (0,55%)	-	10 (3,91%)	108,71
estimationls.cpp	1187 (1,85%)	297 (1,86%)	595 (0,73%)	-	8 (3,13%)	89,06
init_output.cpp	71 (0,11%)	18 (0,39%)	25 (0,03%)	-	-	356,76
mean.cpp	251 (0,39%)	63 (0,39%)	85 (0,10%)	-	-	181,69
pred_2D.cpp	2888 (4,51%)	722 (4,51%)	1444 (1,76%)	-	2 (0,78%)	81,43
write_alphamu.cpp	503 (0,79%)	126 (0,79%)	253 (0,31%)	-	-	283,05
zero_block.cpp	152 (0,22%)	38 (0,22%)	76 (0,09%)	-	-	298,19

Tabla 3.7 Resultados de la síntesis de los módulos de manera independiente

3.4.2 Resultados de la implementación hardware del algoritmo de compresión LCE

Los resultados de síntesis de la implementación hardware del algoritmo LCE sobre la FPGA XQR5VFX130, se muestran en la Tabla 3.8. Nótese que estos resultados incluyen los módulos funcionalmente independientes generados además de incorporar la lógica de control interna y para el control de las interfaces presentada en el módulo *hwctrl*.

Módulo	LUTs	CLBs	DFF o Latches	Bloq. RAM	Bloq. DSP	Frecuencia (MHz)
Algoritmo LCE	7746 (12,1%)	1937 (12,11%)	4281 (5,23%)	4 (0,88%)	25 (9,77%)	86,96

Tabla 3.8 Resultados de la síntesis del algoritmo LCE – Virtex-5QV (XQR5VFX130)

Con el fin de aportar datos adicionales sobre la bondad de la implementación realizada y obtener la mayor información posible para poder comparar los resultados de síntesis obtenidos, en la Tabla 3.9, se presentan los resultados de la síntesis de la implementación hardware del algoritmo LCE sobre la FPGA RTAX2000S.

Módulo	Registros	Comb.	Bloq. RAM	Frecuencia (MHz)
Algoritmo LCE	5733 (53,32%)	18101 (84,18%)	7 (10,94%)	18,65

Tabla 3.9 Resultados de la síntesis del algoritmo LCE – RTAX (RTAX2000S)

De los resultados presentados en la Tabla 3.8 y la Tabla 3.9, se concluye que, el algoritmo de compresión de imágenes hiperespectrales LCE, resulta un algoritmo viable para ser implementado en diferentes dispositivos FPGA calificados para operar en el entorno espacial. En el peor de los casos se emplean el 84 % de las celdas combinacionales de la FPGA RTAX2000S para su operación. La máxima frecuencia de operación se obtiene en el caso de la

FPGA XQR5VFX130 y es cercana a los 87 MHz, mientras que la mínima frecuencia se alcanza en el caso de la RTAX2000S siendo de 18,6 MHz.

Con el fin de evaluar las prestaciones de la implementación hardware de un algoritmo de compresión resulta interesante la evaluación del número de muestras por segundo proporcionada a la salida del codificador. En este sentido, se ha examinado la implementación hardware del algoritmo LCE hallándose el número de muestras por ciclo alcanzado. La Tabla 3.10 muestra tanto el número de ciclos requerido para la predicción de la primera banda como para el resto de bandas. Además, se muestra la latencia del diseño para cada banda. Dado que para la primera banda, $i = 0$, se realiza la predicción en dos dimensiones, el número de ciclos para esta banda difiere con respecto del número de ciclos requeridos para el cálculo del resto de bandas, $i \neq 0$.

Banda	Latencia	Muestras por ciclo
Primera banda (predicción 2D, $i = 0$)	2640 ciclos de reloj	1 muestra / 10 ciclos de reloj
Resto de bandas ($i \neq 0$)	793 ciclos de reloj	1 muestra / 3 ciclos de reloj

Tabla 3.10 Latencia y número de muestras por ciclo de reloj de la implementación hardware del algoritmo LCE

Para cada bloque de 16 x 16 muestras, se presenta en la Tabla 3.11, el número de muestras por ciclo que se alcanza como media para las imágenes de los sensores AIRS, AVIRIS y MODIS. Asimismo, se presenta la tasa de datos en términos de muestras por segundo máxima que alcanza la implementación hardware del algoritmo LCE tras el emplazamiento y ruteo a la máxima frecuencia de operación obtenida.

Sensor	Número de bandas	Muestras por ciclo (media)	Muestras por segundo (18,65 MHz)	Muestras por segundo (86,96 MHz)
AIRS	1501	1 muestra / 3,114 ciclos	5,99 Mmuestras/s	27,93 Mmuestras/s
AVIRIS	224	1 muestra / 3,142 ciclos	5,93 Mmuestras/s	27,68 Mmuestras/s
MODIS	17	1 muestra / 3,535 ciclos	5,27 Mmuestras/s	24,6 Mmuestras/s

Tabla 3.11 Muestras por ciclo de media para imágenes de los sensores AIRS, AVIRIS y MODIS

La máxima tasa de datos obtenida es de 27,93 Mmuestras/s para la implementación hardware del algoritmo sobre la FPGA XQR5VFX130 e imágenes del sensor AIRS, mientras que la mínima tasa de datos de 5,27 Mmuestras/s se ha obtenido para la FPGA RTAX2000S e imágenes del sensor MODIS. Las tasas de datos obtenidas pueden ser mejoradas, mediante la implementación de varios algoritmos LCE operando en paralelo sobre la misma FPGA. Esto puede ser realizado gracias a que la operación del algoritmo LCE opera sobre bloques independientes de 16 x 16 píxeles. Se ha de resaltar el hecho de que, en este caso, es necesaria la inclusión de lógica adicional para la planificación de la operación paralela de los distintos algoritmos LCE implementados.

Si se realiza la comparación de los resultados de implementación obtenidos para el algoritmo LCE con los disponibles en el estado del arte, se concluye que el algoritmo de compresión con pérdidas LCE presenta una menor ocupación de recursos y una mayor tasa de datos gracias a su menor complejidad (ver capítulo 5). Por otra parte, las implementaciones del algoritmo LCE sobre los dispositivos FPGA presentados alcanzan tasas de datos inferiores a las implementaciones realizadas sobre GPU [24]. Sin embargo, estos dispositivos no se encuentran en la actualidad cualificados para operar en el entorno espacial y, además, presentan consumos de potencia demasiado elevados para poder ser desplegados en un satélite.

3.5 Conclusiones

En este capítulo se presenta la implementación hardware del algoritmo de compresión de imágenes hiperespectrales LCE. Esta implementación se ha diseñado con el fin de realizar una caracterización del algoritmo en cuanto a recursos necesarios para su operación empleando una aproximación mediante el empleo de la herramienta CatapultC. Para ello, se analiza el código de referencia ANSI C del algoritmo, determinando las funciones que lo componen. Se ha dividido el código de referencia y se ha establecido la funcionalidad de cada uno de los diferentes bloques funcionales que lo conforman. A continuación, se ha realizado su implementación y verificación mediante la herramienta CatapultC. Se diseña un bloque funcional para la conexión de cada uno de los módulos implementados elaborándose la descripción en código VHDL de la arquitectura diseñada, verificándose para comprobar su correcto funcionamiento.

Se ha realizado la síntesis del código diseñado sobre dos dispositivos FPGA calificados para el espacio de diferentes fabricantes: Xilinx Virtex-5QV (XQR5VFX130) y Microsemi RTAX (RTAX2000S). Se ha comprobado que, el diseño, puede implementarse en los dispositivos FPGA objetivo, proporcionando un peor caso de ocupación del 84,18 % de celdas combinacionales sobre la FPGA RTAX2000S. Por el contrario, el mejor caso obtenido para el dispositivo XQR5VFX130 proporciona una máxima ocupación de LUTs del 12,1 %. Se analizaron los resultados de frecuencia máxima de operación y transferencia de datos que se obtienen al sintetizar el diseño en ambos dispositivos obteniéndose una frecuencia máxima de operación de 87 MHz con una tasa de bits máxima de 27,93 Mmuestras/s para imágenes del sensor AIRS sobre la FPGA XQR5VFX130.

Los resultados obtenidos satisfacen los objetivos propuestos en esta tesis doctoral ya que se propone una implementación hardware viable del algoritmo de compresión de imágenes hiperespectrales LCE de naturaleza predictiva y con pérdidas, para ser empleado en dispositivos FPGA cualificados para el entorno espacial mediante metodologías de síntesis de alto nivel.

Capítulo 4

Implementación Hardware de las Operaciones Aritméticas de la Transformada Ortogonal por Parejas

---

La transformada ortogonal por parejas (POT) puede emplearse como decorrelador espectral en algoritmos de compresión de imágenes hiperespectrales basados en transformada.

Como se presenta en el capítulo de análisis del estado del arte, la principal desventaja de los algoritmos basados en transformada es su mayor complejidad para ser implementados a bordo si son comparados con algoritmos de tipo predictivo. La transformada POT, gracias a su reducida complejidad, se presenta como una alternativa atractiva a ser implementada como decorrelador espectral en los sistemas de compresión de imágenes hiperespectrales a bordo de satélites de observación de la tierra.

4.1 Introducción

La Transformada Ortogonal por Parejas, en adelante POT (del inglés, *Pairwise Orthogonal Transform*), se encuadra dentro de los algoritmos de compresión de imágenes hiperespectrales estudiados como un algoritmo basado en transformada que, dadas sus características, puede ser empleado tanto en la compresión sin pérdidas como con pérdidas. Este algoritmo nace en el comité CCSDS (del inglés, *Consultative Committee for Space Data Systems*) con la idea de ser empleado como decorrelador espectral en la extensión del estándar para compresión de imágenes en 2D, el CCSDS 122.0-B-1 [33], de manera que pueda ser utilizado para la compresión de imágenes hiperespectrales.

La transformada POT es una transformada adaptativa de baja complejidad (en términos de coste computacional y almacenamiento) que requiere únicamente pequeñas cantidades de información de control para su operación. Los datos provenientes de los sensores hiperespectrales se procesan línea a línea. El algoritmo agrupa estas líneas y realiza la transformada por parejas, adaptándose de este modo a las tecnologías de sensores disponibles a bordo de satélites de observación de la tierra.

En este capítulo se describen los detalles de la transformada POT haciendo especial hincapié en las características que lo convierten en adecuado para su implementación en las tecnologías hardware disponibles en el sector espacial. En este trabajo se demuestra su viabilidad para ser implementada en dispositivos de tipo FPGA disponibles en el sector espacial.

Los resultados obtenidos resultan relevantes a la hora de evaluar la inclusión de la transformada POT en una determinada misión espacial para la decorrelación espectral en la extensión del estándar CCSDS 122.0. En concreto, los elementos aritméticos de la aproximación entera de la transformada POT se realizan directamente desde una descripción RTL lo que permite obtener una baja ocupación en términos de recursos, haciendo viable su síntesis sobre la familia de dispositivos cualificados para el espacio RTAX2000S y RTAX2000S-DSP del fabricante Microsemi.

4.2 Descripción de la transformada POT

La transformada POT ha sido desarrollada en la Universidad Autónoma de Barcelona (UAB) en el seno del Departamento de Ingeniería de la Información y de las Comunicaciones. Se basa en la aplicación de la estrategia *divide-y-vencerás* a la transformada KLT (del inglés, *Karhunen-Loève Transform*) [31]. La transformada resultante es una composición de transformadas KLTs en la que cada una emplea, únicamente, dos componentes de la imagen como entrada.

La transformada KLT es considerada como la transformada óptima para la eliminación de la correlación espectral en imágenes hiperespectrales [70]. En esta transformada, todos los componentes de entrada son decorrelados entre sí independientemente de la cantidad de energía que compartan, es decir, independientemente del valor de su covarianza. Por el contrario, la transformada POT presenta una estructura que decorrela los componentes que comparten una alta cantidad de energía, mientras que ignora los componentes con baja energía, ya que éstos presentan una baja incidencia en términos de compresión.

La transformada POT se organiza en múltiples niveles. Dentro del primer nivel, se aplica una transformada KLT de dos componentes a cada pareja de componentes consecutivos de la imagen. En los siguientes niveles, se repite la misma aproximación con la particularidad de que en cada nivel, se decorrela únicamente la componente fundamental (o principal) del nivel anterior. La Figura 4.1 muestra el funcionamiento descrito de la transformada POT para el caso en el que los componentes de entrada sean pares, caso en el que la transformada se denomina transformada POT balanceada. En el caso de que los componentes de entrada sean impares, Figura 4.2, este componente impar se propaga al siguiente nivel del algoritmo. Los componentes impares se seleccionan alternativamente de los extremos derecho e izquierdo de cada nivel. En este caso, la transformada recibe el nombre de transformada POT no balanceada al existir un número impar de componentes de entrada.

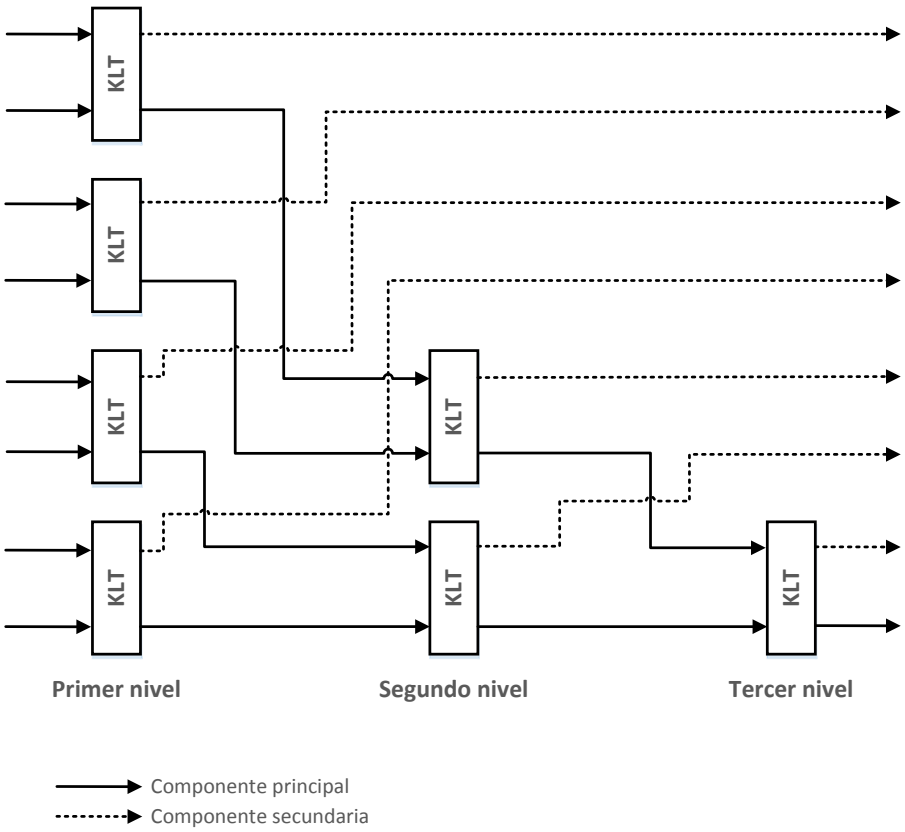


Figura 4.1 Estructura multinivel de la transformada POT balanceada (ocho componentes de entrada)

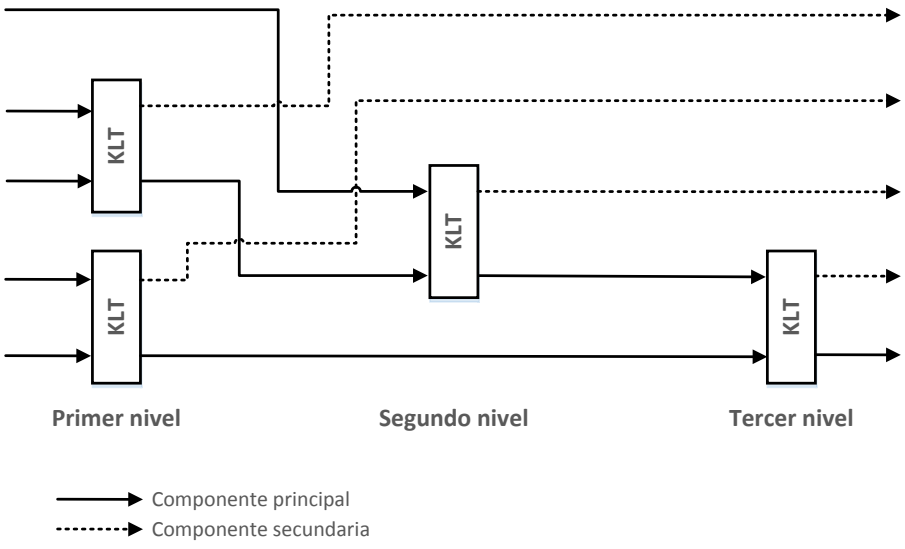


Figura 4.2 Estructura multinivel de la transformada POT no-balanceada (cinco componentes de entrada)

Con esta estructura de funcionamiento, la mayor parte de la energía se acumula en las componentes principales ya que cada KLT de dos componentes opera como la transformada KLT clásica.

La definición clásica de la transformada KLT de N componentes proporciona a su salida un Y tal que:

$$Y = \mathrm{KLT}_{\sum_X}(X) = Q^T X \tag{4.1}$$

En donde,

- $\sum_X = \left(\frac{1}{N}\right) XX^T$ representa la matriz de covarianza de X.
- Q es la matriz de transformación ortogonal resultado de la descomposición en autovalores de $\sum_X = Q\Lambda Q^{-1}$ ($\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_N)$, $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_N|$). La matriz diagonal Λ representa la covarianza de la matriz Y, mientras que, los valores $\lambda_1, \dots, \lambda_N$ representan las varianzas de cada componente tras la transformada.

Es necesario destacar que los valores de entrada han de presentar media nula. En caso contrario, se ha de incluir una etapa de eliminación de la media con el fin de centrar los valores en torno a cero.

Si se simplifica el proceso de descomposición en autovalores, tal y como es el caso de la transformada KLT de dos componentes, la complejidad en la implementación de la transformada disminuye notablemente. La solución aportada por la transformada POT hace uso de la transformada KLT de dos componentes para proporcionar una solución numérica estable que además no requiere de un proceso iterativo en el cálculo de la matriz de transformación Q. Por lo tanto, la transformada POT se fundamenta en el cálculo de la siguiente expresión:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = Q^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{4.2}$$

Por otro lado, pueden expresarse las matrices Q y Λ como:

$$Q = \begin{pmatrix} p & q \\ t & u \end{pmatrix} \tag{4.3}$$

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \tag{4.4}$$

Obteniéndose t, q, p, u, λ_1 y λ_2 mediante las siguientes expresiones:

$$t = -q = \frac{b}{|b|}\sqrt{\frac{1}{2} - \frac{(a-d)}{2s}} \tag{4.5}$$

$$p = u = \sqrt{\frac{1}{2} - \frac{(a-d)}{2s}} = \sqrt{1-t^2} \tag{4.6}$$

$$\lambda_1 = \frac{a+d+s}{2} \tag{4.7}$$

$$\lambda_2 = \frac{a+d-s}{2} \tag{4.8}$$

$$s = \sqrt{(a-d)^2 + 4b^2} \tag{4.9}$$

Siendo, la matriz de covarianza $\sum_X$ de los componentes de entrada X:

$$\Sigma_X = \begin{pmatrix} a & b \\ b & d \end{pmatrix} \tag{4.10}$$

Para garantizar que la matriz de transformación Q tiene una solución estable se ha de definir $b \neq 0$ de manera que el resultado de $b/|b|$ será en todos los casos $+1$ o -1. Si las entradas de la transformada KLT de dos componentes no comparten energía, el valor de s se aproximará a 0, lo que puede causar una división por 0. Con el fin de evitarlo, puede emplearse una matriz identidad como matriz Q para garantizar la estabilidad de la solución.

En general, el cálculo de la transformada KLT puede descomponerse en una red de pesos que se aplica de forma que no se introduzcan pérdidas en el algoritmo. Esta estructura, se conoce con el nombre de transformada RKLT (del inglés, *Reversible KLT*) [93] [94]. La transformada RKLT se basa en la descomposición de la matriz de transformación Q en matrices elementales en la que cada una equivale a la aplicación de una secuencia de pesos. El

resultado de la transformada se obtiene como una multiplicación sucesiva de matrices elementales. Dado que la transformada POT consiste en la aplicación multinivel de transformadas KLT de dos componentes, puede aplicarse esta descomposición en pesos para su cómputo. De este modo, el cálculo de Q^T en el algoritmo de la transformada POT puede expresarse de la siguiente forma:

$$Q^T = \begin{pmatrix} p & t \\ q & u \end{pmatrix} = \begin{pmatrix} \sqrt{1-t^2} & t \\ -t & \sqrt{1-t^2} \end{pmatrix} = \\ = \begin{pmatrix} 1 & 0 \\ 0 & s \end{pmatrix} P \begin{pmatrix} 1 & 0 \\ w_3 & 1 \end{pmatrix} \begin{pmatrix} 1 & w_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ w_1 & 1 \end{pmatrix} \tag{4.11}$$

La ecuación (4.11) admite dos soluciones:

$$\begin{cases} P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix};\ w_1 = w_3 = \dfrac{p-1}{t}\ ; w_2 = t \quad ; s = 1 \ , & \text{si } |t| \geq |p| \\ P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix};\ w_1 = w_3 = \dfrac{1-t}{p}\ ; w_2 = -p\ ; s = -1, & \text{si } |t| < |p| \end{cases} \tag{4.12}$$

Al ser Q una matriz ortogonal, se satisface la condición $t^2 + p^2 = 1$, con lo que el establecer la condición $|t| \geq |p|$ es suficiente para garantizar que no existe división por cero. Nótese que al existir dos soluciones posibles a la ecuación (4.11) es necesario añadir en la red de pesos una permuta condicional. La Figura 4.3 presenta la red de pesos que permite el cálculo de cada una de las transformadas KLTs de dos componentes que forman parte de la estructura de la transformada POT, es decir, la representación gráfica de la solución a la ecuación (4.2).

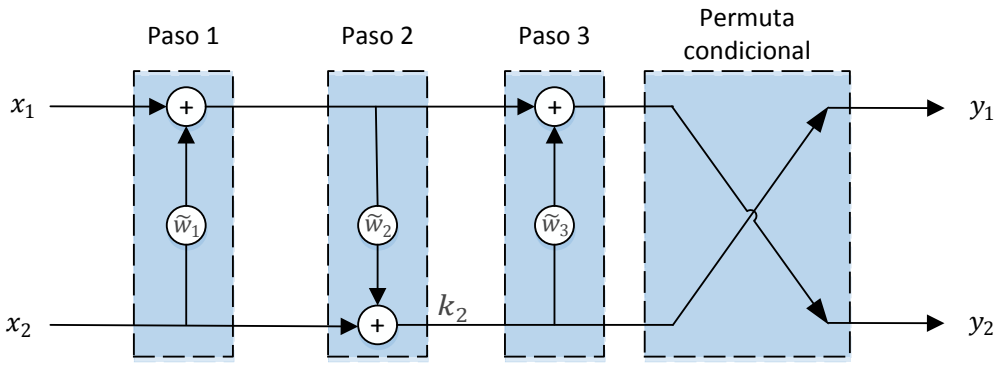


Figura 4.3 Estructura de la red de pesos de la transformada KLT de dos componentes

En [31] se destacan las principales ventajas de la transformada POT con respecto a la transformada KLT original:

- **Información de control**. La transformada POT requiere de una pequeña cantidad de información de control a transmitir para invertir las operaciones en el receptor. Por ejemplo, para una imagen hiperespectral de 224 bandas tomada con el espectrómetro AVIRIS (del inglés, *Airbone Visible/Infrared Imaging Spectrometer*) se requieren 101kB de información de control para la transformada KLT mientras que, en la transformada POT se necesitan únicamente 1,34 kB.
- **Capacidad de procesado por líneas**. Esta característica resulta especialmente relevante ya que los sensores hiperespectrales actuales, en su mayoría, capturan la información pixel a pixel. Tal modo de funcionamiento es característico de los sensores *whiskbroom* (que capturan la información espectral de las bandas intercaladas por píxel). Por su lado, los sensores *pushbroom* capturan todas las bandas espectrales de una línea de píxeles.
- **Coste computacional**. La transformada KLT presenta un coste computacional mayor que la transformada POT. El factor dominante en la transformada KLT se encuentra asociado al cuadrado del número de componentes de entrada $O(n^2)$

mientras que, por el contrario, en la transformada POT el factor dominante asociado a las muestras de entrada es lineal $O(n)$.

- **Necesidad de almacenamiento**. La transformada KLT presenta requisitos de almacenamiento proporcionales al tamaño total de la imagen, generalmente del orden de cientos de megabytes. Derivado de la posibilidad de procesado por líneas, la transformada POT requiere únicamente del almacenamiento del orden de kilobytes.

La expansión del rango dinámico de un algoritmo se define como el incremento en bits del rango dinámico de las muestras de entrada debido a los cálculos presentes en el propio algoritmo. En el caso de la transformada POT, el rango dinámico de los datos se incrementa en 0,5 bits en cada nivel más un bit derivado de la etapa de eliminación de la media [31]. Generalizando, una vez aplicada la transformada POT a los múltiples niveles necesarios para el procesado de todos los componentes de entrada, la expansión del rango dinámico total toma la expresión $0{,}5 \cdot \lceil \log_2(n) \rceil + 1$ bits, siendo n el número de componentes de entrada. Por ejemplo, el incremento del rango dinámico de la transformada POT cuando se emplea en la decorrelación espectral de imágenes hiperespectrales del sensor AVIRIS (224 bandas) resulta en 5 bits. Esta característica resulta un inconveniente cuando se aborda la implementación hardware de la transformada POT. Limitar la máxima expansión del rango dinámico de un algoritmo con el objetivo de ser implementado a nivel hardware proporciona la ventaja de poder dimensionar a priori, tanto las operaciones internas a nivel de bit evitando truncamientos no deseados como sus necesidades de almacenamiento.

4.2.1 Transformada *Isorange* POT

La transformada *Isorange* POT [32] nace con el objetivo de limitar la máxima expansión del rango dinámico de la transformada POT. En ella, se modifica la red de pesos de la transformada POT original, de manera que se evita el empleo de aritmética en punto flotante, y se utiliza solamente aritmética entera en punto fijo y redondeos en las operaciones. La nueva solución Y^+ se trata, por lo tanto, de una aproximación de la ecuación (4.2)

ya que se introducen errores en su cálculo. Estos errores responden a dos causas principales: la limitación de la precisión en las operaciones y los redondeos empleados en la red de pesos. Los resultados muestran que estas modificaciones producen ligeras penalizaciones en la aplicación de la transformada *Isorange* POT tanto para el caso con pérdidas como sin pérdidas. La máxima penalización en la compresión con pérdidas es de 0,93 dB en la relación señal a ruido, SNR (del inglés, *Signal to Noise Ratio*), mientras que, cuando se emplea la transformada en la compresión sin pérdidas la diferencia en el ratio de compresión es de 0,02 bpp.

En su cálculo, la transformada *Isorange* POT introduce en la red de pesos una matriz de desplazamiento para cada transformada KLT de dos componentes. Esta matriz se define mediante la ecuación (4.13), en donde $b = 1$.

$$S_b = \begin{pmatrix} 2^{-b} & 0 \\ 0 & 2^{+b} \end{pmatrix} \tag{4.13}$$

El objetivo de la matriz de desplazamiento es multiplicar en cada nivel la componente principal por el factor 2^{-b} con el fin de reducir en b bits su rango dinámico. Nótese que, a pesar de que la componente principal es la propagada en cada transformada KLT de dos componentes, Figura 4.1 y Figura 4.2, la inclusión de este desplazamiento en cada nivel evita la expansión de su rango dinámico. Por el contrario, la componente secundaria se multiplica por el factor 2^{+b}, lo que produce un incremento de b bits en su rango dinámico. La componente secundaria se emplea únicamente en una de las transformadas KLT de dos componentes presentes en el algoritmo. Por lo tanto, la expansión del rango dinámico de la producida por la matriz de desplazamiento queda limitada a los b bits ($b = 1$) de la componente secundaria.

Mediante las modificaciones descritas se limita la expansión del rango dinámico de los datos a 3 bits en el peor caso. Esta expansión de 3 bits se desglosa del siguiente modo: 0,5 bits de la transformada KLT de dos componentes original, 1 bit derivado de la etapa de eliminación de la media,

1 bit por las operaciones de desplazamiento y 0,5 bits por operaciones de redondeo y aritmética entera de punto fijo.

4.2.2 Aproximación entera de la transformada POT

La aproximación entera de la transformada POT se fundamenta en la transformada *Isorange* POT presentada en la sección anterior. En ella se modifican las operaciones de la transformada *Isorange* POT de forma que se emplea únicamente aritmética entera. El empleo de aritmética entera permite obtener resultados exactos a nivel de bit, facilitando la verificación de la implementación hardware de la transformada. Además, esta definición en aritmética entera permite la posterior comparación de los resultados obtenidos por la implementación presentada en este capítulo con posibles implementaciones futuras.

Como se describe en las secciones anteriores, la transformada POT se fundamenta una composición de tres tipos de operaciones que, cuando se combinan en una estructura multinivel, proporcionan una imagen transformada. Dichas operaciones reciben el nombre de: eliminación de la media, operación por parejas balanceada y operación por parejas no-balanceada.

En las siguientes secciones se presenta la definición de las operaciones aritméticas de la aproximación entera de la transformada POT, que la hacen apropiada para ser implementadas en el hardware disponible a bordo de un satélite.

4.2.2.1 Operación de eliminación de la media

La operación de eliminación de la media mapea una secuencia de valores enteros de entrada denominados $x_i = \{x_0, x_1, \dots, x_{N-1}\}$, siendo $N \geq 1$, en una secuencia de salida $y_i = \{y_0, y_1, \dots, y_{N-1}\}$. Los valores originales son modificados de tal forma que la secuencia de salida tiene una media aritmética de valor aproximadamente cero. En la Figura 4.4 se muestra de manera gráfica esta operación. En la misma, se observa que, junto con la

secuencia de salida, esta operación de eliminación de la media proporciona el valor calculado de la media.

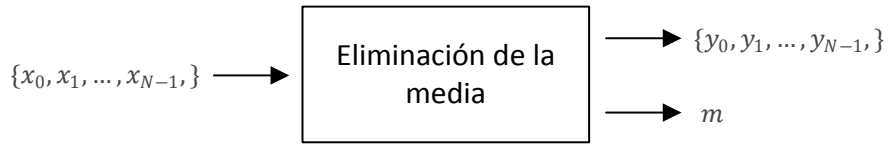


Figura 4.4 Operación de eliminación de la media

Dada una secuencia de entrada dada x_i, el valor m es una aproximación entera de la media aritmética de dicha secuencia, su valor se calcula mediante la siguiente ecuación:

$$m = \left\lfloor \frac{1}{N} \sum_{i=0}^{N-1} x_i \right\rfloor \tag{4.14}$$

Una vez calculado el valor de m, el resultado de la secuencia de salida y_i se obtiene según lo siguiente:

$$y_i = x_i - m \ (i = 0, 1, \dots, N-1) \tag{4.15}$$

4.2.2.2 Operación por parejas balanceada

La operación por parejas balanceada se define de tal forma que la salida de dicha operación proporciona secuencias que no comparten correlación entre ellas.

Sean $x_i = \{x_0, x_1, \dots, x_{N-1}\}$ e $y_i = \{y_0, y_1, \dots, y_{N-1}\}$ dos secuencias de entrada unidimensionales compuestas por N muestras pares de valor entero y con media aritmética de valor cercano a cero, la operación por parejas balanceada produce dos secuencias de salida, $\chi_i = \{\chi_0, \chi_1, \dots, \chi_{N-1}\}$ y $\phi_i = \{\phi_0, \phi_1, \dots, \phi_{N-1}\}$, en la que cada una consiste en N muestras con una media aritmética aproximadamente cero.

Las secuencias de entrada x_i e y_i se corresponden con valor de las muestras a decorrelar. La secuencia de salida χ_i se corresponde con la componente principal de la transformada KLT de dos componentes descrita

en la Figura 4.1, mientras que la secuencia de salida ϕ_i se corresponde con la componente secundaria.

La Figura 4.5, proporciona una visión gráfica de esta operación.

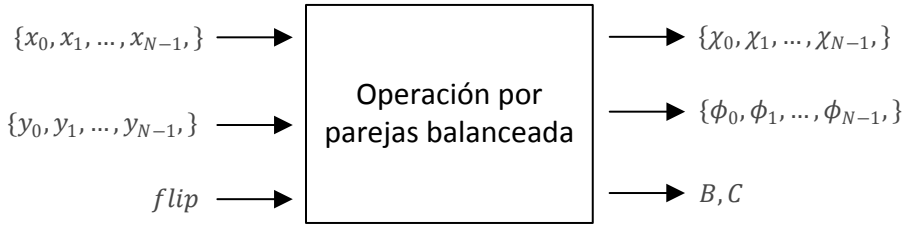


Figura 4.5 Operación por parejas balanceada

La ejecución de la operación por parejas balanceada se modifica adaptativamente en función del valor de las secuencias de entrada. Por lo tanto, teniendo en cuenta los valores de entrada, se calculan los parámetros de entrenamiento B y C. En el algoritmo, estos parámetros particularizan la operación por parejas balanceada. Además, estos parámetros son necesarios para invertir esta operación en el descompresor, por lo que son almacenados y transmitidos como información de control.

Adicionalmente, tal y como se observa en la Figura 4.5, la operación por parejas balanceada requiere de una variable de control de entrada denominada $flip$. Esta variable toma únicamente los valores uno o cero con el fin de modificar el signo de todos los elementos de las secuencias de salida $\{\chi_0, \chi_1, \dots, \chi_{N-1}\}$ y $\{\phi_0, \phi_1, \dots, \phi_{N-1},\}$.

Dentro de esta operación por parejas balanceada se hace uso de una constante denominada Ω. Su función es indicar la precisión de los resultados dentro de la operación por parejas balanceada. Puede tomar valores en el intervalo de números enteros [9, 16] y permanece constante dentro de todo el proceso de decorrelación de una imagen de entrada. Esta constante puede usarse para configurar la complejidad y prestaciones de la implementación del algoritmo. Esto es, cuanto mayor sea el valor de Ω (mayor precisión en el resultado de las operaciones), mejores prestaciones de codificación en términos de compresión, mientras que, por el contrario, para valores de Ω

menores se obtienen peores prestaciones de codificación (menor resolución en los resultados con lo que se introducen mayores errores en las operaciones).

Como requisito previo para el cálculo de los parámetros de entrenamiento B y C, se han de calcular los valores de varianzas y covarianzas de las secuencias de entrada x_i e y_i mediante las siguientes ecuaciones:

$$\widetilde{\sigma_x^2} = \sum_{i=0}^{N-1} x_i^2 - N \cdot \left[\frac{1}{N}\sum_{i=0}^{N-1} x_i\right]^2 \tag{4.16}$$

$$\widetilde{\sigma_y^2} = \sum_{i=0}^{N-1} y_i^2 - N \cdot \left[\frac{1}{N}\sum_{i=0}^{N-1} y_i\right]^2 \tag{4.17}$$

$$\widetilde{\sigma_{x,y}} = \sum_{i=0}^{N-1} (x_i \cdot y_i) - N \cdot \left[\frac{1}{N}\sum_{i=0}^{N-1} x_i\right] \cdot \left[\frac{1}{N}\sum_{i=0}^{N-1} y_i\right] \tag{4.18}$$

Los parámetros de entrenamiento B y C se calculan mediante las siguientes ecuaciones:

$$B = \begin{cases} \dfrac{\widetilde{\sigma_{x,y}}}{\left|\widetilde{\sigma_{x,y}}\right|}, & \widetilde{\sigma_{x,y}} \neq 0 \\ -1, & \widetilde{\sigma_{x,y}} = 0 \end{cases} \tag{4.19}$$

$$C = \begin{cases} \min\left(\max\left(\left[2^{\Omega-4} \cdot \dfrac{\widetilde{\sigma_x^2} - \widetilde{\sigma_y^2}}{2 \cdot \widetilde{\sigma_{x,y}}}\right], -2^{\Omega-2}\right), 2^{\Omega-2} - 1\right), & \widetilde{\sigma_{x,y}} \neq 0 \\ 2^{\Omega-2} - 1, & \widetilde{\sigma_{x,y}} = 0 \ y \ \widetilde{\sigma_x^2} \geq \widetilde{\sigma_y^2} \\ -2^{\Omega-2}, & \text{resto de los casos} \end{cases} \tag{4.20}$$

Una vez obtenidos los valores de B y C, se pueden obtener los valores decorrelados de las secuencias de salida a través de la red de pesos. Esta red requiere el cálculo de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$, además de dos variables. Una primera que indique la necesidad de modificar el signo del resultado (variable s), y una segunda que indique la necesidad de permuta condicional en los valores de las secuencias de salida $\{\chi_0, \chi_1, \ldots, \chi_{N-1}\}$ y $\{\phi_0, \phi_1, \ldots, \phi_{N-1},\}$. La necesidad de ambas variables se deriva de las dos soluciones posibles a la ecuación (4.11).

El resultado de la operación por parejas balanceada, es decir, los valores de la componente principal χ_i y la componente secundaria ϕ_i, se obtiene mediante la red de pesos que se presenta en la Figura 4.6. Como se observa en la Figura 4.6, los valores de las secuencias x_i e y_i son multiplicados en tres pasos por los valores calculados de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$. Asimismo, se aprecia que el signo del resultado puede ser condicionalmente modificado, así como que las secuencias de salida pueden ser condicionalmente permutadas.

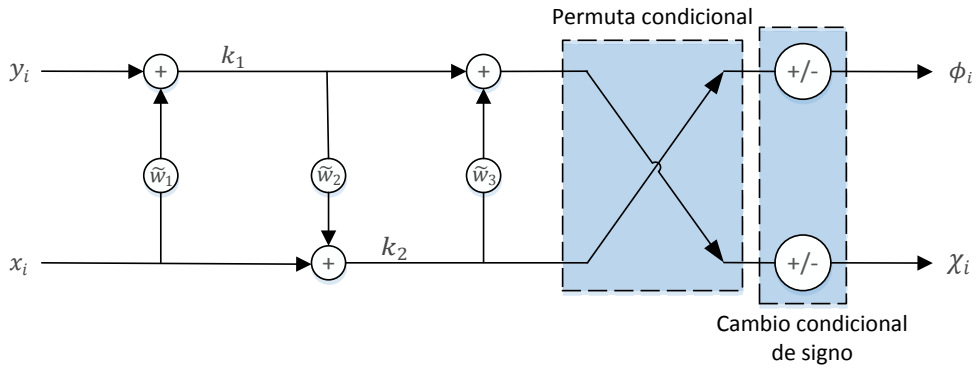


Figura 4.6 Red de pesos perteneciente a operación por parejas balanceada

Los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ se obtienen, a partir de B y C, mediante los siguientes pasos:

1) Sean $\tilde{t}$ y $\tilde{p}$ definidas como:

$$\tilde{t} = \begin{cases} 0, & \text{si } C = 2^{\Omega-2} - 1 \\ B \cdot 2^{\Omega-1}, & \text{si } C = -2^{\Omega-2} - 1 \\ B \cdot \left\lfloor \sqrt{2^{2\Omega-3} - \left\lfloor \dfrac{2^{2\Omega+1} \cdot C}{\lfloor \sqrt{2^8 \cdot C^2 + 2^{2\Omega}} + 0{,}5 \rfloor} \right\rfloor + 0{,}5} \right\rfloor, & \text{resto de los casos} \end{cases} \tag{4.21}$$

$$\tilde{p} = \left\lfloor \left(2^{\Omega+1}\right) \sqrt{1 - \left(\frac{\tilde{t}}{2^{\Omega-1}}\right)^2} + 0{,}5 \right\rfloor \tag{4.22}$$

2) Y, por otra parte, $\widetilde{\sqrt{2}}$ como:

$$\widetilde{\sqrt{2}} = \left\lfloor \left(2^{\Omega+2}\right)\sqrt{2} + 0{,}5 \right\rfloor \tag{4.23}$$

3) Los valores de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ se calculan mediante las siguientes ecuaciones en función del valor del parámetro de entrenamiento C:
   - Si $C < 0$:

$$\widetilde{w}_1 = \left\lfloor \left(2^{\Omega-4}\right)\frac{2\tilde{p} - \widetilde{\sqrt{2}}}{\tilde{t}} + 0{,}5 \right\rfloor \tag{4.24}$$

$$\widetilde{w}_2 = \left\lfloor \frac{\widetilde{\sqrt{2}} \cdot \tilde{t}}{2^{\Omega+3}} + 0{,}5 \right\rfloor \tag{4.25}$$

$$\widetilde{w}_3 = \left\lfloor \left(2^{\Omega-4}\right)\frac{4\tilde{p} - \widetilde{\sqrt{2}}}{\tilde{t}} + 0{,}5 \right\rfloor \tag{4.26}$$

     En este caso las secuencias de salida χ_i y ϕ_i no son permutadas.

   - Si $C \geq 0$, se han de permutar las secuencias de salida χ_i y ϕ_i, mientras que el cálculo de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ queda de la siguiente forma:

$$\widetilde{w}_1 = \left\lfloor \left(2^{\Omega-3}\right)\frac{\widetilde{\sqrt{2}} - 16\tilde{t}}{\tilde{p}} + 0{,}5 \right\rfloor \tag{4.27}$$

$$\widetilde{w}_2 = \left\lfloor -\frac{\widetilde{\sqrt{2}} \cdot \tilde{p}}{2^{\Omega+4}} + 0{,}5 \right\rfloor \tag{4.28}$$

$$\widetilde{w}_3 = \left\lfloor \left(2^{\Omega-3}\right)\frac{\widetilde{\sqrt{2}} - 8\tilde{t}}{\tilde{p}} + 0{,}5 \right\rfloor \tag{4.29}$$

La Tabla 4.1 resume el procedimiento descrito con anterioridad para cálculo de los pesos y la aplicación condición de permutación en función del parámetro de entrenamiento C.

Valor de C	Cálculo de los pesos	Permuta condicional
$C < 0$	$\widetilde{w}_1 = \left\lfloor (2^{\Omega-4}) \dfrac{2\tilde{p} - \widetilde{\sqrt{2}}}{\tilde{t}} + 0{,}5 \right\rfloor$ $\widetilde{w}_2 = \left\lfloor \dfrac{\widetilde{\sqrt{2}} \cdot \tilde{t}}{2^{\Omega+3}} + 0{,}5 \right\rfloor$ $\widetilde{w}_3 = \left\lfloor (2^{\Omega-4}) \dfrac{4\tilde{p} - \widetilde{\sqrt{2}}}{\tilde{t}} + 0{,}5 \right\rfloor$	No
$C \geq 0$	$\widetilde{w}_1 = \left\lfloor (2^{\Omega-3}) \dfrac{\widetilde{\sqrt{2}} - 16\tilde{t}}{\tilde{p}} + 0{,}5 \right\rfloor$ $\widetilde{w}_2 = \left\lfloor -\dfrac{\widetilde{\sqrt{2}} \cdot \tilde{p}}{2^{\Omega+4}} + 0{,}5 \right\rfloor$ $\widetilde{w}_3 = \left\lfloor (2^{\Omega-3}) \dfrac{\widetilde{\sqrt{2}} - 8\tilde{t}}{\tilde{p}} + 0{,}5 \right\rfloor$	Sí

Tabla 4.1 Operación por parejas balanceada: Cálculo de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ y aplicación de la permuta condicional en función del parámetro de entrenamiento C

El cambio de signo condicional de las secuencias de salida viene determinado por el valor del parámetro s. Su valor se calcula mediante la siguiente ecuación:

$$s = \begin{cases} 1, & \text{si } (C \geq 0 \text{ o } B \geq 0) \text{ y } flip = 0 \\ -1, & \text{si } (C \geq 0 \text{ o } B \geq 0) \text{ y } flip = 1 \\ -1, & \text{si } (C < 0 \text{ y } B < 0) \text{ y } flip = 0 \\ 1, & \text{si } (C < 0 \text{ y } B < 0) \text{ y } flip = 1 \end{cases} \tag{4.30}$$

El signo de los valores de la componente principal, la secuencia de salida χ_i, se ha de modificar cuando s toma el valor -1. Por el contrario, el signo de los valores de la componente secundaria, la secuencia de salida ϕ_i, se modifica cuando $C < 0$ y s toma el valor -1, o bien cuando $C \geq 0$ y s toma el valor 1.

4.2.2.3 Operación por parejas no-balanceada

En el caso de que la transformada POT se aplique sobre imágenes que contengan un número impar de bandas espectrales, Figura 4.2, es necesario modificar las expresiones de la operación por parejas balanceada, dando como resultado las expresiones que a continuación se presentan. Por lo tanto, la operación por parejas no-balanceada consiste en la aplicación de la operación por parejas balanceadas en la que se realizan dos variaciones:

1) Los parámetros de entrenamiento B y C se calculan de la siguiente forma:

$$B = \begin{cases} \dfrac{\widetilde{\sigma_{x,y}}}{\left|\widetilde{\sigma_{x,y}}\right|}, & \widetilde{\sigma_{x,y}} \neq 0 \\ -1, & \widetilde{\sigma_{x,y}} = 0 \end{cases} \tag{4.31}$$

$$C = \begin{cases} \min\left(\max\left(\left\lfloor 2^{\Omega-4}\dfrac{2\widetilde{\sigma_x^2} - \widetilde{\sigma_y^2}}{\left\lfloor\frac{1448}{1024}\cdot\left|\widetilde{\sigma_{x,y}}\right|\right\rfloor}\right\rfloor, -2^{\Omega-2}\right), 2^{\Omega-2}-1\right), & \widetilde{\sigma_{x,y}} \neq 0 \\ 2^{\Omega-2}-1, & \widetilde{\sigma_{x,y}} = 0 \text{ y } 2\widetilde{\sigma_x^2} \geq \widetilde{\sigma_y^2} \\ -2^{\Omega-2}, & \text{resto de los casos} \end{cases} \tag{4.32}$$

2) Los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ se calculan en función del parámetro de entrenamiento C de la siguiente forma:

   - Si $C < 0$, no se permutan las secuencias de salida y los pesos se calculan como:

$$\widetilde{w}_1 = \left\lfloor \left(2^{\Omega-3}\right)\frac{\left\lfloor\dfrac{\widetilde{\sqrt{2}}\cdot\tilde{p}}{2^{\Omega+2}} + 0{,}5\right\rfloor - 2^{\Omega+2}}{\tilde{t}} + 0{,}5 \right\rfloor \tag{4.33}$$

$$\widetilde{w}_2 = \left\lfloor \frac{\tilde{t}}{2} + 0{,}5 \right\rfloor \tag{4.34}$$

$$\widetilde{w}_3 = \left\lfloor \left(2^{\Omega-5}\right)\frac{\left\lfloor\dfrac{\widetilde{\sqrt{2}}\cdot\tilde{p}}{2^{\Omega-1}} + 0{,}5\right\rfloor - 2^{\Omega+4}}{\tilde{t}} + 0{,}5 \right\rfloor \tag{4.35}$$

- Si $C \geq 0$ los pesos se obtienen mediante las expresiones a continuación y las secuencias de salidas son permutadas.

$$\widetilde{w}_1 = \left\lfloor \left(2^{\Omega-1}\right) \frac{\left\lfloor \frac{\widetilde{\sqrt{2}} \cdot \left(2^{\Omega-2} - \tilde{t}\right)}{2^{\Omega}} + 0{,}5 \right\rfloor}{\tilde{p}} + 0{,}5 \right\rfloor \tag{4.36}$$

$$\widetilde{w}_2 = \left\lfloor -\frac{\widetilde{\sqrt{2}} \cdot \tilde{p}}{2^{\Omega+4}} + 0{,}5 \right\rfloor \tag{4.37}$$

$$\widetilde{w}_3 = \left\lfloor \left(2^{\Omega-1}\right) \frac{\left\lfloor \frac{\widetilde{\sqrt{2}} \cdot \left(2^{\Omega} - \tilde{t}\right)}{2^{\Omega+2}} + 0{,}5 \right\rfloor}{\tilde{p}} + 0{,}5 \right\rfloor \tag{4.38}$$

La Tabla 4.2 resume el procedimiento descrito para cálculo de los pesos y la aplicación condición de permutación en función del parámetro de entrenamiento C para la operación por parejas no-balanceada.

Valor de C	Cálculo de los pesos	Permuta condicional
$C < 0$	$\widetilde{w}_1 = \left\lfloor (2^{\Omega-3}) \frac{\left\lfloor \frac{\widetilde{\sqrt{2}} \cdot \tilde{p}}{2^{\Omega+2}} + 0{,}5 \right\rfloor - 2^{\Omega+2}}{\tilde{t}} + 0{,}5 \right\rfloor$ $\widetilde{w}_2 = \left\lfloor \frac{\tilde{t}}{2} + 0{,}5 \right\rfloor$ $\widetilde{w}_3 = \left\lfloor (2^{\Omega-5}) \frac{\left\lfloor \frac{\widetilde{\sqrt{2}} \cdot \tilde{p}}{2^{\Omega-1}} + 0{,}5 \right\rfloor - 2^{\Omega+4}}{\tilde{t}} + 0{,}5 \right\rfloor$	No

Valor de C	Cálculo de los pesos	Permuta condicional
$C \geq 0$	$\widetilde{w}_1 = \left\lfloor (2^{\Omega-1}) \dfrac{\left\lfloor \dfrac{\widetilde{\sqrt{2}} \cdot (2^{\Omega-2} - \tilde{t})}{2^{\Omega}} + 0{,}5 \right\rfloor}{\tilde{p}} + 0{,}5 \right\rfloor$ $\widetilde{w}_2 = \left\lfloor -\dfrac{\widetilde{\sqrt{2}} \cdot \tilde{p}}{2^{\Omega+4}} + 0{,}5 \right\rfloor$ $\widetilde{w}_3 = \left\lfloor (2^{\Omega-1}) \dfrac{\left\lfloor \dfrac{\widetilde{\sqrt{2}} \cdot (2^{\Omega} - \tilde{t})}{2^{\Omega+2}} + 0{,}5 \right\rfloor}{\tilde{p}} + 0{,}5 \right\rfloor$	Sí

Tabla 4.2 Operación por parejas no-balanceada: Cálculo de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ y aplicación de la permuta condicional en función del parámetro de entrenamiento C

4.3 Implementación de la aproximación entera de la transformada POT

En esta sección se describe la implementación hardware realizada teniendo como objetivo la síntesis de las operaciones de la aproximación entera de la transformada POT sobre la familia de FPGAs RTAX del fabricante Microsemi. La decisión de realizar la implementación sobre este tipo de FPGAs se encuentra motivada porque estos dispositivos se encuentran cualificados para operar en el entorno espacial y, además, poseen una amplia utilización en el sector y experiencia en vuelo. La aproximación entera de las operaciones pertenecientes a la transformada *Isorange* POT descrita en la sección anterior, cobra especial importancia dada las limitaciones en términos de potencia de cálculo y de recursos disponibles de este tipo de dispositivos si se comparan con FPGAs comerciales no cualificadas para espacio que pueden encontrarse actualmente en el mercado.

Dentro de la familia RTAX del fabricante Microsemi se han seleccionado los dispositivos de tipo FPGA cualificados para el espacio RTAX2000S y RTAX2000S-DSP para la síntesis de las operaciones de la aproximación entera de la transformada POT. Es destacable resaltar que la FPGA RTAX2000S-DSP contiene recursos dedicados adicionales para el procesado digital de señales (DSP) que permiten la implementación de operaciones aritméticas complejas de manera eficiente. En la Tabla 4.3 se presentan las principales características de ambas FPGAs.

	Dispositivo	
	RTAX2000S	**RTAX2000S-DSP**
Capacidad		
Puertas lógicas equivalentes	2.000.000	2.000.000
Puertas ASIC equivalentes	250.000	250.000
Módulos		
Registros	10.752	9.856
Celdas combinacionales	21.504	19.712
Almacenamiento		
Bloques RAM	64 (36kB)	64 (36kB)

	Dispositivo	
	RTAX2000S	**RTAX2000S-DSP**
Procesado Digital de Señales		
Bloques DSP	0	64
Distribución de reloj		
Hardware dedicado	4	4
Rutado	4	4
Número de Entradas/Salidas		
Bancos de Entrada/Salida	8	8
Entradas/Salidas disponibles para el usuario (máximo)	684	684

Tabla 4.3 Recursos disponibles en las FPGAs RTAX2000S y RTAX2000-DSP

La implementación de las operaciones pertenecientes a la aproximación entera de la transformada POT se realiza a nivel RTL mediante el empleo del lenguaje de descripción hardware VHDL. Tanto a nivel investigador como a nivel industrial, resulta fundamental confirmar la viabilidad de la transformada para ser desplegada a bordo y, por lo tanto, posibilitar su inclusión en la extensión del estándar CCSDS 122.0 [33].

En el sector espacial, la Agencia Espacial Europea (ESA) exige el cumplimiento del estándar ECSS-Q-ST-60-02C [95] para el desarrollo y verificación tanto de ASICs como FPGAs que se pretendan usar a bordo de un satélite. Por este motivo, se ha seguido el estándar ECSS-Q-ST-60-02C en términos de diseño y verificación de la implementación de las operaciones de la aproximación entera de la transformada POT con el fin de obtener resultados representativos de un desarrollo espacial tradicional. Se ha realizado la descripción a nivel RTL de las principales operaciones de la transformada POT: eliminación de la media, operación por parejas balanceada y operación por parejas no-balanceada, y la red de pesos. Estas etapas han sido optimizadas con el fin de alcanzar una baja ocupación de recursos hardware haciendo posible la síntesis de este diseño sobre los dispositivos cualificados para el espacio RTAX2000S y RTAX2000S-DSP.

Un aspecto relevante a la hora de la implementación de este algoritmo basado en la aproximación entera de la transformada POT es el hecho de que se han implementado mediante LUTs (del inglés, *LookUp Table*) las operaciones con mayor carga computacional. De esta forma se reduce la complejidad de implementación del algoritmo y, por consiguiente, su impacto a nivel de recursos hardware.

4.3.1 Consideraciones de diseño

Dada la alta carga computacional de la aproximación entera de la transformada POT, el principal objetivo del diseño a nivel RTL es la optimización a nivel de ocupación de recursos de la FPGA. Principalmente, los esfuerzos realizados se han centrado en la reducción de la complejidad de las operaciones, planificando secuencialmente las operaciones cuando ha existido la posibilidad.

4.3.1.1 Diseño modular

El diseño de las operaciones de la aproximación entera de la transformada POT se ha dividido en módulos funcionalmente independientes en los que cada uno contiene una operación y sus interfaces. Los módulos han sido descritos en el lenguaje de descripción hardware VHDL de acuerdo al estándar ECSS-Q-ST-60-02C. Este estándar, dentro sus requisitos, establece esta metodología de diseño con el fin de aumentar la granularidad del diseño y garantizar su completa verificación.

Se han diseñado en módulos independientes las siguientes operaciones: eliminación de la media, cálculo de la varianza y covarianza, el cálculo de los valores de los parámetros de entrenamiento B y C y la red de pesos. Una ventaja derivada de este tipo de implementación es la posibilidad de eliminar módulos en función de las necesidades reales de la implementación. Por ejemplo, si las secuencias de entradas presentan media nula, se puede realizar una implementación sin hacer uso de los módulos de eliminación de la media reduciendo los recursos necesarios para la implementación de la aproximación entera de la transformada POT.

4.3.1.2 Divisor serie

La implementación realizada de la división entera necesaria para el cálculo de los valores de la media aritmética, ecuación (4.14), y en el cálculo del parámetro de entrenamiento C, ecuación (4.20), se ha resuelto mediante un divisor serie en lugar de uno paralelo, con el fin de reducir la cantidad de recursos empleados. Las interfaces de este módulo se presentan en la Figura 4.7.

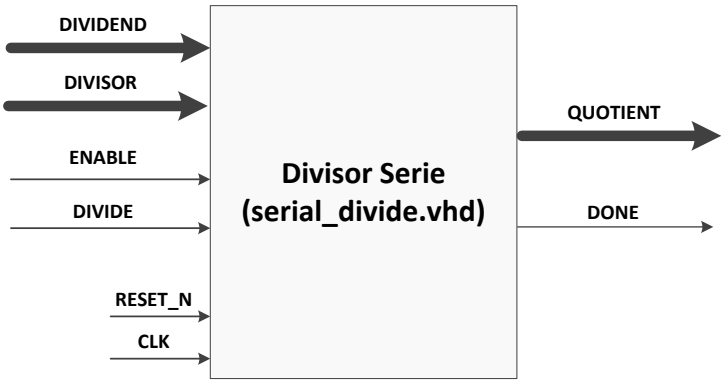


Figura 4.7 Divisor serie - interfaces

El divisor serie proporciona un bit del cociente de la división por ciclo de reloj. La operación de división comienza en el momento que se activa la señal de entrada *DIVIDE* durante un ciclo de reloj. El cociente, resultado de la división, se completa a la salida de este módulo (N + D) ciclos de reloj después de haber activado la señal *DIVIDE*, en donde N representa el tamaño en bits del numerador (señal *DIVIDEND*) y D, el tamaño en bits del denominador (señal *DIVISOR*). Se ha decidido a nivel de implementación que, si el valor del denominador es cero el resultado de la división será un cociente con todos sus bits fijados al valor lógico '1'. Por lo tanto, el divisor serie implementado proporciona a su salida una solución estable para cualquier valor de sus entradas.

4.3.1.3 Empleo de parámetros genéricos

Con el fin de proporcionar una descripción VHDL parametrizable, se han definido los siguientes parámetros como genéricos en el código VHDL. De

esta forma es posible realizar diferentes estimaciones de la ocupación en términos de recursos del diseño mediante la variación de estos parámetros.

1) Número de muestras de la secuencia de entrada (N): gBLK_SIZE
2) Número de bits de las muestras de entrada o bits por píxel (bpp): gPIXEL_WIDTH

Asimismo, se ha incluido un parámetro adicional que indica si el número de muestras de la secuencia de entrada (N) es un número potencia de dos o no, denominado gPO2_IMAGE. En el caso de que este parámetro sea activado, se evita la implementación de los divisores series empleados en el algoritmo. Estos divisores se implementan por medio de desplazadores lógicos a nivel de bit.

A modo de ejemplo, la Figura 4.8 presenta la definición de la entidad perteneciente al módulo de eliminación de la media a nivel VHDL. En ella se puede observar el impacto del uso de los parámetros genéricos cuando se realiza la descripción hardware.

```
entity mean is
    generic(
        gPO2_IMAGE      : std_logic := '0'; -- '1' number of pixels is power of 2
        gPIXEL_WIDTH    : integer := 16; -- width of each pixel
        gBLK_SIZE       : integer := 256  -- number of samples for a given region (N)
    );
    port(
        -- System signals
        FPGA_CLK     : in  std_logic;   -- System clock
        SYNC_RST     : in  std_logic;   -- Logic sync reset
        ENABLE       : in  std_logic;   -- enable module signal
        PIXEL_VALID  : in  std_logic;
        PIXEL        : in  signed(gPIXEL_WIDTH - 1 downto 0);
        MEAN         : out signed(gPIXEL_WIDTH - 1 downto 0);
        MEAN_VALID   : out std_logic
    );

end mean;
```

Figura 4.8 Declaración de la entidad del módulo de eliminación de la media

4.3.1.4 LUT para el cálculo de los pesos

De las ecuaciones descritas en la sección 4.2.2, se infiere que, pese a las ventajas proporcionadas por la aproximación entera de la transformada

POT, ésta continúa presentando una alta complejidad computacional. Particularmente, este es el caso del cálculo de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$. En general, una alta complejidad en las operaciones reduce las prestaciones de un algoritmo cuando se realiza su implementación en una FPGA cualificada para el espacio, como lo son tanto la RTAX2000S y la RTAX2000-DSP.

Una estrategia ampliamente empleada en las implementaciones hardware de algoritmos complejos es calcular a priori los valores de las operaciones con mayor complejidad computacional y almacenarlos en memoria. Por lo tanto, como estrategia de implementación, se ha optado por realizar el cálculo del valor de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ a priori y almacenarlos en una LUT en el interior de la FPGA. Esta LUT debe proporcionar los valores de los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ a su salida, a partir de los parámetros B, C, $flip$ y una variable que indique si la transformada es balanceada o no-balanceada. Asimismo, la LUT indica la necesidad de realizar la permuta y el cambio de signo de las secuencias de salida.

Se ha realizado una estimación de los recursos necesarios en el caso de que la implementación de la LUT se realizara mediante el empleo lógica combinacional. Los resultados obtenidos se presentan en la Tabla 4.4. Dado que las celdas combinacionales necesarias aumentan considerablemente cuando lo hace Ω (llegando a ocupar el 14% de una FPGA RTAX2000S para valores de $\Omega = 11$) en este trabajo se propone como solución alternativa el almacenamiento de estos valores en los bloques de memoria RAM (del inglés, *Random Access Memory*) interna disponible en la familia de FPGAs RTAX. Esto aumenta la cantidad de lógica disponible en la FPGA para la implementación del algoritmo. Por otra parte, es necesario resaltar que en una misión real estos valores pueden estar almacenados a bordo en memoria no-volátil tipo EEPROM (del inglés, *Electrically Erasable / Programmable Read-Only Memory*) externa a la FPGA. Esta solución resulta atractiva tanto a nivel de implementación del algoritmo como a nivel de planificación de misión. Además, los valores de la LUT pueden ser modificados desde la estación terrena mediante los procedimientos de actualización de parámetros realizados típicamente en las misiones espaciales (comandos *patch* y *dump*).

Ω	Celdas combinacionales
9	917
10	1685
11	3010
12	5446

Tabla 4.4 Ocupación a nivel de celdas combinacionales de la LUT

4.3.2 Descripción a nivel de transferencia de registros (RTL)

En esta sección se presenta la descripción a nivel de transferencia de registros de los módulos diseñados en la implementación de la transformada POT. A nivel general, el módulo diseñado recibe las secuencias de entrada y proporciona a su salida secuencias decorreladas siguiendo las expresiones presentadas en la sección 4.2.2.

La Figura 4.9 presenta las interfaces que componen el diseño. La descripción detallada de cada señal perteneciente a la interfaz se presenta en la Tabla 4.5.

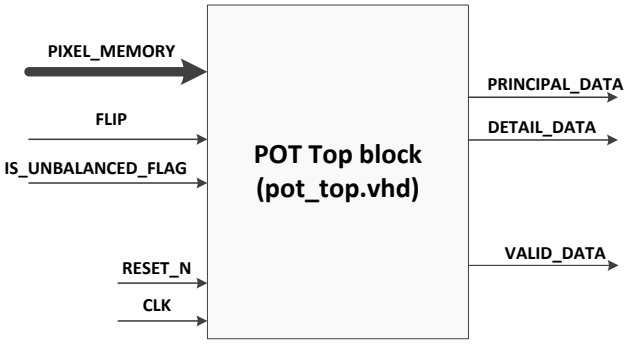


Figura 4.9 Interfaces del módulo que implementa la transformada POT

Nombre	Ancho	Tipo
clk	**1 bit**	**Entrada**
Reloj del sistema activo a nivel alto.		
rst	**1 bit**	**Entrada**
Reset asíncrono activo a nivel alto.		
flip	**1 bit**	**Entrada**
Señal de entrada empleada en el cambio condicional de signo.		
is_unbalanced_flag	**1 bit**	**Entrada**
Señal que indica si la operación por parejas es balanceada, '0' lógico, o no-balanceada, '1' lógico.		
pixel_memory	**(gBLK_SIZE x gPIXEL_WIDTH) bits**	**RAM doble puerto**
Interfaz a memoria RAM de doble puerto que contiene los píxeles de la secuencia de entrada.		
principal_data	**(gPIXEL_WIDTH + 3) bits**	**Salida**
Secuencia de valores asociados a la componente principal. Su rango dinámico es 3 bits mayor que el ancho en bits de los píxeles de entrada.		
detail_data	**(gPIXEL_WIDTH + 3) bits**	**Salida**
Secuencia de valores asociados a la componente secundaria. Su rango dinámico es 3 bits mayor que el ancho en bits de los píxeles de entrada.		
valid_data	**1 bit**	**Salida**
Señal activa a nivel alto que indica la validez de un nuevo valor de las secuencias de salida principa_data y detail_data. Su duración es de un ciclo de la señal de reloj para cada valor de ambas secuencias.		

Tabla 4.5 Descripción de las interfaces del módulo que implementa la transformada POT

Internamente el diseño presenta el diagrama de bloques funcionales de la Figura 4.10. Se encuentra compuesto por tres bloques principales: eliminación de la media y cálculo de los parámetros de entrenamiento, LUT que proporciona el valor de los pesos y la red de pesos.

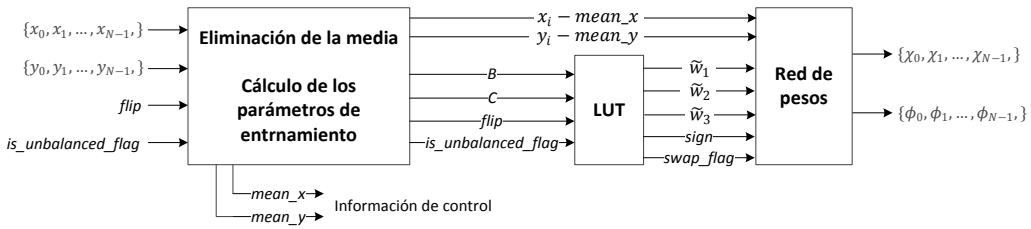


Figura 4.10 Diagrama de bloques funcional de las operaciones implementadas de la transformada POT

4.3.2.1 Arquitectura interna

El diagrama de bloques a nivel funcional presentado en la Figura 4.10, se ha de descomponer en módulos funcionalmente independientes con el fin realizar una implementación de acuerdo al estándar ECSS-Q-ST-60-02C. Estos módulos se implementan y verifican posteriormente mediante el lenguaje de descripción hardware VHDL.

La Figura 4.11 muestra la arquitectura interna de los módulos diseñados. Se compone de los siguientes módulos:

- **Controlador de memoria** (mem_ctrl.vhd): Controlador de la memoria de entrada. Se ha implementado un controlador de memoria que además actúa de árbitro en el acceso. Además, se aprovecha el hecho de tener disponibles los valores de los píxeles de entrada para realizar el cálculo de la media. De esta forma se realiza el cálculo de la media de forma secuencial: se suma en cada ciclo de reloj un nuevo valor y se acumula en un registro interno para cuando se llegue al final de bloque de píxeles, definido en el valor de gBLK_SIZE, realizar la división (o desplazamiento lógico) para obtener el valor de la media.
- **Eliminación de la media** (mean_sub.vhd): Este módulo implementa la ecuación (4.15) de tal forma que en cada ciclo de reloj se realiza la lectura de un valor de la secuencia de entrada y se realiza la resta de dicho valor y la media. Dado que los valores han de estar disponibles para varios módulos en instantes posteriores, son almacenados en registros internos

formando una memoria de tipo FIFO (del inglés, *First-In-First-Out*). Este módulo se instancia dos veces en el diseño ya que es necesario eliminar la media de ambas secuencias de entrada. Además, esta instanciación en paralelo permite aumentar la tasa de datos interna de la implementación.

- **Cálculo de la varianza** (sigma.vhd): Este módulo implementa las operaciones descritas en la ecuaciones (4.16) y (4.17). Para su cálculo el módulo se apoya en los módulos square_sum.vhd y mean.vhd. El primero, realiza el cálculo de la suma de los cuadrados de los valores de entrada mientras que, el segundo, calcula la media de dichos valores El diseño emplea dos instancias del módulo, uno para cada secuencia de entrada.
- **Cálculo de la covarianza** (sigma_bxy.vhd): En el mismo se implementa la ecuación (4.18). Existe una dependencia de datos entre este módulo y los que implementan el cálculo de la varianza de las muestras, hecho que penaliza la prestación de la implementación en términos de tasa de bits. Esta dependencia se debe al cálculo de la media de los valores de la secuencia de entrada que se ha decidido no replicar en el interior de este módulo ya que se emplean varios divisores que suponen un aumento en el área.
- **Cálculo del parámetro de entrenamiento B** (b_calc.vhd): Este módulo proporciona el valor del parámetro de entrenamiento B analizando únicamente el signo del valor de la covarianza proporcionada a la salida del módulo sigma_bxy.vhd. Si el signo de la covarianza es positivo, el valor del parámetro de entrenamiento B es 1, en caso contrario, B es igual a -1.
- **Cálculo del parámetro de entrenamiento C** (c_calc.vhd y c_calc_unbalanced.vhd): El cálculo del valor del parámetro de entrenamiento C se realiza en el interior de estos módulos. Para el caso de la operación por parejas balanceada se emplea el módulo c_calc.vhd en donde se implementa la ecuación (4.20). Por otra parte, en el caso de la operación por parejas

no-balanceada se usa el módulo c_calc_unbalanced.vhd implementándose la ecuación (4.32).

- **LUT de pesos** (weights_LUT.vhd): Como se ha comentado, para el cálculo del valor de los pesos se emplea una LUT. La LUT proporciona a su salida los valores de los pesos en función del valor de las entradas simplificando de esta forma la implementación de las ecuaciones descritas en la Tabla 4.1 y la Tabla 4.2 al acceso a una memoria RAM interna.
- **Red de pesos** (lifting_network.vhd): En el interior de este módulo se implementa la red de pesos descrita en la Figura 4.6. El módulo presenta una fuerte dependencia de los datos de entrada calculados a lo largo del algoritmo, por este motivo la implementación se ha realizado empleando señales de validación de datos para cada una de las entradas al módulo. De esta forma se garantiza la correcta temporización de la implementación, es decir, se asegura que las operaciones se realizan con datos validados.

Con el fin de poder invertir las operaciones de la transformada POT en el descompresor, es necesario que, además de las secuencias de salida (principal y detalle), se transmita la información de control asociada a dichas operaciones. Esta información de control consta de los siguientes parámetros:

- El valor de Ω empleado en la estructura multinivel de la transformada.
- El valor de la señal $flip$ empleado.
- Para cada uno de los niveles:
  - Los valores de la media m de cada secuencia de entrada del algoritmo.
  - Los valores de los parámetros de entrenamiento B y C calculados en cada operación por parejas.

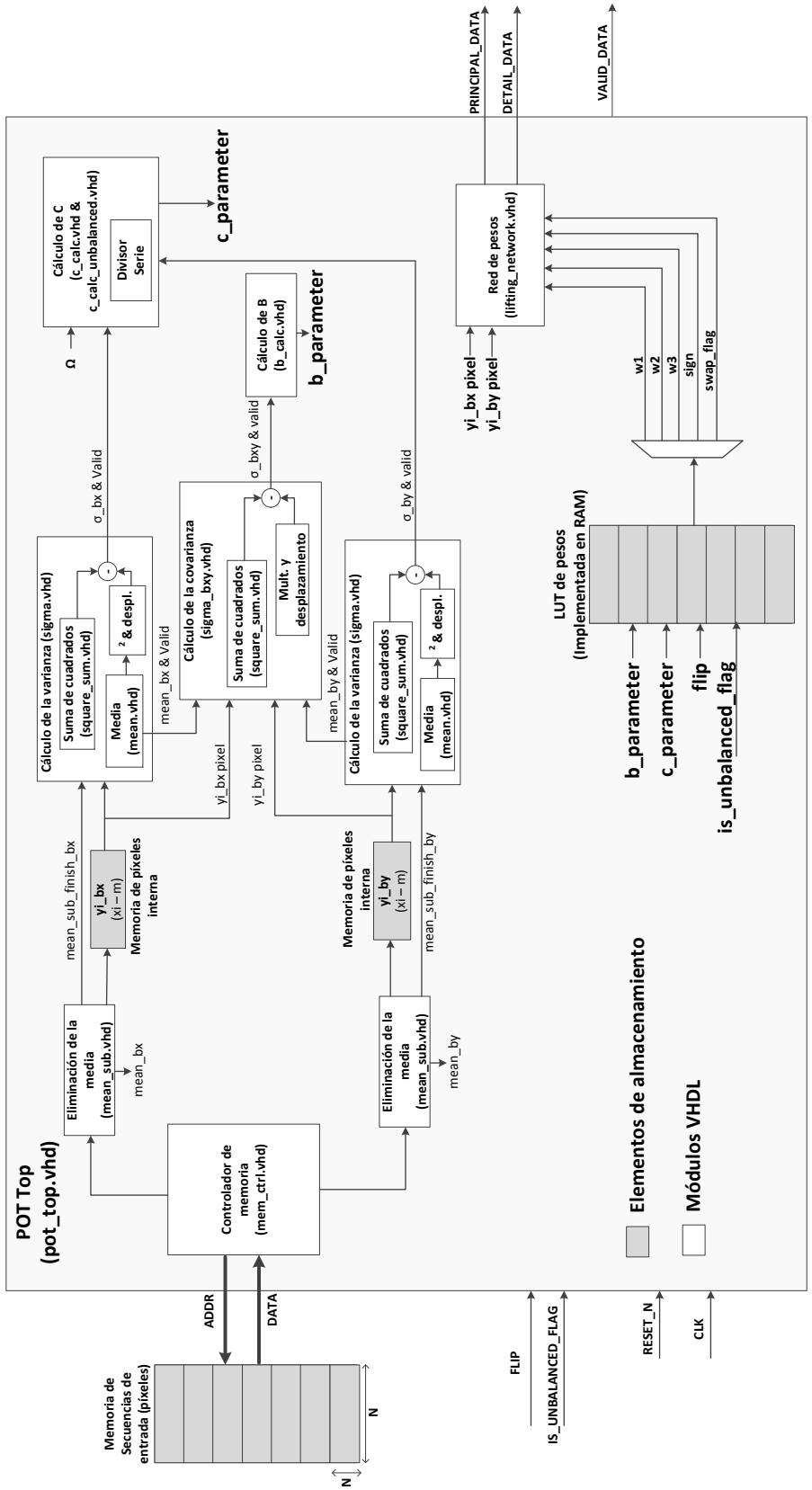


Figura 4.11 Arquitectura interna del diseño implementado

A nivel de flujo de datos el algoritmo procesa las muestras de entrada aplicando secuencialmente los siguientes pasos:

1. En primer lugar, se calcula y elimina la media aritmética de las muestras de entrada, calculándose la varianza y covarianza de dichas muestras. En este paso, se obtienen los valores de los parámetros de entrenamiento B y C.
2. A continuación, una vez calculados los parámetros de entrenamiento, es posible obtener los pesos $\widetilde{w}_1$, $\widetilde{w}_2$ y $\widetilde{w}_3$ mediante el empleo de la LUT. Como se describe en la sección 4.3.1, en el diseño se considera que los valores de la LUT se almacenan en una memoria EEPROM externa y son cargados en la RAM interna de la FPGA en la inicialización del sistema. Los bloques que componen la memoria RAM interna se encuentran protegidos ante posibles errores causados por la radiación existente en el entorno espacial mediante EDAC (del inglés, *Error Detection And Correction*), siendo posible la corrección de un error y la detección de dos errores en cada dato almacenado.
3. Finalmente, los valores de las secuencias decorreladas se obtienen a partir de los pesos mediante la red de pesos.

La Figura 4.12 muestra la planificación del algoritmo diseñada y el flujo de datos entre los diferentes módulos que componen la implementación de la aproximación entera de la transformada POT. Las dependencias de datos se muestran mediante las flechas entre módulos. Este diagrama muestra la ejecución paralela de aquellos módulos que no presentan dependencias de datos. Además, puede observarse que se adelanta la ejecución de aquellas partes dentro de los propios módulos que no presentan dependencias de datos con otros módulos. Este es el caso del módulo que implementa el cálculo de la covarianza en donde se adelanta el cálculo de la suma de cuadrados, ecuación (4.18), y se espera por las medias calculadas en los módulos que computan la varianza de la secuencia de entrada x y de la secuencia y para finalizar la ejecución del módulo.

La duración en ciclos de reloj de la ejecución de cada módulo se representa entre paréntesis. La duración depende del valor de los parámetros genéricos gBLK_SIZE y gPIXEL_WIDTH que a su vez dependen de las características de la imagen hiperespectral de entrada. Es importante señalar que, las duraciones descritas en la Figura 4.12 corresponden a la implementación realizada mediante divisores serie sin dezplazadores lógicos (gPO2_IMAGE = '0'), es decir, representan el peor caso en términos de ejecución en ciclos de reloj de cada módulo. Si se remplazan los divisores serie por dezplazadores lógicos (gPO2_IMAGE = '1') las duraciones descritas entre paréntesis dependientes del parámetro gPIXEL_WIDTH se eliminan, dando lugar a una ejecución de menor duración.

Como se describe en este capítulo, la aproximación entera de la transformada POT se encuentra compuesta por una estructura multinivel en la que en cada nivel se emplea una transformada KLT, por lo tanto, resulta interesante estudiar el posible paralelismo de ejecuciones sucesivas de una misma implementación de la transformada. Esta relación se refleja en la Figura 4.12, mediante la dependencia de datos representada por la línea punteada cuyo origen se encuentra a la salida del módulo controlador de memoria y cálculo de la media, de duración gBLK_SIZE + gPIXEL_WIDTH ciclos de reloj, de la transformada POT en su ejecución (n) y finaliza en el módulo de la transformada POT en su ejecución (n+1). De este modo, una vez finalizado el cálculo de la media puede comenzarse la lectura de los datos de las siguientes secuencias, es decir, no es necesario esperar a terminar una ejecución completa de la implementación de la transformada POT para comenzar la siguiente ejecución, de este modo, se incrementa la tasa de datos procesada por el algoritmo.

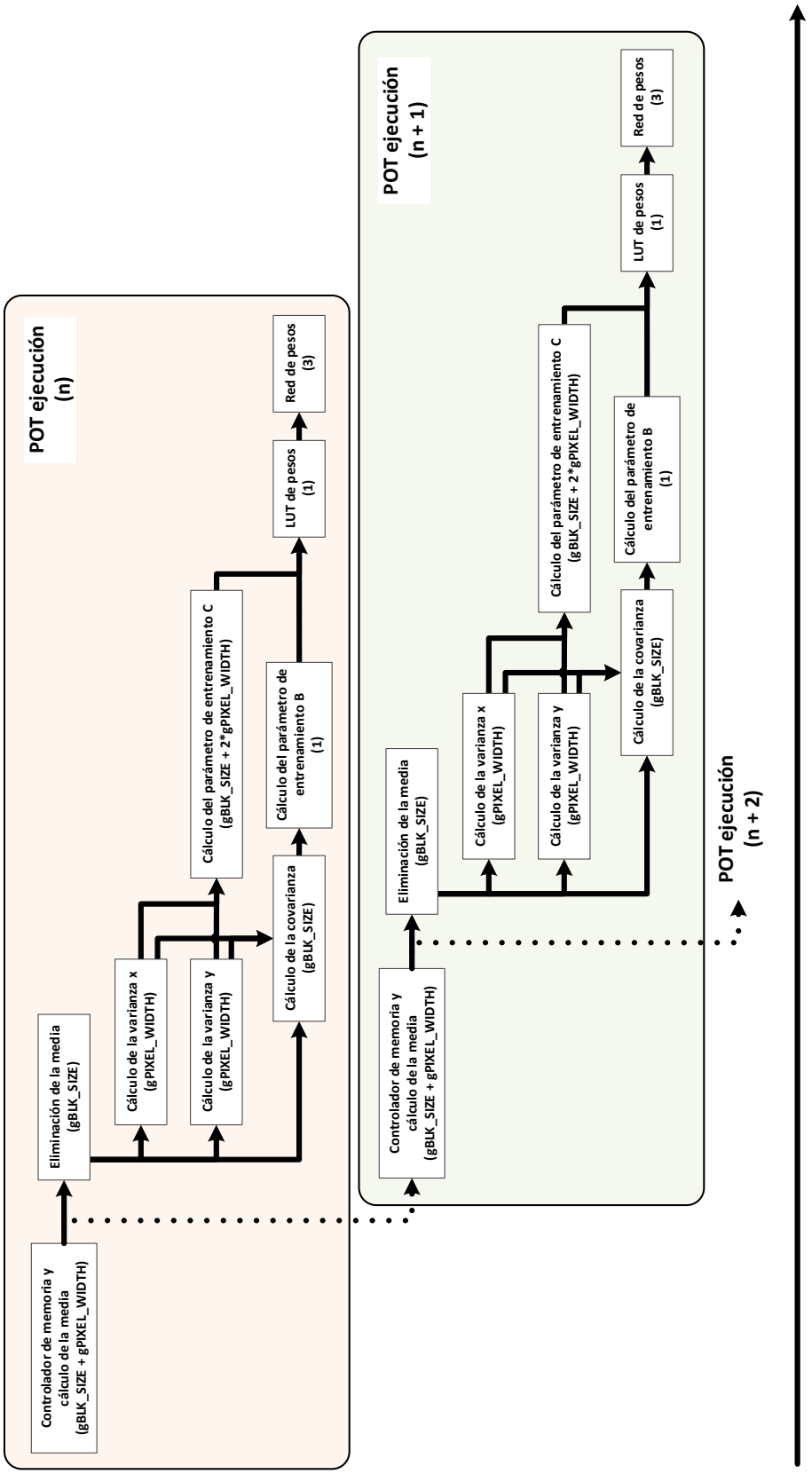


Figura 4.12 Planificación y dependencias de datos de la implementación

4.3.3 Verificación funcional

El proceso de verificación tiene como objetivo detectar los posibles errores de diseño que puedan ser originados en cada una de las etapas que atraviesa un sistema durante su concepción y desarrollo.

Durante la verificación funcional se toma como referencia la especificación previamente realizada para el diseño y se compara con la implementación que se desea validar, esperando que ambas describan exactamente el mismo sistema. En el caso de la implementación de la aproximación entera de la transformada POT, para hacer la verificación funcional se emplearán las ecuaciones presentadas a lo largo de este capítulo como especificación. Se han implementado las ecuaciones en el entorno de pruebas mediante código no sintetizable empleando operaciones en punto flotante y realizando redondeos, de este modo se acelera el proceso creación del entorno y se dispone de una referencia para realizar la verificación.

Como se ha comentado en este capítulo, los módulos a verificar han sido descritos en el lenguaje de descripción hardware VHDL de acuerdo al estándar ECSS-Q-ST-60-02C. Este estándar establece una metodología de verificación con el fin de garantizar que el diseño cumple con los requisitos definidos. En la metodología se definen dos niveles de verificación: verificación a nivel de módulo y verificación a nivel de sistema. Durante la verificación a nivel de módulo se ha de comprobar que el módulo diseñado cumple los requisitos definidos para el mismo y que además se estimulan todas las líneas de código proporcionando una cobertura del 100%, justificándose las posibles desviaciones. Por otra parte, la verificación a nivel sistema compara el comportamiento de todo el diseño con respecto a los requisitos de alto nivel establecidos.

Por lo tanto, la verificación funcional de la aproximación entera de la transformada POT se ha realizado en dos niveles:

1. **Nivel módulo:** Se ha verificado mediante simulación con la herramienta ModelSim el correcto funcionamiento de cada

módulo diseñado. Para ello, se ha hecho uso de la descripción entera de las ecuaciones presentadas en la sección 2.2.1. Estas ecuaciones se han implementado en cada uno de los testbench realizados para cada módulo comparando ciclo a ciclo de reloj la salida del DUV con el resultado esperado. Por otra parte, se ha iterado en la realización de la verificación funcional hasta conseguir una cobertura de código del 100% a nivel de sentencias.

2. **Nivel sistema:** Se verifica, haciendo uso de la herramienta ModelSim, que tras la unión de los módulos diseñados y verificados no existen errores en la planificación del algoritmo en su totalidad. Además, mediante esta verificación funcional se han comprobado las dependencias de datos existentes en el algoritmo.

4.4 Resultados obtenidos

La síntesis del módulo diseñado ha sido realizada sobre las FPGAs RTAX2000S y RTAX2000S-DSP del fabricante Microsemi. Éstas FPGAs están cualificadas para operar en el espacio y tienen una amplia experiencia de uso en vuelo. La principal diferencia entre ambos dispositivos es que la FPGA RTAX2000S-DSP contiene bloques dedicados para el procesado digital de la señal, DSP. Esto posibilita la implementación de operaciones en estos bloques DSP liberando la ocupación en términos de recursos combinacionales de la implementación sobre la FPGA.

Se ha realizado la síntesis de diferentes configuraciones del algoritmo en las que se han variado los parámetros genéricos, es decir, se han establecido diferentes valores para el número de muestras de entrada (N) y el número de bits por pixel (bpp). Además, se ha variado entre implementaciones sobre imágenes con muestras de entrada múltiplos de dos, llamadas PO2, y no múltiplos de dos. En todos los casos se ha definido un valor $\Omega = 16$ al ser el mayor valor posible dentro del rango, es decir, el valor que dará lugar a mayores ocupaciones de recursos, representando así el peor caso. En la Figura 4.13 se presenta el árbol de configuración de parámetros tenido en cuenta así como las configuraciones resultantes. Estas configuraciones proporcionan las combinaciones denominadas C1, C2, C3, C4, C5, C6 y C8 objeto de síntesis.

Nótese que, a pesar de que el número de muestras de entrada (N) sea potencia de dos, se considerará una implementación no-potencia de dos si no se establece explícitamente este hecho activando el parámetro PO2, esto es, siempre que PO2 esté desactivado, se inferirán los divisores serie de números enteros, incluso si el divisor es potencia de dos. Este es el caso que se presenta en las configuraciones definidas en el rango de C5 a C8.

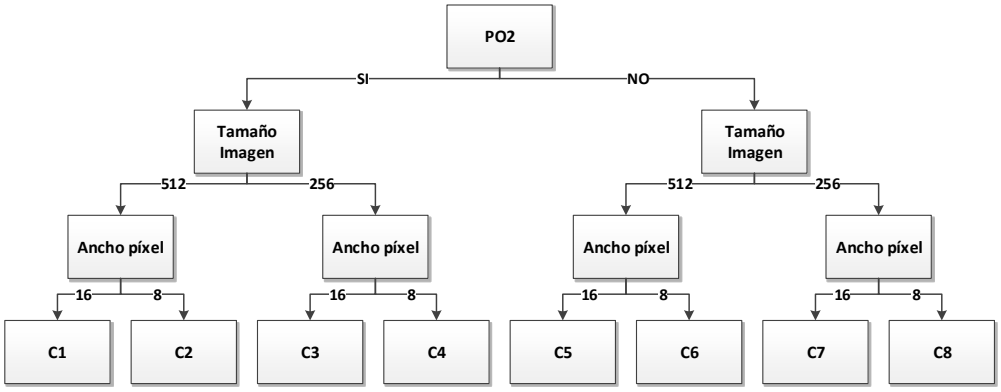


Figura 4.13 Árbol de configuración de parámetros y configuraciones asociadas

La Tabla 4.6 presenta la ocupación de recursos hardware y la máxima frecuencia de operación de los módulos implementados para las diferentes configuraciones sobre la FPGA RTAX2000S. La ocupación de recursos hardware se presenta en términos de número de celdas combinacionales empleadas, número de registros y bloques de memoria RAM usados. También se muestran los valores de la frecuencia máxima de síntesis alcanzada en cada configuración.

Config.	PO2	N	bpp	Comb. (%)	Sec. (%)	BRAM (%)	Frecuencia (MHz)
C1	Y	512	16	64.82	24.96	48.75	29.92
C2	Y	512	8	25.25	13.67	28.13	37.00
C3	Y	256	16	63.84	24.48	45.00	30.92
C4	Y	256	8	24.13	13.05	28.13	37.58
C5	N	512	16	73.04	29.65	48.75	29.92
C6	N	512	8	30.81	17.01	28.13	37.00
C7	N	256	16	70.60	29.14	45.00	36.08
C8	N	256	8	29.22	15.98	28.13	37.58

Tabla 4.6 Resultados de síntesis sobre la FPGA RTAX2000S para las diferentes configuraciones

De la misma forma, la Tabla 4.7 proporciona los resultados de ocupación y máxima frecuencia de operación de la síntesis sobre la FPGA RTAX2000S-DSP. En este caso, la FPGA dispone de bloques DSP con lo que la ocupación se proporciona en dicha tabla.

Config.	PO2	N	bpp	Comb. (%)	Sec. (%)	BRAM (%)	Bloques DSP (%)	Frecuencia (MHz)
C1	Y	512	16	21.73	18.48	48.75	33.75	53.83
C2	Y	512	8	14.07	12.74	28.13	22.50	72.00
C3	Y	256	16	21.07	18.01	45.00	33.75	54.67
C4	Y	256	8	13.51	12.07	28.13	22.50	73.50
C5	N	512	16	28.69	22.71	48.75	33.75	53.83
C6	N	512	8	18.80	15.46	28.13	22.50	72.00
C7	N	256	16	27.80	22.05	45.00	33.75	54.67
C8	N	256	8	17.89	14.78	28.13	22.50	73.50

Tabla 4.7 Resultados de síntesis sobre la FPGA RTAX2000S-DSP para las diferentes configuraciones

Con el fin de facilitar la comparativa de los resultados obtenidos, se presentan los valores de la Tabla 4.6 y de la Tabla 4.7, de forma gráfica en la Figura 4.14 y la Figura 4.15, donde se observa que la totalidad de los módulos diseñados pueden ser sintetizados en una FPGA RTAX2000S para cualquiera de las configuraciones de las definidas. Si se comparan los resultados de síntesis obtenidos para ambas FPGAs, se aprecia que la ocupación en términos de celdas combinacionales se reduce en la FPGA RTAX2000S-DSP. Este hecho se produce gracias al empleo de los bloques DSP disponibles en este dispositivo para las operaciones que componen el algoritmo. Por lo tanto, estas operaciones que se implementan en el caso de la FPGA RTAX2000S mediante lógica combinacional se implementan en la FPGA RTAX2000S-DSP mediante los bloques DSP disponibles.

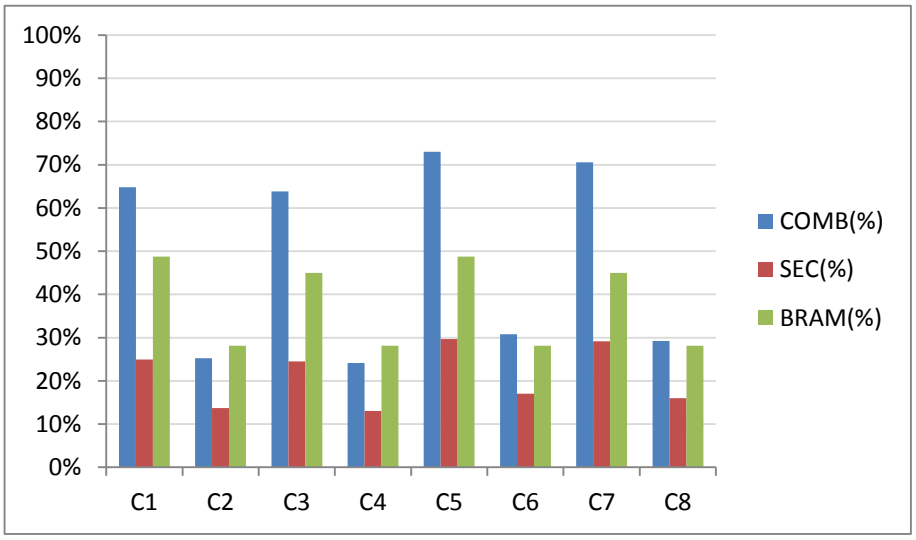


Figura 4.14 Ocupación de las diferentes configuraciones en la FPGA RTAX2000S

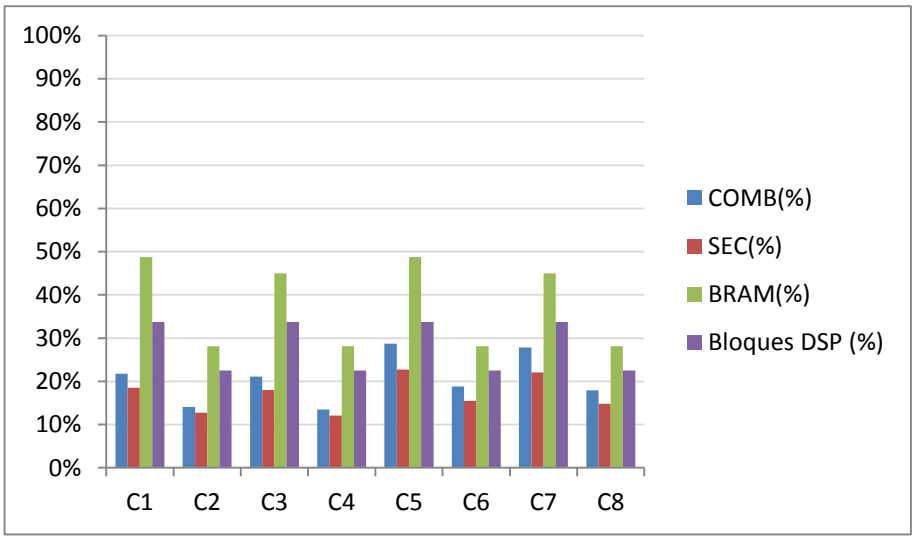


Figura 4.15 Ocupación de las diferentes configuraciones en la FPGA RTAX2000S-DSP

Asimismo, de la Figura 4.14 y la Figura 4.15, se extrae el aumento en términos de ocupación de recursos cuando las muestras de entrada se representan mediante 16 bpp. Sin embargo, no existe un aumento significativo en el número de recursos utilizados cuando se modifica el número de muestras de entrada (N) de 256 a 512.

La máxima frecuencia obtenida de las implementaciones se presenta y compara en la Figura 4.16. Estos datos permiten ver la diferencia entre los resultados de síntesis de las FPGAs RTAX2000S y RTAX2000S-DSP. A nivel

general los valores de frecuencia máxima obtenidos en las implementaciones sobre la RTAX2000S-DSP son claramente superiores. Esto se debe a que el camino crítico, aquel comprendido entre dos registros consecutivos haciendo uso de lógica combinacional, se simplifica gracias a los bloques DSP disponibles en esta FPGA. La mínima frecuencia obtenida en el caso de la RTAX2000S es de 29,92 MHz, mientras que la máxima frecuencia es de 37,58 MHz. En el caso de la FPGA RTAX2000S-DSP se obtiene una frecuencia máxima de operación de 73,50 MHz y una frecuencia mínima de operación de 53,83 MHz. Nótese que los valores de frecuencia máxima de operación decrecen a medida que el ancho en bits por píxel aumenta. Por el contrario, no se observan diferencias significativas en términos de frecuencia máxima de operación causadas por un aumento en el número de muestras de entradas a procesar por el algoritmo.

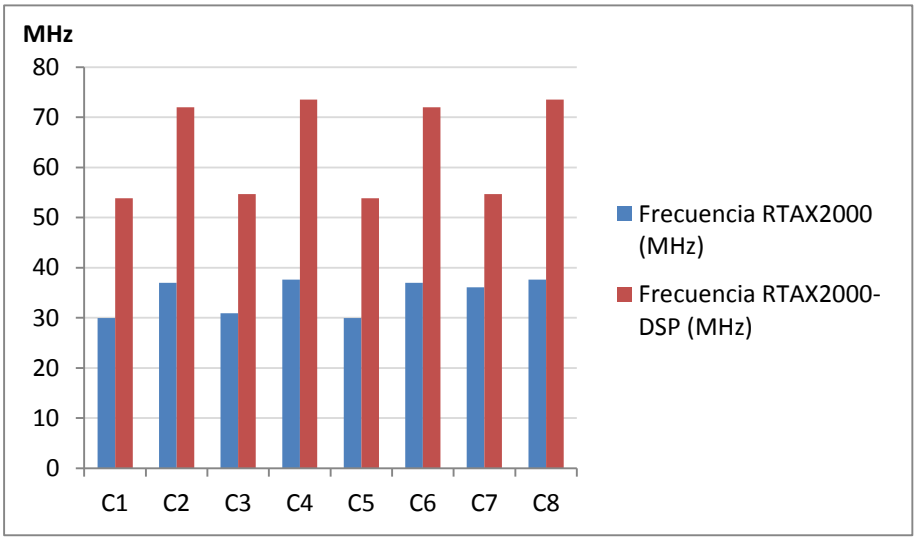


Figura 4.16 Frecuencia máxima de operación para las diferentes configuraciones sobre las FPGAs: RTAX2000S y RTAX2000S-DSP

Una vez presentados los resultados obtenidos para la implementación de la transformada POT en cada una de las FPGAs, resulta interesante conocer los módulos que presentan un mayor impacto en la ocupación de recursos. En este sentido, la Figura 4.17 y la Figura 4.18 muestran la ocupación porcentual de recursos en la FPGA RTAX2000S de cada módulo sobre el total de recursos empleado para las configuraciones C1 y C5. Estas configuraciones sirven como referencia ya que, en el resto de configuraciones, se han observado distribuciones de ocupación de recursos similares a las presentadas.

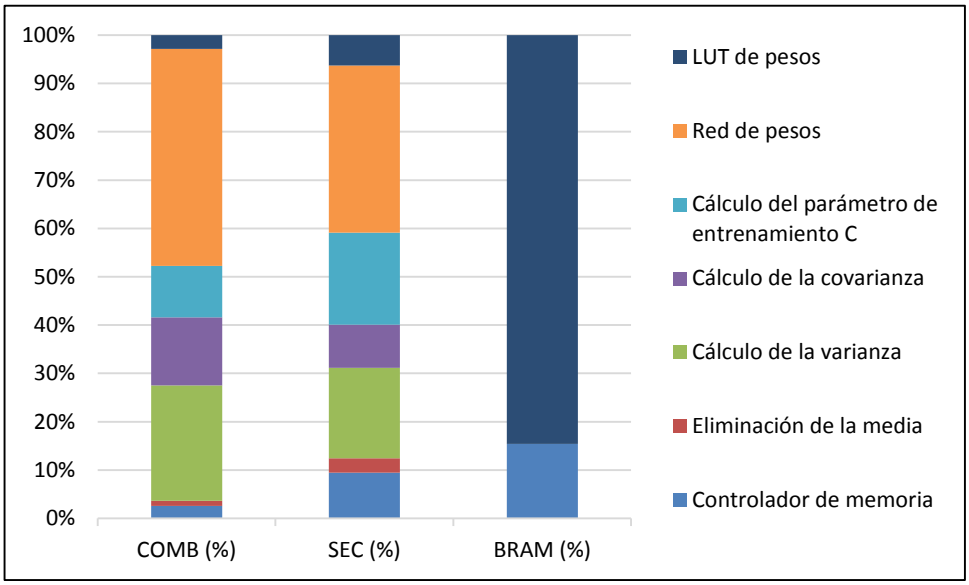


Figura 4.17 Configuración C1: Porcentaje de ocupación relativa de cada módulo sobre la FPGA RTAX2000S

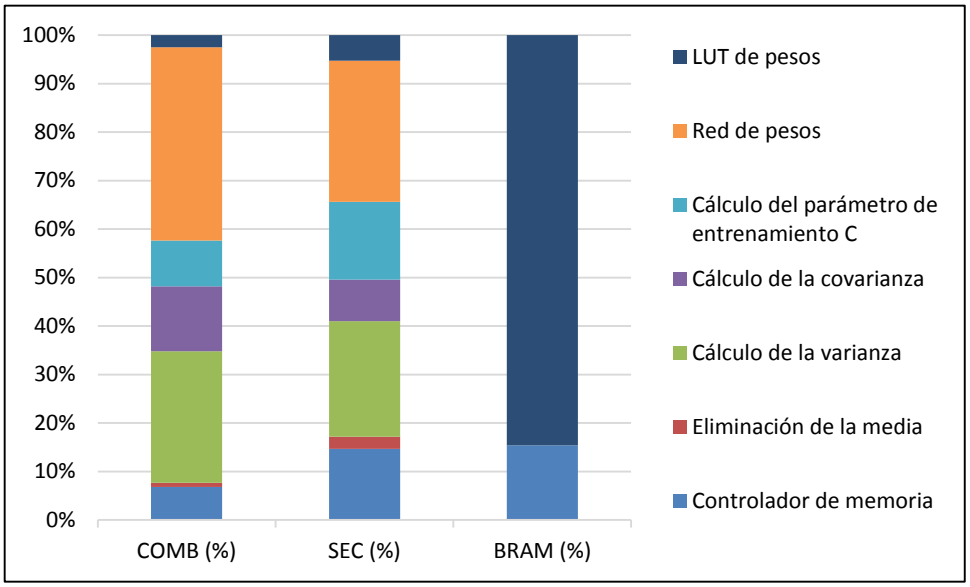


Figura 4.18 Configuración C5: Porcentaje de ocupación relativa de cada módulo sobre la FPGA RTAX2000S

De la Figura 4.17 y la Figura 4.18, se concluye que el módulo que contribuye en mayor medida en la ocupación de recursos tanto combinacionales como secuenciales es la red de pesos. Este hecho viene motivado por la necesidad del empleo de multiplicadores para su implementación. En cuanto a bloques de memoria RAM empleados, se observa

que la LUT de pesos representa más del 80 % de las necesidades de almacenamiento del algoritmo.

De manera análoga, sobre los resultados obtenidos para la FPGA RTAX2000-DSP, se ha analizado el porcentaje de ocupación de cada módulo sobre el total de recursos empleados. De las diferentes configuraciones implementadas, se ha escogido la configuración C1 y C5, Figura 4.19 y Figura 4.20, como referencia ya que, se han observado resultados similares para el resto de configuraciones.

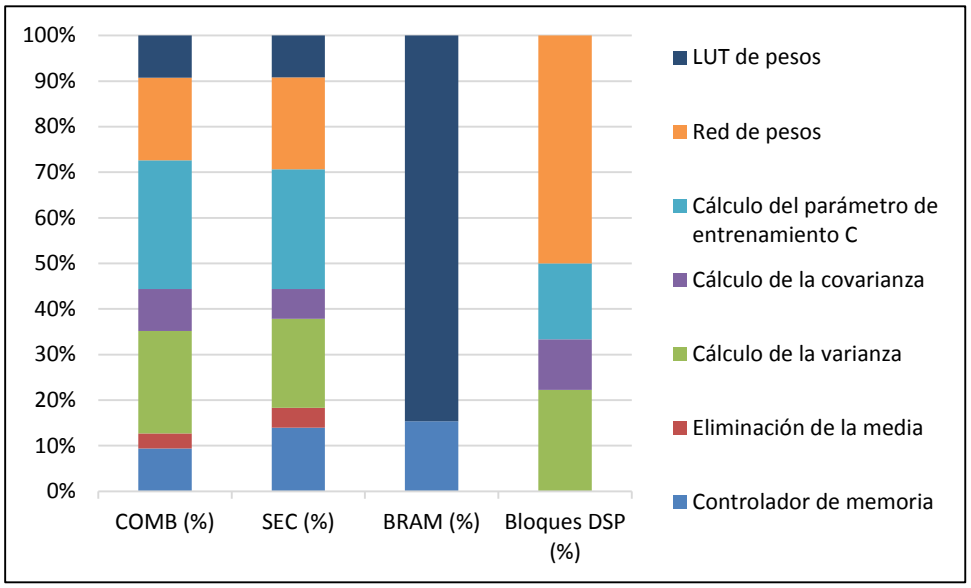


Figura 4.19 Configuración C1: Porcentaje de ocupación relativa de cada módulo sobre la FPGA RTAX2000S-DSP

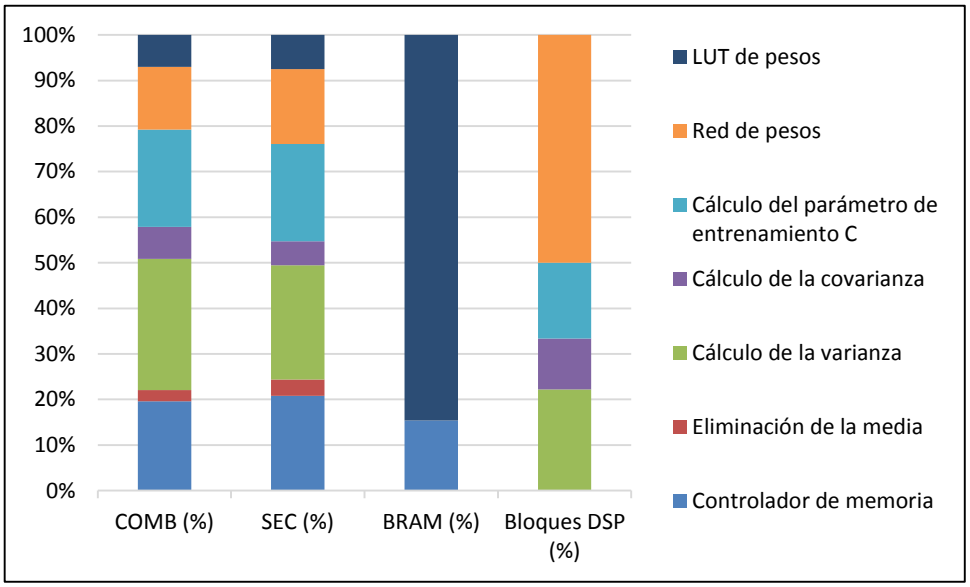


Figura 4.20 Configuración C5: Porcentaje de ocupación relativa de cada módulo sobre la FPGA RTAX2000S-DSP

En el caso de la FPGA RTAX2000S-DSP, se observa que los módulo que presenta una mayor ocupación relativa de recursos combinacionales y secuenciales son el cálculo de la varianza y el cálculo del parámetro de entrenamiento C, Figura 4.19 y Figura 4.20. Este hecho supone una diferencia destacable con respecto a los resultados obtenidos en la implementación sobre la FPGA RTAX2000S, en donde el módulo de mayor ocupación relativa es la red de pesos. Esta variación se debe a que los multiplicadores necesarios en la red de pesos, implementados en el caso de la FPGA RTAX2000S mediante recursos combinacionales y secuenciales, se implementan haciendo uso de los bloques DSP dedicados disponibles en la FPGA RTAX2000S-DSP. Por este motivo, el módulo dominante en cuanto a ocupación de bloques DSP es la red de pesos como se observa en la Figura 4.19 y la Figura 4.20. De la misma forma que ocurre en la implementación sobre la FPGA RTAX2000S, la LUT de pesos es el módulo que presenta mayores necesidades de almacenamiento (bloques RAM) en la implementación sobre la FPGA RTAX2000S-DSP.

Se había planteado al inicio de este capítulo que el principal objetivo de este trabajo es mostrar la viabilidad de la implementación de la aproximación entera de la transformada POT sobre un dispositivo de lógica programable tipo FPGA, por lo que la implementación no ha sido optimizada

a nivel de tasa de datos sino a nivel de uso de recursos. En cualquier caso, puede estudiarse la tasa de datos resultante de la implementación realizada dentro del sistema de compresión. La implementación propuesta proporciona una muestra de salida de la transformada cada cuatro ciclos de reloj. Esto proporciona una tasa de bits mínima para la configuración C1 de 12,5 Mmuestras/s sobre la FPGA RTAX2000S mientras que en la configuración C8 proporciona la máxima tasa de bits, 18,4 Mmuestras/s sobre la FPGA RTAX2000S-DSP.

Para completar el sistema de compresión 3-D de una imagen multiespectral o hiperespectral, además de la transformada espectral POT es necesario emplear una transformada 2-D. El estándar de compresión de imágenes CCSDS-122.0 define el empleo de la transformada wavelet discreta, DWT (del inglés, *Discrete Wavelet Transform*), como decorrelador 2-D. Por lo tanto, es necesario que, al margen de la implementación realizada de la aproximación entera de la transformada POT, existan implementaciones de la transformada DWT disponibles para operar a bordo de un satélite.

En la actualidad, el ASIC CWICOM (del inglés, *CCSDS Wavelet Image COMpression*) ha sido diseñado y cualificado para operar en el entorno espacial. Este ASIC implementa la transformada wavelet definida en el estándar CCSDS-122.0. A nivel general, en una misión real, el sistema se define mediante una primera etapa en la que se implementa la aproximación entera de la transformada POT sobre una FPGA, acompañada de una segunda etapa en la que se usa el ASIC CWICOM para obtener la imagen comprimida. El ASIC CWICOM es capaz de proporcionar muestras a su salida con una tasa de bits de hasta 60 Mmuestras/s [96]. A pesar de que la implementación de ambos elementos de compresión se encuentran en la misma magnitud, futuras implementaciones de la aproximación entera de la transformada POT se centrarán en la mejora de la tasa de bits de salida de tal forma que se obtenga una muestra a la salida por ciclo de reloj. Esta mejora se contempla tras la aparición de nuevos dispositivos FPGA cualificados para el espacio como puede ser la familia de FPGAs RTG4 que proporcionan capacidades y recursos no disponibles con anterioridad en la industria espacial.

4.5 Conclusiones

Este capítulo evalúa la complejidad computacional a nivel de implementación hardware de la aproximación entera de la transformada POT. Se proporciona una descripción de las operaciones involucradas en su cálculo con el fin de ser usadas e implementadas haciendo uso, únicamente, de aritmética entera, reduciendo de esta forma su complejidad en la implementación sobre un dispositivo de lógica reprogramable tipo FPGA. Como estrategia de implementación, se incorpora el uso de una LUT con el fin de eliminar las operaciones de mayor carga computacional de la transformada POT.

Se ha realizado la síntesis del código diseñado sobre dos FPGAs calificadas para espacio (RTAX2000S y RTAX2000S-DSP) haciendo uso de diferente configuraciones en términos de número de muestras de entrada representadas por diferentes anchos de bit. Se ha comprobado que, para cualquier configuración de las estudiadas, el diseño puede implementarse en las FPGAs objetivo, proporcionando un peor caso de ocupación del 74 % de celdas combinacionales sobre la FPGA RTAX2000S. Por el contrario, el mejor caso obtenido para el mismo dispositivo proporciona una máxima ocupación de celdas combinacionales del 24,13 %. Además, se ha observado que la FPGA RTAX2000S-DSP ofrece la posibilidad de reducir sustancialmente el total de celdas combinacionales usadas mediante el uso de los bloques DSP dedicados que implementan esta familia de dispositivos. Asimismo, esta familia de FPGAs alcanza mayores valores de frecuencia máxima de operación, dando lugar en el mejor caso a una frecuencia de 73 MHz.

Los resultados obtenidos de la implementación de la transformada POT muestran que su implementación hardware a bordo sobre dispositivos FPGA calificados para el espacio es viable. De este modo se satisface el objetivo propuesto en esta tesis doctoral centrado en el estudio e implementación del algoritmo de compresión basado en la transformada POT aportando diferentes soluciones que pueden ser empleadas en futuras misiones espaciales.

Capítulo 5

Implementación del estándar CCSDS 123.0 con pérdidas y control de la tasa de bits

---

Los algoritmos de compresión predictivos representan una alternativa atractiva a aquellos basados en transformadas para ser desplegados a bordo de un satélite. Su principal ventaja reside en su menor complejidad computacional. Sin embargo, presentan una clara desventaja en cuanto a la posibilidad de incluir mecanismos para el control de la tasa de bits.

En este capítulo, se presenta la implementación de un algoritmo de compresión basado en el predictor descrito en el estándar CCSDS 123.0-B-1 en el que se incluye una etapa dedicada para el control de la tasa de bits y diferentes alternativas al codificador entrópico. Por lo tanto, la implementación puede emplearse como extensión del estándar CCSDS 123.0 en la que se introducen pérdidas y control de la tasa de bits.

5.1 Introducción

Actualmente, el documento CCSDS 123.0-B-1 [25] publicado por el comité CCSDS (del inglés, *Consultative Committee for Space Data Systems*) establece el estándar de compresión sin pérdidas de imágenes hiperespectrales para ser desplegado a bordo de satélites. Este estándar se basa en un algoritmo predictivo de compresión en donde no es posible la inclusión de pérdidas en el proceso de compresión y descompresión de datos. Por lo tanto, como se ha ido desgranando en capítulos anteriores, al ser un algoritmo sin pérdidas presenta para cada imagen un límite teórico de compresión que queda definido por el teorema de Entropía de Shannon [42].

Con el fin alcanzar ratios de compresión mayores, el comité CCSDS se encuentra en fase de estandarización de una extensión con pérdidas del estándar CCSDS 123.0. Además, es deseable que esta extensión posea mecanismos de control en la tasa de bits de salida del algoritmo, es decir, debe ser capaz de obtener a su salida una tasa de bits constante definida como entrada en el algoritmo. Disponer de estas características permite estimar en estadios tempranos de planificación de misión el ancho de banda ocupado por el sistema de compresión facilitando a su vez la optimización del mismo.

El algoritmo implementado y verificado en este capítulo toma como base el estándar CCSDS 123.0 y lo extiende con el fin de obtener un algoritmo capaz de operar con pérdidas y con control de la tasa de bits. Para ello, se toma como elemento de referencia el código escrito en lenguaje C/C++ proporcionado por la Universidad Politécnica de Turín que implementa dicho estándar. Sobre este código de referencia se ha implementado el mecanismo de control de la tasa de bits haciendo uso de un cuantificador adaptativo y las modificaciones necesarias para transformar el código de referencia en un código sintetizable mediante herramientas de síntesis de alto nivel.

La implementación RTL presentada en este capítulo se ha obtenido mediante la plataforma CatapultC HLS teniendo como objetivo la síntesis del código sobre la FPGA XQR5VFX130 de la familia Virtex-5QV del fabricante Xilinx.

5.2 Algoritmo de compresión predictivo con pérdidas y control de la tasa de bits

En esta sección se describe el algoritmo de compresión con pérdidas de imágenes hiperespectrales predictivo y con control de la tasa de bits objeto de implementación. El algoritmo propuesto toma como base el estándar sin pérdidas CCSDS 123.0 [25] extendiéndolo de tal forma que pueda operar como un algoritmo predictivo con pérdidas y que además cuente con mecanismos de control de tasa de bits. Además, el algoritmo resultante preserva las características del algoritmo CCSDS 123.0 pudiendo emplearse como algoritmo sin pérdidas mediante la correcta selección de los parámetros de entrada.

5.2.1 Descripción del estándar CCSDS 123.0-B-1

El estándar de compresión sin pérdidas de imágenes multiespectrales e hiperespectrales CCSDS 123.0-B-1 describe un algoritmo que permite la compresión sin pérdidas de imágenes tridimensionales generadas por los sensores hiperespectrales a bordo de satélites. Este estándar se basa en el compresor *Fast Losseless* [56] [57], en adelante FL, adaptado al empleo de aritmética entera únicamente. El algoritmo estandarizado se ha concebido con el objetivo de ser desplegado a bordo de un satélite por lo que se han tenido en cuenta tanto el requisito de baja complejidad computacional como el de baja necesidad almacenamiento interno que, dadas las características de los dispositivos hardware disponibles a bordo, resultan fundamentales.

Como se muestra en la Figura 5.1, el compresor consta de dos etapas funcionales: el predictor y el codificador entrópico.

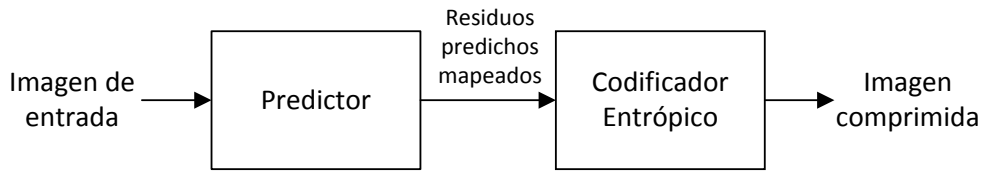


Figura 5.1 Estructura del compresor CCSDS 123.0-B-1

5.2.1.1 Predictor

La etapa predictiva proporciona a su salida los residuos predichos mapeados de cada muestra de entrada. Para ello, emplea un algoritmo lineal adaptativo en la predicción del valor de cada muestra de la imagen. El cálculo de la muestra predicha emplea las muestras contenidas en la vecindad tridimensional más cercana a la muestra de entrada. A la diferencia entre el valor de la muestra predicha y la muestra de entrada se le denomina residuo predicho y representa la entrada a la operación de mapeado. En ella, se asigna un número entero sin signo con el mismo rango dinámico que la muestra de entrada al valor del residuo predicho obteniéndose de esta forma el residuo predicho mapeado. La Figura 5.2 presenta los principales módulos de la etapa funcional del predictor.

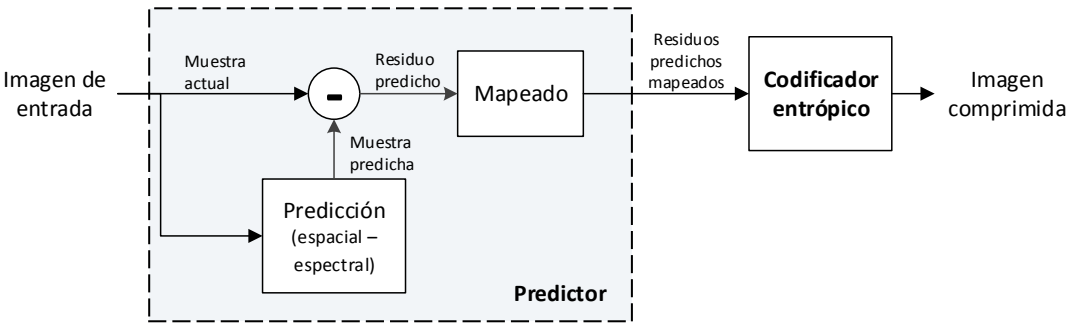


Figura 5.2 Detalle de los principales módulos que componen el predictor

Una imagen hiperespectral puede representarse matemáticamente como un cubo tridimensional que contiene $N_z \times N_y \times N_x$ píxeles, siendo:

- N_z el número de bandas de la imagen
- N_y el número de píxeles de una fila que contiene la imagen
- N_x el número de filas de la imagen

Siguiendo esta notación se puede definir una muestra perteneciente a una imagen hiperespectral como $s_{z,y,x}$, en adelante muestra actual, en donde $z \in [0 \leq z \leq N_z - 1]$, $y \in [0 \leq y \leq N_y - 1]$ y $x \in [0 \leq x \leq N_x - 1]$. El valor de la muestra predicha, $\hat{s}_{z,y,x}$, depende de los valores de las muestras cercanas en una vecindad compuesta por la banda actual y P bandas anteriores, siendo

P un valor entero entre 0 y 15, $P \in [0 \leq P \leq 15]$. La Figura 5.3, presenta la vecindad tenida en cuenta en el cálculo del valor de la muestra predicha.

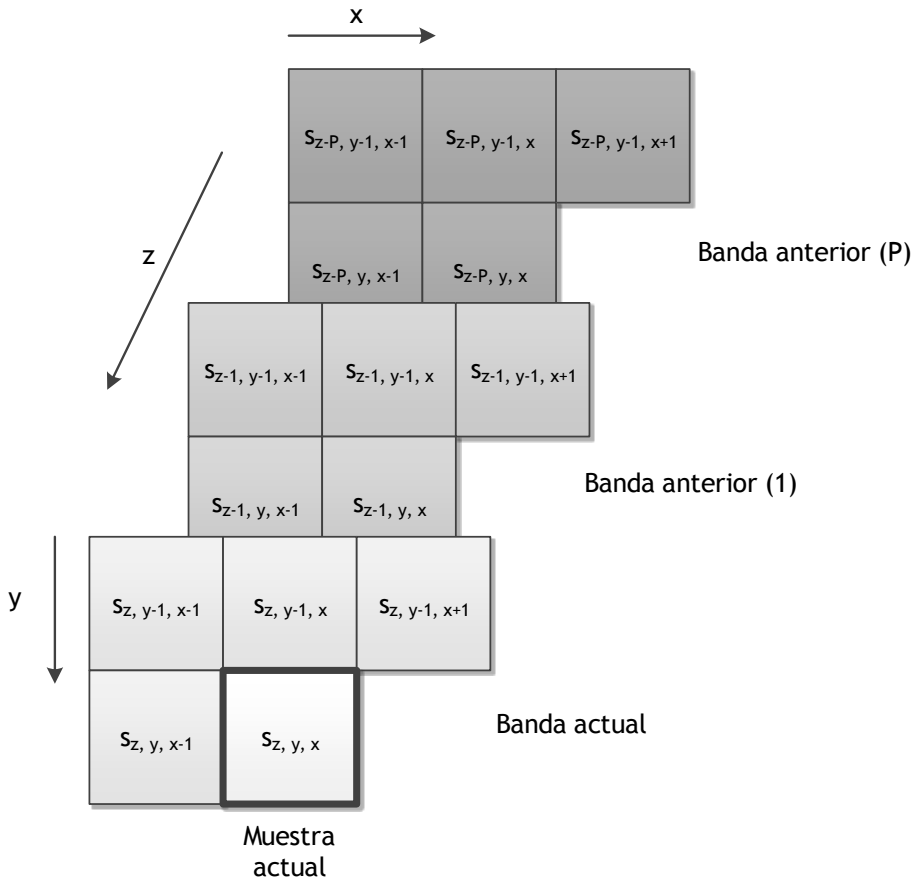


Figura 5.3 Cálculo de la muestra predicha - vecindad

Como primer paso, se computa la suma local de las muestras contenida en la vecindad. La suma local se define como la suma ponderada de las muestras correspondientes a una banda espectral z adyacentes a la muestra actual a comprimir ($s_{z,y,x}$). Existen dos modos para realizar el cómputo de la suma local: orientado a vecinos y orientado a columnas. La Figura 5.4 presenta coloreadas las muestras de la vecindad usadas en el cálculo de la suma local dependiendo del modo escogido por el usuario.

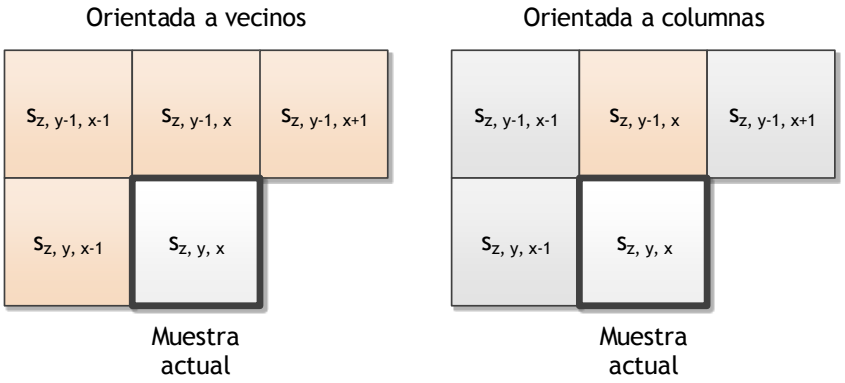


Figura 5.4 Suma local: orientada a vecinos (izquierda) y orientada a columnas (derecha)

La suma local orientada a vecinos se define mediante la expresión (5.1), mientras que el cálculo de la suma local orientada a columnas se presenta en la ecuación (5.2). Nótese que ambas expresiones son correctamente truncadas en aquellos casos en que no existen vecinos disponibles como, por ejemplo, cuando la muestra actual pertenece a la primera fila ($y = 0$), o lo que es lo mismo las muestras actuales de la forma $s_{z,0,x}$.

$$\sigma_{z,y,x} = \begin{cases} s_{z,y,x-1} + s_{z,y-1,x-1} + s_{z,y-1,x} + s_{z,y-1,x+1}, & y > 0, 0 < x < N_x - 1 \\ 4 \cdot s_{z,y,x-1}, & y = 0, x > 0 \\ 2 \cdot (s_{z,y-1,x} + s_{z,y-1,x+1}), & y > 0, x = 0 \\ s_{z,y,x-1} + s_{z,y-1,x-1} + 2 \cdot s_{z,y-1,x}, & y > 0, x = N_x - 1 \end{cases} \tag{5.1}$$

$$\sigma_{z,y,x} = \begin{cases} 4 \cdot s_{z,y-1,x}, & y > 0, 0 < x < N_x - 1 \\ 4 \cdot s_{z,y,x-1}, & y = 0, x > 0 \end{cases} \tag{5.2}$$

El valor de la suma local se emplea en el cómputo del valor de las diferencias locales. Existen dos tipos de diferencias locales dependiendo de la vecindad escogida para su cómputo: la diferencia local central y las diferencias locales direccionales. La Figura 5.5 presenta los tipos de diferencias locales en función de su tipo.

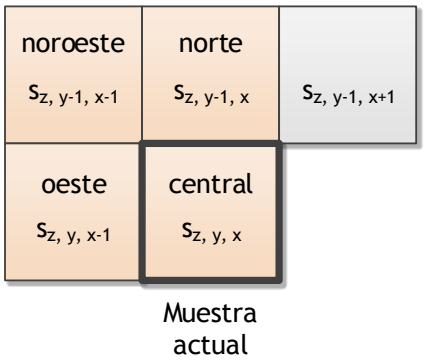


Figura 5.5 Tipos de diferencias locales en función de su localización

La diferencia local central, $d_{z,y,x}$, emplea para su cálculo la suma local y el valor de la muestra actual, tal y como se define en la ecuación (5.3).

$$d_{z,y,x} = 4 \cdot s_{z,y,x} - \sigma_{z,y,x} \tag{5.3}$$

Como se muestra en la Figura 5.5, existen tres diferencias locales direccionales denominadas: norte, $d^{N}_{z,y,x}$, oeste, $d^{O}_{z,y,x}$, y noroeste, $d^{NO}_{z,y,x}$. Las ecuaciones (5.4), (5.5) y (5.6) describen el cálculo de cada una de las diferencias locales.

$$d^{N}_{z,y,x} = \begin{cases} 4 \cdot s_{z,y-1,x} - \sigma_{z,y,x}, & y > 0 \\ 0, & y = 0 \end{cases} \tag{5.4}$$

$$d^{W}_{z,y,x} = \begin{cases} 4 \cdot s_{z,y,x-1} - \sigma_{z,y,x}, & y > 0, x > 0 \\ 4 \cdot s_{z,y-1,x} - \sigma_{z,y,x}, & y > 0, x = 0 \\ 0, & y = 0 \end{cases} \tag{5.5}$$

$$d^{NW}_{z,y,x} = \begin{cases} 4 \cdot s_{z,y-1,x-1} - \sigma_{z,y,x}, & y > 0, x > 0 \\ 4 \cdot s_{z,y-1,x} - \sigma_{z,y,x}, & y > 0, x = 0 \\ 0, & y = 0 \end{cases} \tag{5.6}$$

Como parámetro de entrada el usuario puede escoger entre dos modos para el cómputo de la predicción de una imagen: la predicción completa o la predicción reducida. En el modo de predicción reducida, la predicción depende de la suma ponderada de las diferencias locales centrales contenidas en las P bandas espectrales precedentes, por lo tanto, no se emplean las diferencias locales direccionales pudiendo evitarse su cálculo y así reduciendo la complejidad computacional del predictor. En el modo de predicción completa, la predicción depende del cálculo de la suma ponderada de las

diferencias locales centrales en las P bandas precedentes y de las tres diferencias locales direccionales calculadas para la banda actual.

Existen estudios que evalúan comparativamente la idoneidad del modo de predicción completo frente al modo de predicción reducido [55]. El uso del modo de predicción reducido combinado con la suma local orientada a columnas alcanza tamaños menores de imagen comprimida para imágenes de entrada no calibradas generadas por sensores tipo *pushbroom*, es decir, que capturan todas las bandas espectrales de una línea de píxeles. Por el contrario, el uso del modo de predicción completo en combinación con la suma local orientada a vecinos proporciona tamaños menores imágenes comprimidas para imágenes de entrada calibradas y aquellas generadas por sensores tipo *whiskbroom*. Estos sensores capturan la información espectral de las bandas intercaladas por píxel.

Puede definirse el vector de diferencias locales $U_{z,y,x}$ como el vector columna que contiene las diferencias locales direccionales (dependiendo del modo de predicción empleado) y una diferencia local para cada una de las P bandas escogidas para la predicción. La ecuación (5.7) presenta el vector de diferencias locales en el caso de la predicción completa, mientras que, en la ecuación (5.8) se define el vector de diferencias locales para el caso del modo de predicción reducida.

$$U_{z,y,x} = \begin{bmatrix} d_{z,y,x}^{N} \\ d_{z,y,x}^{W} \\ d_{z,y,x}^{NW} \\ d_{z-1,y,x} \\ \vdots \\ d_{z-P,y,x} \end{bmatrix} \tag{5.7}$$

$$U_{z,y,x} = \begin{bmatrix} d_{z-1,y,x} \\ d_{z-2,y,x} \\ \vdots \\ d_{z-P,y,x} \end{bmatrix} \tag{5.8}$$

Para el cálculo de la muestra predicha escalada, $\tilde{s}_{z,y,x}$, se emplea el valor de la diferencia central local predicha, $\hat{d}_{z,y,x}$. Esta diferencia local

predicha se calcula mediante el producto escalar del vector de diferencias locales, $U_{z,y,x}$, y su valor de peso correspondiente, tal y como se define en la ecuación (5.9). De esta forma puede formarse un vector columna de pesos, $W_{z,y,x}$, en el que cada fila contiene el valor de ponderación de cada diferencia local. La definición de este vector de pesos se presenta en la ecuación (5.10) para el modo de predicción completa y en la ecuación (5.11) para el modo de predicción reducida.

$$\widehat{d}_{z,y,x} = W_{z,y,x} \cdot U_{z,y,x} \tag{5.9}$$

$$W_{z,y,x} = \begin{bmatrix} w^N_{z,y,x} \\ w^W_{z,y,x} \\ w^{NW}_{z,y,x} \\ w_{z-1,y,x} \\ \vdots \\ w_{z-P,y,x} \end{bmatrix} \tag{5.10}$$

$$W_{z,y,x} = \begin{bmatrix} w_{z-1,y,x} \\ w_{z-2,y,x} \\ \vdots \\ w_{z-P,y,x} \end{bmatrix} \tag{5.11}$$

La actualización del valor de los pesos pertenecientes al vector de pesos, depende del error de predicción escalado, $e_{z,y,x}$, ecuación (5.12), y se realiza una vez finalizado el cálculo de la predicción de la muestra actual antes de realizar la predicción de la muestra siguiente.

$$e_{z,y,x} = 2 \cdot s_{z,y,x} - \tilde{s}_{z,y,x} \tag{5.12}$$

La muestra predicha, $\hat{s}_{z,y,x}$, es calculada mediante la ecuación (5.13).

$$\hat{s}_{z,y,x} = \left\lfloor \frac{\tilde{s}_{z,y,x}}{2} \right\rfloor \tag{5.13}$$

Finalmente, como salida del predictor, el valor del residuo predicho mapeado se define como el mapeo en números enteros del residuo predicho, $\Delta_{z,y,x}$. Dicho residuo predicho, es la diferencia entre el valor de la muestra predicha, $\hat{s}_{z,y,x}$, y la muestra actual, ecuación (5.14).

$$\Delta_{z,y,x} = s_{z,y,x} - \hat{s}_{z,y,x} \tag{5.14}$$

5.2.1.2 Codificador entrópico

La imagen comprimida consiste en una cabecera que contiene los parámetros de compresión empleados y en un cuerpo, que contiene la codificación de los residuos predichos mapeados. El codificador entrópico es el responsable de la generación del cuerpo de la imagen comprimida. Para ello, ajusta adaptativamente los parámetros de codificación adaptándose a los cambios estadísticos de los residuos predichos mapeados.

Los residuos predichos mapeados se codifican secuencialmente en el orden especificado (BSQ o BI) por el usuario, indicándose además en la cabecera de la imagen comprimida. Nótese que este orden de codificación es independiente del orden de los datos en la imagen de origen y al orden de procesado en el predictor.

El usuario puede elegir entre dos tipos de codificación entrópica para la codificación de los residuos predichos mapeados: la codificación entrópica adaptativa por muestras o la codificación entrópica adaptativa por bloques. La codificación entrópica adaptativa por muestras alcanza, generalmente, menores tamaños de imagen comprimida que la codificación entrópica adaptativa por bloques [55].

En el caso de la codificación entrópica adaptativa por muestras, cada residuo predicho mapeado se codifica mediante una palabra binaria de longitud variable. Estos códigos de longitud variable son adaptativamente escogidos en base a parámetros estadísticos actualizados muestra a muestra. Es importante señalar el hecho de que, para cada banda espectral, se mantienen parámetros estadísticos independientes y que, el tamaño de la imagen comprimida, no depende del orden de codificación de los residuos predichos mapeados.

La secuencia de residuos predichos mapeados se divide en bloques en el caso de la codificación entrópica por bloques. La codificación se realiza adaptativamente en cada bloque dependiendo del orden de codificación

escogido, por lo tanto, el tamaño de la imagen comprimida depende del orden de codificación empleado ya que cada bloque de residuos predichos mapeados puede contener valores de la misma o de diferentes bandas espectrales.

5.2.2 Algoritmo de control de la tasa de bits como extensión del estándar CCSDS 123.0-B-1

El control de la tasa de bits es considerado un reto en los algoritmos de compresión predictivos debido a que no existe una relación matemática simple entre la tasa de bits y el residuo predicho cuantificado. El principal objetivo de los métodos de optimización del binomio tasa de bits – distorsión es minimizar la distorsión de una imagen o video comprimido en el que se ha establecido una limitación sobre la tasa de bits.

El Departamento de Electrónica y Telecomunicación de la Universidad Politécnica de Turín, ha desarrollado un algoritmo de control de tasa de bits, con optimización de la distorsión, para la compresión de imágenes hiperespectrales [81]. El algoritmo se utiliza con codificadores predictivos que puedan ser empleados a bordo de un satélite. En el esquema propuesto, la imagen de entrada es dividida en bloques de 16 x 16 píxeles y para cada bloque, se realiza una estimación de la tasa de bits y la distorsión introducida mediante el cálculo de un conjunto de pasos de cuantificación.

En detalle el algoritmo se compone de las siguientes etapas:

1. **Etapa de entrenamiento.** Ésta realiza la estimación de la varianza de los residuos no-cuantificados empleando el predictor definido en el estándar CCSDS 123.0 presentado en la sección 3.2.1.1 para un número de líneas dentro de cada bloque. El número de líneas a emplear es un parámetro de entrada que define el usuario.
2. **Etapa de optimización.** El tamaño del escalón de cuantificación a emplear para cada bloque se obtiene en esta etapa. En primer lugar, se calcula el conjunto de escalones de cuantificación que proporcionan la tasa de bits objetivo pero

que resultan sub-óptimos en términos de distorsión. En segundo lugar, se emplea el algoritmo denominado "dieta selectiva" en donde se seleccionan aquellos pasos de cuantificación que proporcionan una menor distorsión.

Sobre el diagrama de bloques del estándar CCSDS 123.0 original (ver Figura 5.2), la Figura 5.6 muestra (en color) los nuevos bloques que deben ser incluidos para el control de la tasa de bits. Tanto la etapa de entrenamiento como la etapa de optimización se encuentran en el módulo denominado control tasa de bits. Se observa que, al margen de la etapa de control de la tasa de bits, se incluye como parte del predictor el descompresor local. La necesidad de incluir el descompresor local viene motivada porque las muestras de los vecinos necesarias para la predicción de la muestra actual son muestras reconstruidas, no muestras originales, ya que se han introducido pérdidas en la codificación mediante la cuantificación de los residuos predichos.

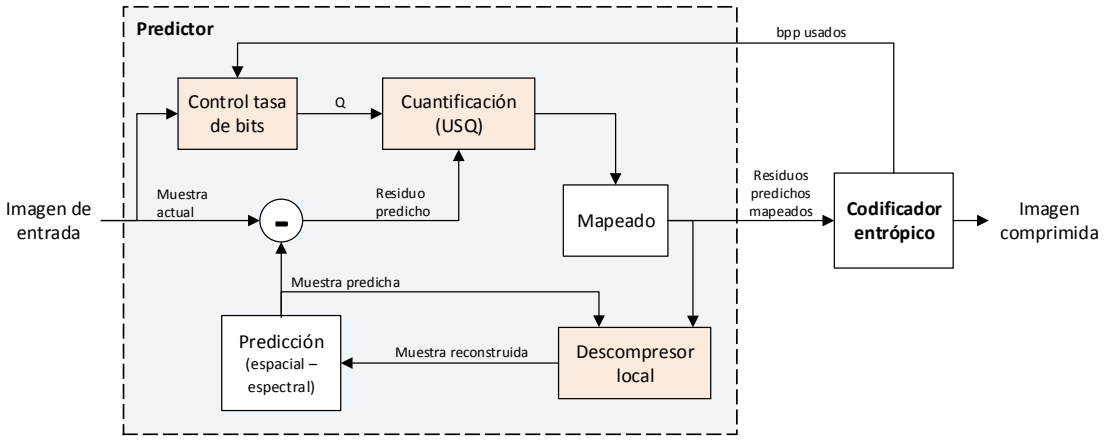


Figura 5.6 Diagrama de bloques del algoritmo de compresión predictivo con pérdidas y control de la tasa de bits

Este algoritmo de control de tasa de bits y optimización de distorsión se ha propuesto como extensión del estándar de compresión de imágenes hiperespectrales sin pérdidas CCSDS 123.0 con el fin de proporcionar una solución a la compresión con pérdidas, casi-sin pérdidas (*near lossless*) e híbrida (casi-sin pérdidas con control de tasa de bits) bajo un mismo algoritmo.

La principal desventaja en la implementación de este algoritmo reside en su naturaleza serie, ya que es necesario realizar tanto la etapa de entrenamiento como de optimización antes de realizar la predicción del bloque actual. Este hecho supone una penalización en la implementación en términos de latencia si se compara con el estándar CCSDS 123.0.

Con el fin de mejorar la latencia en la ejecución de este algoritmo y limitar de esta forma la penalización introducida, puede plantearse una aproximación paralela en la que en cada bloque se calculen tanto la etapa de entrenamiento como la etapa de optimización necesarias para la predicción del bloque siguiente. Es notorio que la implementación de esta aproximación ha de tener en cuenta las dependencias de datos existentes en cada una de las fases del algoritmo. Por un lado, es necesario proporcionar un predictor actualizado en la fase de entrenamiento, y por otro, se ha de proporcionar una tasa de bits objetivo para el bloque siguiente basado en la tasa de bits del bloque actual. En [97] se muestra una posible implementación en la que, teniendo en cuenta las dependencias de datos se propone una planificación paralela viable del algoritmo.

Otra aportación incluida en [81], reside en la inclusión de un codificador de rango como codificador entrópico. Mientras que el estándar CCSDS 123.0 define dos opciones de codificación entrópica: la codificación entrópica adaptativa por muestras o la codificación entrópica adaptativa por bloques; el algoritmo presentado añade un codificador de rango, que fundamentalmente es un codificador aritmético cuya principal ventaja es la capacidad de proporcionar tasas de bits por debajo de 1 bpp. Sin embargo, para su funcionamiento es necesario mantener cuatro modelos estadísticos independientes de codificadores de rango para cada banda: indicador de valor cero/no-cero (2 símbolos, 1 bit), valores inferiores al umbral definido (128 símbolos, 7 bits), valores contenidos en el byte menos significativo superiores al umbral definido (256 símbolos, 8 bits) y valores pertenecientes al byte más significativo superiores al umbral definido (256 símbolos, 8 bits). Este hecho proporciona un aumento en las necesidades de almacenamiento con respecto a los codificadores entrópicos basados en el estándar sin pérdidas.

El algoritmo descrito en esta sección, es una primera aproximación para la implementación a bordo de un algoritmo de tipo predictivo con pérdidas, control de tasa de bits y optimización de la distorsión introducida. Sin embargo, presenta limitaciones derivadas de las dependencias de datos en las etapas introducidas que complican su ejecución en el hardware disponible a bordo de un satélite.

5.2.3 Algoritmo ligero de control de la tasa de bits como extensión del estándar CCSDS 123.0-B-1

En la sección anterior se pusieron de manifiesto las limitaciones del algoritmo propuesto en [81] como algoritmo de control de la tasa de bits. Por este motivo, en [30] se presenta un nuevo algoritmo que aborda las principales limitaciones del anterior, disminuyendo su latencia de ejecución y simplificando el cálculo de la etapa de optimización, o lo que es lo mismo, el cálculo del escalón de cuantificación.

El algoritmo se basa en la aplicación de un único paso de cuantificación que se obtiene a través de un cuantificador escalar uniforme para cada fila de píxeles, $y \in [0 \leq y \leq N_y - 1]$, incluyendo todas sus bandas, $z \in [0 \leq z \leq N_z - 1]$. El cálculo de cada uno de los escalones de cuantificación se realiza mediante la caracterización estadística de los residuos predichos en cada banda. La Figura 5.7 presenta la denominación de cada uno de los píxeles de la imagen hiperespectral dependiendo de su localización que emplea el algoritmo de compresión para su funcionamiento.

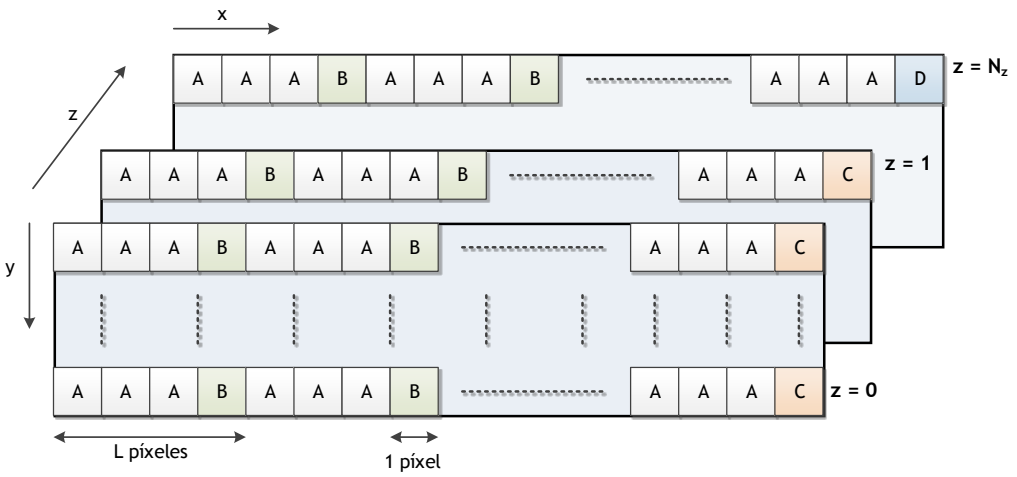


Figura 5.7 Clasificación de los píxeles contenidos en una imagen hiperespectral empleada por el algoritmo de compresión

Los píxeles de denominados tipo A (residuos no-cuantificados de la muestra actual) son propagados a las siguientes etapas del algoritmo de compresión codificándose entrópicamente y proporcionándolos a la salida del compresor. Además, este tipo de píxeles son almacenados en el algoritmo en una memoria local de residuos no-cuantificados. En el momento de que el codificador llega a un píxel tipo B tras L - 1 píxeles de tipo A, se calcula y almacena la mediana de los residuos contenidos en la memoria local de residuos no-cuantificados. Esta secuencia se repite cada L-1 píxeles hasta llegar al final de cada fila y en cada banda espectral z, caso en el que el algoritmo se encuentra con un píxel de tipo C. El valor de L puede ser definido por el usuario, consiguiéndose un buen compromiso de prestaciones para $L = 17$ píxeles [30]. En este caso de píxeles tipo C, el algoritmo no computa únicamente la mediana de los residuos no-cuantificados, sino que realiza el cálculo de la mediana de las medianas almacenadas. De esta forma, para cada banda se almacena una mediana de medianas que se denominará m_z. Finalmente, cuando se alcanza el último píxel de la última banda espectral, esto es, en el caso de $x = N_x$ e $z = N_z$, se alcanza un píxel de tipo D. En ese instante, el algoritmo determina el paso de cuantificación Q para la siguiente fila de píxeles junto con todas sus bandas.

Para el cálculo del escalón de cuantificación Q, se emplea el vector de medianas, m_z, y se busca el valor entero impar en el rango $[1, Q_{max}]$, que satisfaga la siguiente expresión:

$$Q = arg\ _{q\in\{1,3,5\ldots,Q_{max}\}} min \left| \sum_{z=0}^{N_z-1} R(m_z, q) - R_{objetivo} \right| \tag{5.15}$$

Satisfacer la ecuación (5.15) implica obtener un paso de cuantificación Q tal que se proporciona una tasa de bits para una fila junto con todas sus bandas espectrales lo más cercana posible a la tasa de bits objetivo, $R_{objetivo}$.

El valor, $R(m_z, q)$, almacena los valores de un cuantificador escalar uniforme de escalón Q. En (5.16) se presenta la ecuación empleada para este fin.

$$\begin{aligned} R(m, Q) = &-\left(1 - e^{-\frac{Q}{2m}}\right) log_2\left(1 - e^{-\frac{Q}{2m}}\right) \\ &- \frac{e^{-\frac{Q}{2m}}}{\log(2)}\left[log\left(\frac{1 - e^{-\frac{Q}{m}}}{2}\right) + \frac{Q}{2m} - \frac{Q}{m\left(1 - e^{-\frac{Q}{m}}\right)}\right] \end{aligned} \tag{5.16}$$

Puede observarse en la ecuación (5.16) la complejidad que presenta este cálculo. Por este motivo, se propone el uso de una tabla, cuyo contenido se calcula previamente y se almacena en el interior del algoritmo. La cantidad de memoria necesaria para almacenar los valores de la tabla de tasas de bits viene determinada tanto por el número de valores enteros de medianas como por el número de escalones de cuantificación Q a considerar. Se demuestra en [27] que, seleccionando un rango de valores $m \in [0, 1023]$ y $Q \in [1, 3, 5, \ldots, 511]$, se obtienen resultados satisfactorios incluso para obtener tasas de bits por debajo de 1 bit por píxel (bpp). Si se considera además el hecho de que los valores enteros para las tasas de bits dentro de la tabla tengan una resolución de 16 bits, la tabla puede ser realizada con una LUT de 512kB, valor que no supone una limitación para su implementación a bordo de un satélite.

Queda de manifiesto que existen diferencias importantes entre el algoritmo de control de la tasa de bits descrito en la sección 5.2.2 y el descrito en esta sección. En particular, el algoritmo de la sección 5.2.2 presenta el principal inconveniente de tener que ejecutar el predictor en dos ocasiones, una para la estimación de las estadísticas en la etapa de entrenamiento y la segunda para la codificación de la muestra actual, lo que provoca una penalización en términos de latencia del algoritmo. Además, en la etapa de optimización emplea una mayor complejidad derivada del algoritmo de "dieta selectiva" que implementa. Por el contrario, el algoritmo presentado en esta sección supera ambos inconvenientes simplificando el cálculo de estadísticas mediante el cálculo de medianas y empleando una LUT para la obtención de la tasa de bits en función del paso de cuantificación. Por estos motivos, supone una opción más atractiva comparativamente para ser implementado a bordo de un satélite.

5.3 Implementación del estándar CCSDS 123.0 con pérdidas y control de la tasa de bits

El algoritmo tomado como base en la implementación es el descrito en la sección 5.2.2. Este algoritmo ha sido proporcionado por la Universidad Politécnica de Turín gracias al acuerdo de colaboración entre esta institución y el Instituto Universitario de Microelectrónica Aplicada (IUMA). La implementación original del algoritmo se encuentra escrita en lenguaje C/C++, por lo tanto, resulta conveniente su estudio teniendo en mente su posterior implementación mediante metodologías de diseño de síntesis de alto nivel, en adelante HLS (del inglés, *High Level Synthesis*).

La herramienta empleada en la síntesis de alto nivel ha sido CatapultC. Como se describe en el capítulo 3, esta herramienta posee una metodología de diseño dividida en pasos que hace posible la generación de código RTL a partir de código C/C++. Con el fin de obtener un código RTL optimizado para su síntesis, se realizaron las modificaciones presentadas en las siguientes secciones sobre el código original. Para asegurar que las modificaciones realizadas no distorsionan el comportamiento del algoritmo, el código fuente original se toma como referencia ante cualquier cambio introducido, es decir, se compara tras cada cambio el resultado de la compresión entre el algoritmo original y el algoritmo modificado.

La Figura 5.8 muestra el diagrama de flujo seguido con el objeto de obtener el código RTL. Se observa que, al margen de las modificaciones necesarias a realizar sobre el código fuente original, se proponen en el código aportaciones que dan versatilidad al algoritmo resultante, proporcionando además la posibilidad de obtener diferentes implementaciones dependiendo de las necesidades específicas de cada misión espacial.

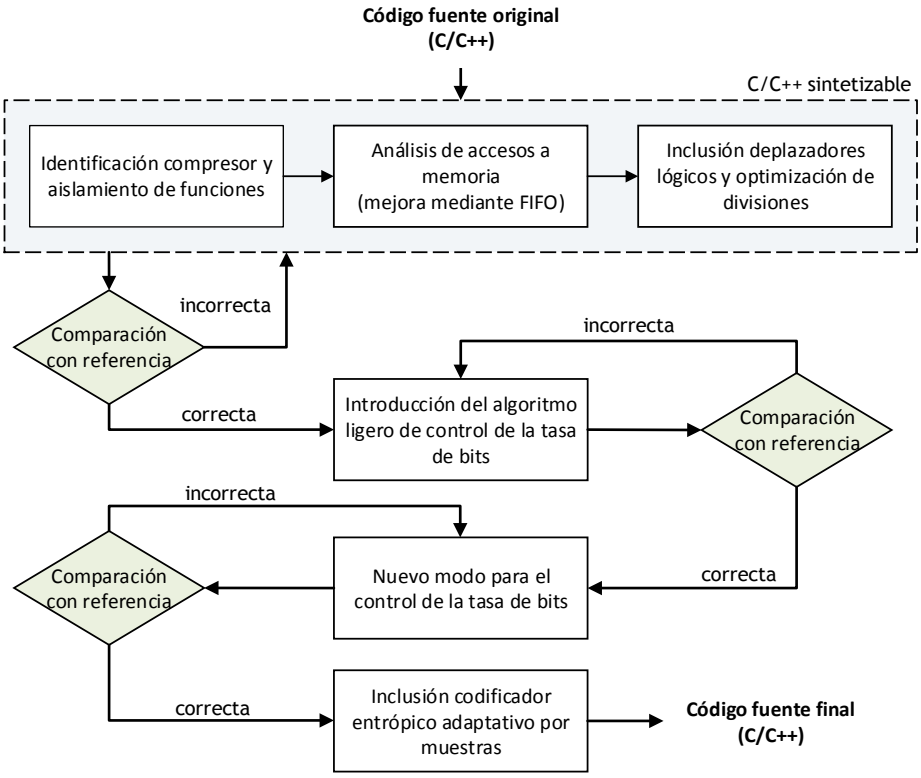


Figura 5.8 Diagrama de flujo empleado en las modificaciones y aportaciones realizadas sobre el código fuente original

5.3.1 Modificaciones realizadas sobre el código original

El primer paso para realizar las modificaciones sobre el algoritmo original es el análisis del código fuente para la identificación del módulo de mayor jerarquía que implemente el algoritmo de compresión en el diseño. La función *predict()* implementa tanto el predictor como el codificador entrópico dentro del algoritmo de compresión. La compresión se realiza píxel a píxel siguiendo recorriendo la imagen en orden BIL (del inglés, *Band Interleaved by* Line). La Figura 5.9 muestra en pseudocódigo el contenido de la función *predict()*.

```
1: for (y = 0; y < Ny; y++)
2:     for (z = 0; z < Nz; z++)
3:         for (x = 0; x < Nx; x++)
4:             readSample(s_{z,y,x}, neighbourhood)
5:             computeDifferences(U_{z,y,x})
6:             computeScaledPredicted(s̃_{z,y,x})
7:             computeMappedResidual(Δ_{z,y,x})
8:             decoder()
9:             updateWeights(W_{z,y,x})
10:            rangeEncoder(Δ_{z,y,x})
11:        end x
12:    end z
13:    rateControl()
14: end y
```

Figura 5.9 Pseudocódigo de la función predict()

En el capítulo 3 se constata que, realizando una descomposición en bloques funcionalmente independientes del código original, CatapultC como herramienta de síntesis de alto nivel, logra resultados adecuados tanto a nivel de ocupación de recursos como de frecuencia de operación. Por este motivo, se realiza una separación a nivel de bloques funcionales del código original en donde cada paso descrito en la Figura 5.9 corresponde a una llamada a una función independiente.

Las memorias son recursos limitados en los dispositivos disponibles para ser empleados en el entorno espacial. Por lo tanto, resulta fundamental minimizar la cantidad de almacenamiento requerida por los algoritmos de compresión de imágenes hiperespectrales que tengan como objetivo ser implementados en estos dispositivos. La consecuencia directa, dada la escasa cantidad de memorias empleadas, es que es necesaria la gestión, control y reducción de los accesos de escritura y lectura a realizar. A nivel arquitectural, se ha modificado el código fuente del algoritmo original con el fin de incluir la estrategia para el almacenamiento de la muestra actual y de la vecindad descrita en [27]. Esta estrategia emplea cinco (5) colas de tipo FIFO (del inglés, *First-In, First-Out*) para el almacenamiento de las muestras correspondientes a la muestra actual y las muestras vecinas a la muestra actual situadas en las posiciones: superior-izquierda, superior, superior-

derecha e izquierda. Para cada una de las muestras vecinas y para la muestra actual se almacena tanto la muestra en la banda espectral que se está procesando como las muestras contenidas en las P bandas anteriores. Mediante esta estrategia se reduce significativamente la cantidad de accesos a memoria siendo únicamente necesaria la lectura de la muestra actual y la muestra de la vecina superior-derecha en cada ciclo de reloj [27].

En una implementación hardware, el cálculo de divisiones resulta altamente agresivo en cuanto a consumo de recursos. Por este motivo, se han identificado estos operadores dentro del algoritmo en el código original y se han sustituido por desplazadores lógicos cuando ha sido posible, es decir, cuando el dividendo es divisible por una potencia de dos (2). Un ejemplo de cómo realizar la sustitución de divisiones por potencias de dos de números enteros con signo y sin signo mediante dezplazadores lógicos se presenta en la Figura 5.10.

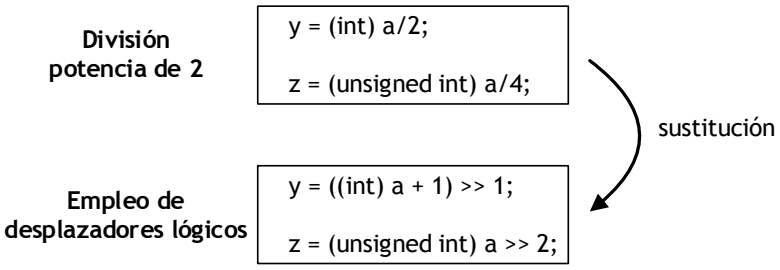


Figura 5.10 Ejemplo de sustitución de divisores potencias de dos mediante desplazadores lógicos

Para aquellos casos en que el divisor no sea potencia de dos se ha incluido la función *div()* disponible dentro de la librería *math.h* proporcionada por CatapultC como alternativa al operador divisor ("/") clásico de C/C++. Esta función se encuentra optimizada para ser implementada en hardware implementando el algoritmo de la división mediante bucles. La función *div()* toma dos variables de entrada (dividendo y divisor) y computa el resultado de la división (cociente). Además, la función garantiza su estabilidad saturando el resultado de la división en el caso de que el divisor sea cero.

Se ha mantenido el operador divisor clásico de C/C++ y la llamada a la función *div()* en el código fuente final. Como se presenta posteriormente en

la sección de resultados, el propósito de mantener ambas llamadas es poder comparar los resultados de síntesis en la implementación hardware con el fin de evaluar la mejora de prestaciones mediante el empleo de la función *div()*. Asimismo, se ha verificado que, para todas las imágenes empleadas en la verificación funcional, los resultados a son idénticos entre la utilización del operador "/" y la función *div()*.

5.3.1.1 Inclusión del algoritmo ligero de control de la tasa de bits

El algoritmo original implementa para el control de la tasa de bits el algoritmo descrito en la sección 5.2.2. En ella, se describen las principales limitaciones a la hora de ser implementado en un hardware capaz de operar a bordo de un satélite. Por este motivo, se ha decidido incluir el algoritmo ligero de control de la tasa de bits a nivel C/C++ para obtener su posterior implementación mediante CatapultC.

La inclusión de la LUT de tasas de bits en el algoritmo supone el primer paso en las modificaciones introducidas sobre el código original. Para ello, mediante la ecuación (5.16) se generan los valores de la tasa de bit para cada uno de los pasos de cuantificación Q, con $Q \in [1, 3, 5, \dots, 511]$, y para cada uno de los valores de medianas m en el rango de valores enteros $[0, 1023]$. A nivel de codificación, se incluyen los valores de las tasas de bit generados mediante una variable compuesta por un vector de números enteros de 16 bit de tamaño 256 x 1024. La indexación de cada uno de los elementos del vector que implementa la LUT de tasas de bits se define mediante la expresión (5.17).

$$LUT_{index} = \left(\frac{Q-1}{2}\right) \cdot 2^{10} + m \tag{5.17}$$

Además de la LUT de tasas de bits, es necesario la introducción en el algoritmo el cómputo de las diferentes medianas calculadas para los píxeles de tipo A, tipo B y tipo C. En el cálculo de las medianas se han empleado dos funciones distintas: *LWRC_Rows()* y *LWRC_Bands()*.

La función *LWRC_Rows()* almacena localmente cada uno de los L-1 píxeles de tipo A (residuos no-cuantificados de la muestra actual) calculando su mediana una vez se alcanza un píxel de tipo B. El tamaño de la variable L determina el número de residuos a almacenar y por lo tanto la cantidad de almacenamiento necesaria. Así pues, la función *LWRC_Rows()* requiere de $(L\ Residuos) \cdot (Tamaño\ Residuo\ bits)$ de almacenamiento para su correcta ejecución. Como se presenta en la sección 5.2.3, se obtiene un buen compromiso de prestaciones para valores de la variable L próximos a 17 píxeles siendo 16 bits el rango dinámico máximo de los residuos. En la implementación realizada se ha escogido este valor con lo que los residuos pueden almacenarse localmente en registros permitiendo una implementación más eficiente si se compara con una implementación en la que se empleen memorias. Esto es debido a que el acceso a un valor de un registro emplea únicamente el índice para su acceso, mientras que el acceso a una memoria requiere de un bus de direcciones e introduce un ciclo de latencia en cada acceso, ya sea de lectura o de escritura. La Figura 5.11 muestra el pseudocódigo de la función *LWRC_Rows()* implementada.

```
1: LWRC_Rows()
2:     newResidual = ŝ_{z,y,x} − s_{z,y,x}
3:     accResidual[] = newResidual
4:     if ((x % cLSIZE) == (cLSIZE - 1))
5:         newMedian = computeMedian(accResidual, cLSIZE);
6:         accMedians[] = newMedian;
7:     end if
8: end LWRC_Rows
```

Figura 5.11 Pseudocódigo de la función LWRC_Rows()

Para el cálculo de las medianas de cada una de las bandas espectrales se emplea la función *LWRC_Bands()*. Esta función se ejecuta cada vez que se llega al último píxel de cada banda, es decir, cada vez que $x = N_x$. En este caso encontramos un píxel de tipo C, por lo tanto, se realiza el cálculo de la mediana de medianas para la banda z en cuestión. Esta mediana de medianas es almacenada en el vector m_z cuyo tamaño depende del número de bandas espectrales de la imagen de entrada. De este modo, el almacenamiento requerido para el vector m_z es de $N_z * (Tamaño\ Residuos\ bits)$, siendo el

rango dinámico máximo de los residuos de 16 bits. El pseudocódigo que implementa esta función se describe en la Figura 5.12.

```
1: LWRC_Bands()
2:     medianOfMedians = medianRC(accMedians, numberMedians);
3:     zMedians[z] = medianOfMedians;
4: end LWRC_Bands
```

Figura 5.12 Pseudocódigo de la función LWRC_Bands()

Para la obtener el siguiente paso de cuantificación, $Q_{siguiente}$, que proporciona la tasa de bits deseada en la siguiente fila de píxeles a comprimir, se calcula la tasa de bits empleada en la compresión de la fila de píxeles actual en el momento que se alcanza un píxel de tipo D. Para tal fin, se emplea el vector de mediana de medianas para cada banda m_z y el paso de cuantificación actual Q. A partir de estos datos, se busca el valor de cada una de las tasas de bits en la LUT de tasas de bits indexando ambos valores, siguiendo la ecuación (5.17), para cada una de las bandas espectrales. Este funcionamiento, ha sido implementado sobre el código original mediante el pseudocódigo presentado en la Figura 5.13. Nótese que remplaza el código presentado en la línea 13 de la Figura 5.9, es decir, la llamada a la función *rateControl()*.

```
1: if (RCmode == RC_MODE_A)
2:     nextQy = LWRC_Mode_A (zMedians, targetbpp, rateLUT, quantStep);
3: else if (RCmode == RC_MODE_B)
4:     nextQy = LWRC_Mode_B (zMedians, targetbpp, written_bytes, rateLUT,
                             quantStep, currWrittenBytes, globalCorrection,
                             localCorrection, newTargetbpp)
5: end if
```

Figura 5.13 Pseudocódigo de las llamadas a las funciones LWRC_Mod_A() y LWRC_Mode_B()

En la Figura 5.13, se observa que, para el cálculo de la tasa de bits usada, se pueden emplear los siguientes modos:

- **Modo A:** En este modo se establece como tasa de bits usada la tasa de bits objetivo del algoritmo. De esta forma, se asume

que la tasa de bits obtenida en la fila de píxeles procesados con anterioridad no tiene desviaciones con respecto a la tasa de bits establecida como objetivo por el usuario. Este modo, si bien resulta sencillo cara a su implementación hardware, no es capaz de detectar desviaciones de la tasa de bits de cada fila procesada lo que penaliza su exactitud.

- **Modo B:** La tasa de bits usada en este modo se obtiene restando la cantidad de bpp (bits por píxel) necesarios para la codificación entrópica de la fila de píxeles actual al valor de la tasa de bits objetivo definida por el usuario. Por lo tanto, se realimenta la salida del codificador entrópico hasta el módulo que realiza el control de la tasa de bits. La tasa de bits objetivo es adaptada dinámicamente en mediante dos correcciones. La corrección global en la que se calcula la nueva tasa de bits objetivo teniendo en cuenta la cantidad bits restantes para toda la imagen y los empleados en la fila de píxeles actual y la corrección local, en donde se modifica la corrección global para o bien usar la cantidad de bpp disponible no empleada en filas de píxeles anteriores o bien corregir el exceso de bpp empleado en filas de píxeles anteriores. Dada la naturaleza adaptativa de este modo, para su correcto funcionamiento es necesario definir tanto el valor máximo como el mínimo de la tasa de bits objetivo en cada fila de píxeles. El valor máximo corresponde a la realización de la compresión sin pérdidas del algoritmo, mientras que el valor mínimo de la tasa de bits establece la de distorsión máxima tolerable de la imagen. En la implementación realizada se escoge un valor mínimo de tasa de bits de 0.01 bpp ya que se pretende evaluar el correcto funcionamiento del control de la tasa de bits.

En [81] se demuestra que el Modo B alcanza mejores resultados en cuanto a la exactitud entre la tasa de bits objetivo y la tasa de bits real que el Modo A si bien presenta una mayor complejidad. En cuanto a la precisión

del control de la tasa de bits, ambos modos muestran resultados similares [81].

Las modificaciones descritas sobre el código original proporcionan un nuevo algoritmo cuyo pseudocódigo se muestra en la Figura 5.14. En ella, se resaltan las líneas en las que existen modificaciones con respecto al pseudocódigo del algoritmo original de la función *predict()* presentado en la Figura 5.9.

```
1: for (y = 0; y < Ny; y++)
2:     for (z = 0; z < Nz; z++)
3:         for (x = 0; x < Nx; x++)
4:             fullPredictor(s_{z,y,x}, Δ_{z,y,x})
5:             LWRC_Rows()
6:             rangeEncoder(Δ_{z,y,x})
7:         end x
8:         LWRC_Bands()
9:     end z
10:    if (RCmode == RC_MODE_A)
11:        nextQy = LWRC_Mode_A ()
12:    else if (RCmode == RC_MODE_B)
13:        nextQy = LWRC_Mode_B ()
14:    end if
15: end y
```

Figura 5.14 Pseudocódigo de la función predict() tras la inclusión del algoritmo ligero de control de la tasa de bits

5.3.1.2 Simplificación del algoritmo ligero de control de la tasa de bits

Como puede desprenderse del control de la tasa de bits mediante el Modo B descrito en la sección 5.3.1.1, este modo conlleva una elevada carga computacional comparada con el Modo A. Por el contrario, el error obtenido entre la tasa de bits objetivo y la real resulta menor gracias a su comportamiento adaptativo. Para su operación, el Modo B emplea tres (3) divisores para obtener: el número de bpp, empleado en la fila de píxeles actual; la corrección global y; finalmente, la corrección local.

Con el fin de simplificar la realimentación desde la salida del codificador entrópico se propone una implementación simplificada del Modo B, esta modificación ha sido denominada Modo B simplificado, en adelante Modo BS. Este modo reduce el número de divisiones empleadas para obtener la tasa de bits usada, que es una entrada necesaria para obtener el siguiente escalón de cuantificación. En este nuevo modo, se calcula el número de bpp empleado en la codificación de la fila de píxeles actual tal y como se hace en el Modo B. Sin embargo, se realiza únicamente la corrección local de la tasa de bits en la que se toma como base la tasa de bits objetivo definido por el usuario y el exceso o ahorro de la tasa de bits empleada en la fila de píxeles anterior. Este hecho explota la alta correlación entre filas de píxeles que presenta una imagen hiperespectral corrigiendo en la siguiente fila de píxeles el error cometido en el control de la tasa de bits ya sea por exceso o por defecto.

Como se describe en detalle posteriormente en la sección 3.4, existe una penalización en términos de exactitud entre el Modo BS y el Modo B original. Este hecho es debido al empleo en la realimentación de únicamente información de la fila de píxeles anterior y no de información de la tasa de bits empleada en la totalidad de la imagen. Por el contrario, la precisión en los resultados del Modo BS resulta similar al Modo B original.

El pseudocódigo del algoritmo resultante para la función predict() se muestra en la Figura 5.15 en la que se han resaltado las líneas de pseudocódigo que presentan cambios con respecto a la Figura 5.14.

```
1: for (y = 0; y < Ny; y++)
2:     for (z = 0; z < Nz; z++)
3:         for (x = 0; x < Nx; x++)
4:             fullPredictor(s_{z,y,x}, Δ_{z,y,x})
5:             LWRC_Rows()
6:             rangeEncoder(Δ_{z,y,x})
7:         end x
8:         LWRC_Bands()
9:     end z
10:    if (RCmode == RC_MODE_A)
11:        nextQy = LWRC_Mode_A ()
12:   else if (RCmode == RC_MODE_B)
13:       nextQy = LWRC_Mode_B ()
12:   else if (RCmode == RC_MODE_BS)
13:       nextQy = LWRC_Mode_BS ()
14:   end if
15: end y
```

Figura 5.15 Pseudocódigo de la función predict() tras la inclusión del algoritmo ligero de control de la tasa de bits y el modo BS

5.3.1.3 Inclusión del codificador entrópico adaptativo por muestras CCSDS 123.0-B-1

El algoritmo original propone un codificador de rango en la etapa de codificación entrópica de los residuos de predicción cuantificados [81]. Como se describe en la sección 5.2.2, la principal limitación del codificador de rango es que necesita de mantener simultáneamente cuatro modelos independientes de codificadores de rango para su funcionamiento. Por este hecho, se propone como alternativa realizar la codificación entrópica mediante el codificador entrópico adaptativo por muestras descrito en el estándar CCSDS 123.0-B-1.

El codificador entrópico adaptativo por muestras empleado en la implementación C/C++ forma parte de la implementación del software de referencia del estándar CCSDS 123.0 realizada por la Agencia Espacial Europea, ESA (del inglés, *European Space Agency*). El código implementa la totalidad del compresor CCSDS 123.0 por lo que ha sido necesario extraer las funciones relacionadas con la codificación entrópica adaptativa por muestras así como sus dependencias dentro del código. Una vez realizada esta

extracción, se introdujo el código perteneciente al codificador adaptativo en el presentado en la Figura 5.15. De este modo el código resultante final con el que se procede a realizar su síntesis mediante la herramienta CatapultC es el presentado en la Figura 5.16 a nivel de pseudocódigo.

```
1: for (y = 0; y < Ny; y++)
2:     for (z = 0; z < Nz; z++)
3:         for (x = 0; x < Nx; x++)
4:             fullPredictor(s_{z,y,x}, Δ_{z,y,x})
5:             LWRC_Rows()
6:             rangeEncoder(Δ_{z,y,x})
7:             encodePixel(Δ_{z,y,x})
8:         end x
9:         LWRC_Bands()
10:    end z
11:    if (RCmode == RC_MODE_A)
12:        nextQy = LWRC_Mode_A ()
13:    else if (RCmode == RC_MODE_B)
14:        nextQy = LWRC_Mode_B ()
15:    else if (RCmode == RC_MODE_BS)
16:        nextQy = LWRC_Mode_BS ()
17:    end if
18: end y
```

Figura 5.16 Pseudocódigo final de la función predict() tras la inclusión de las modificaciones realizadas

La inclusión de este codificador entrópico permite comparar funcionalmente los resultados en cuanto a eficiencia de compresión de ambos codificadores: el adaptativo por muestras y el de rango. Además, se posibilita la comparación de la implementación hardware en términos de uso de recursos y frecuencia de operación tal y como se presenta en la sección de resultados.

5.4 Evaluación de la compresión con pérdidas en los algoritmos propuestos

En esta sección se presentan los resultados que se extraen del algoritmo de compresión con pérdidas, basado en el estándar CCSDS 123.0 y con control de la tasa de bits, implementado. El algoritmo tomado como base para la implementación es el descrito, a nivel de pseudocódigo, en la Figura 5.16.

Este algoritmo, si bien se basa en el algoritmo descrito en la sección 5.2.2, presenta una serie de modificaciones cuya envergadura hace que deba considerarse como un nuevo algoritmo de compresión de imágenes hiperespectrales. Por este motivo, antes de proceder con la implementación hardware es necesario garantizar que el nuevo algoritmo produzca resultados correctos y con el PSNR adecuado.

5.4.1 Evaluación del algoritmo propuesto

Este apartado tiene como objetivo evaluar el funcionamiento del nuevo algoritmo desarrollado, así como garantizar que las modificaciones realizadas a nivel C/C++ son consistentes y producen resultados acorde a lo esperado. Las evaluaciones realizadas del algoritmo propuesto han sido llevadas a cabo con los siguientes propósitos:

- **Control de la tasa de bits:** Verificar la precisión y exactitud en cuanto al control de la tasa de bits del algoritmo propuesto comparado con la tasa de bits definida como objetivo.
- **Calidad de la imagen comprimida**: Evaluar cuantitativamente la calidad de la imagen reconstruida y comparar los diferentes resultados obtenidos del algoritmo propuesto. El indicador de calidad de imagen empleado para realizar el estudio comparativo es la relación señal a ruido de pico, PSNR (del inglés, *Peak Signal-to-Noise Ratio*). El PSNR, también denominado ratio tasa de bits – distorsión, se obtiene mediante el cociente entre error cuadrático medio, MSE (del inglés, *Mean*

Squared Error) y el valor máximo que pueda alcanzar un píxel en la imagen a comprimir, es decir, 2^b en donde b es igual al valor de bits por píxel definido en la Tabla 5.1 para cada imagen. El cálculo de los valores de MSE y PSNR se describe en las ecuaciones (5.18) y (5.19).

$$MSE = \frac{1}{N_z N_y N_x} \sum_{z=0}^{N_z-1} \sum_{y=0}^{N_y-1} \sum_{x=0}^{N_x-1} \left| s_{z,y,x} - \hat{s}_{z,y,x} \right|^2 \tag{5.18}$$

$$PSNR = 10 \cdot log_{10} \left(\frac{2^b - 1}{MSE} \right) \tag{5.19}$$

Es relevante remarcar el hecho de que resulta fundamental la evaluación del binomio control de la tasa de bits y degradación de la calidad en la imagen reconstruida con el fin de comparar los resultados obtenidos. Generalmente, cuanto mejor sea la exactitud del control de la tasa de bits y mayor la calidad de la imagen reconstruida, valores mayores de PSNR y por tanto menor distorsión en la imagen, se podrá inferir la superioridad en prestaciones de un algoritmo frente a otro. Así pues, para una misma tasa de bits objetivo, el algoritmo que proporcione valores mayores de PSNR se podrá afirmar que posee mejores prestaciones.

Se ha seleccionado el conjunto de imágenes tanto hiperespectrales (H) como multiespectrales (M), presentado en la Tabla 5.1, para la evaluación del algoritmo propuesto.

Imagen	Sensor	Tipo	Filas	Columnas	Bandas	P	Bits por píxel
SC0_RAW	AVIRIS	H	512	680	224	15	16
SC167	CRISM	H	510	640	545	3	16
MOUNTAIN	Landsat	M	1024	1024	6	5	8
GRAN9	AIRS	H	135	90	1501	10	14
Erta Ale	HYPERION	H	3187	256	242	15	12

Tabla 5.1 Cuerpo de imágenes empleadas en la verificación funcional del algoritmo de compresión

Se han escogido un total de cinco imágenes que se consideran representativas para la extracción de resultados. El cuerpo de imágenes seleccionado engloba imágenes de los siguientes sensores: AVIRIS, CRISM, Landsat, AIRS e Hyperion. Esto garantiza la independencia de las pruebas realizadas de las características del sensor empleado como generador de imagen de entrada.

5.4.1.1 Resultados del control de la tasa de bits

De la Tabla 5.2 a la Tabla 5.6, se presentan los resultados obtenidos en cuanto a exactitud y precisión del control de la tasa de bits para las diferentes alternativas posibles del algoritmo propuesto. De este modo se extraen resultados para el algoritmo basado en predictor y codificador de rango, en adelante PCR, y para el algoritmo compuesto por el predictor y el codificador adaptativo por muestras, en adelante PCAM.

Los resultados se han obtenido teniendo en cuenta diferentes tasas de bit objetivo (2 bpp, 3 bpp y 4 bpp) en la compresión casi-sin pérdidas de todo el conjunto de imágenes presentado en la Tabla 5.1. Además, se han evaluado los diferentes modos de funcionamiento (Modo A, Modo B y Modo BS) y los diferentes codificadores entrópicos implementados (codificador de rango y codificador adaptativo por muestras). Nótese que se han estimulado ambos algoritmos, PCR y PCAM, con idénticas tasas de bit objetivo con el fin de

comparar los resultados obtenidos en cuanto a exactitud y precisión en el control de la tasa de bits.

En los resultados presentados en las siguientes tablas se muestra, en columnas, las tasas de bit objetivo evaluadas, el modo de control de tasa de bit empleado, el valor de la tasa de bit obtenido y la diferencia, en bpp, entre la tasa de bit objetivo y la obtenida (columna error). Un error negativo indica que se ha obtenido una tasa de bit menor de la requerida, mientras que, un error positivo indica que la tasa de bit obtenida supera la definida como objetivo.

IMAGEN: AVIRIS SC0_RAW (512x680x224)					
Tasas de bits objetivo	**Modo**	**PCR**		**PCAM**	
		Tasa de bits (bpp)	**Error (bpp)**	**Tasa de bits (bpp)**	**Error (bpp)**
2 bpp	**A**	2,257	0,257	2,268	0,268
	B	2,002	0,002	2,003	0,003
	BS	2,010	0,010	2,011	0,011
3 bpp	**A**	2,478	-0,522	2,475	-0,525
	B	3,001	0,001	3,001	0,001
	BS	2,999	-0,001	2,999	-0,001
4 bpp	**A**	2,682	-1,318	2,632	-1,368
	B	4,020	0,020	4,011	0,011
	BS	3,978	-0,022	3,977	-0,023

Tabla 5.2 Resultados de exactitud en el control de la tasa de bits para la imagen AVIRIS SC0_RAW

IMAGEN: CRISM SC167 (510x640x545)					
Tasas de bits objetivo	Modo	PCR		PCAM	
		Tasa de bits (bpp)	Error (bpp)	Tasa de bits (bpp)	Error (bpp)
2 bpp	A	2,862	0,862	2,767	0,767
	B	2,006	0,006	2,041	0,041
	BS	2,207	0,207	2,052	0,052
3 bpp	A	3,205	0,205	3,107	0,107
	B	3,189	0,189	3,031	0,031
	BS	3,175	0,175	3,018	0,018
4 bpp	A	3,508	-0,492	3,397	-0,603
	B	4,157	0,157	4,028	0,028
	BS	4,123	0,123	3,992	-0,008

Tabla 5.3 Resultados de exactitud en el control de la tasa de bits para la imagen CRISM SC167

IMAGEN: Landsat MOUNTAIN (1024x1024x6)					
Tasas de bits objetivo	Modo	PCR		PCAM	
		Tasa de bits (bpp)	Error (bpp)	Tasa de bits (bpp)	Error (bpp)
2 bpp	A	1,609	-0,391	1,517	-0,483
	B	2,005	0,005	2,001	0,001
	BS	2,004	0,004	1,999	-0,001
3 bpp	A	2,003	-0,997	1,725	-1,275
	B	3,005	0,005	3,000	0,000
	BS	3,001	0,001	2,996	-0,004
4 bpp	A	2,259	-1,741	2,021	-1,979
	B	3,901	-0,099	3,968	-0,032
	BS	3,931	-0,069	3,986	-0,014

Tabla 5.4 Resultados de exactitud en el control de la tasa de bits para la imagen Landsat MOUNTAIN

IMAGEN: AIRS GRAN9 (135x90x1501)					
Tasas de bits objetivo	Modo	PCR		PCAM	
		Tasa de bits (bpp)	Error (bpp)	Tasa de bits (bpp)	Error (bpp)
2 bpp	A	2,126	0,126	2,343	0,343
	B	1,963	-0,037	2,050	0,050
	BS	2,047	0,047	2,091	0,091
3 bpp	A	2,469	-0,531	2,717	-0,283
	B	2,942	-0,058	3,000	0,000
	BS	2,980	-0,020	3,145	0,145
4 bpp	A	2,733	-1,267	2,976	-1,024
	B	3,922	-0,078	4,000	0,000
	BS	3,909	-0,091	4,075	0,075

Tabla 5.5 Resultados de exactitud en el control de la tasa de bits para la imagen AIRS GRAN9

IMAGEN: HYPERION Erta Ale (3187x256x242)					
Tasas de bits objetivo	Modo	PCR		PCAM	
		Tasa de bits (bpp)	Error (bpp)	Tasa de bits (bpp)	Error (bpp)
2 bpp	A	3,410	1,410	3,117	1,117
	B	2,010	0,010	2,004	0,004
	BS	2,018	0,018	2,006	0,006
3 bpp	A	3,861	0,861	3,544	0,544
	B	3,007	0,007	3,004	0,004
	BS	3,007	0,007	3,002	0,002
4 bpp	A	4,156	0,156	3,815	-0,185
	B	4,006	0,006	4,002	0,002
	BS	4,006	0,006	4,001	0,001

Tabla 5.6 Resultados de exactitud en el control de la tasa de bits para la imagen HYPERION Erta Ale

El algoritmo de compresión con control de la tasa de bits mediante el modo A obtiene peores resultados en cuanto a exactitud y precisión en el control de la tasa de bits que los algoritmos que emplean el modo B o el modo

BS. Este hecho se debe a que, al no existir una realimentación desde el codificador entrópico que indique el ahorro (o exceso) en bpp para la fila de píxeles procesada, no es posible emplear este ahorro (o exceso) en la compresión de la fila de píxeles siguiente. Se puede concluir que, el algoritmo con control de la tasa de bits mediante el modo A no resulta apropiado cuando se requieren altas tasas de bits, aquellas proporcionadas por la compresión sin pérdidas, ya que presenta una mayor complejidad y pobres resultados en el control de la tasa de bits.

Por otra parte, el algoritmo de compresión con control de la tasa de bits mediante el modo BS alcanza valores de exactitud y precisión en términos de bpp para la tasa de bits similares a los alcanzados por el algoritmo de compresión con control de la tasa de bits mediante el modo B original. Se ha observado una diferencia máxima absoluta de 0,202 bpp entre el modo B y el modo BS en la compresión mediante el algoritmo PCR en la imagen SC167 del sensor CRISM. Este hecho confirma que, a pesar de la reducción en cuanto a su complejidad computacional, el comportamiento del modo BS como método de control de la tasa de bits es comparable al comportamiento del modo B original.

5.4.1.2 Resultados de la ratio tasa de bits - distorsión

La calidad de la imagen reconstruida ha sido calculada mediante el indicador PSNR para cada una de las imágenes presentadas en la Tabla 5.1. Los resultados se han obtenido bajo los mismos estímulos que los presentados en la sección anterior, es decir, se han considerado las diferentes tasas de bit objeto, los diferentes modos de funcionamiento (Modo A, Modo B y Modo BS) y los diferentes codificadores entrópicos implementados (codificador de rango y adaptativo por muestras). De este modo, se han extraído los resultados presentados desde la Tabla 5.7 a la Tabla 5.11.

IMAGEN: AVIRIS SC0_RAW (512x680x224)			
Tasas de bits objetivo	Modo	PCR	PCAM
		PSNR (dB)	PSNR (dB)
2 bpp	A	80,610	80,610
	B	76,428	73,762
	BS	78,448	76,571
3 bpp	A	82,218	82,218
	B	85,441	85,533
	BS	84,673	84,618
4 bpp	A	83,358	83,358
	B	91,708	91,823
	BS	88,604	88,589

Tabla 5.7 Resultados de calidad de imagen reconstruida (PSNR) en el control de la tasa de bits para la imagen AVIRIS SC0_RAW

IMAGEN: CRISM SC167 (510x640x545)			
Tasas de bits objetivo	Modo	PCR	PCAM
		PSNR (dB)	PSNR (dB)
2 bpp	A	80,917	80,917
	B	75,344	76,742
	BS	74,869	74,625
3 bpp	A	83,172	83,172
	B	81,407	79,413
	BS	82,208	81,200
4 bpp	A	85,099	85,099
	B	88,807	88,067
	BS	88,736	87,894

Tabla 5.8 Resultados de calidad de imagen reconstruida (PSNR) en el control de la tasa de bits para la imagen CRISM SC167

IMAGEN: Landsat MOUNTAIN (1024x1024x6)			
Tasas de bits objetivo	Modo	PCR	PCAM
		PSNR (dB)	PSNR (dB)
2 bpp	A	42,723	42,723
	B	45,253	46,215
	BS	45,295	46,206
3 bpp	A	45,275	45,275
	B	51,197	52,451
	BS	51,379	52,478
4 bpp	A	47,011	47,011
	B	56,342	53,624
	BS	58,504	55,786

Tabla 5.9 Resultados de calidad de imagen reconstruida (PSNR) en el control de la tasa de bits para la imagen Landsat MOUNTAIN

IMAGEN: AIRS GRAN9 (135x90x1501)			
Tasas de bits objetivo	Modo	PCR	PCAM
		PSNR (dB)	PSNR (dB)
2 bpp	A	45,090	45,090
	B	43,407	42,122
	BS	43,093	42,604
3 bpp	A	47,661	47,661
	B	49,533	47,642
	BS	50,432	49,825
4 bpp	A	49,741	49,741
	B	53,284	51,259
	BS	53,991	53,123

Tabla 5.10 Resultados de calidad de imagen reconstruida (PSNR) en el control de la tasa de bits para la imagen AIRS GRAN9

IMAGEN: HYPERION Erta Ale (3187x256x242)			
Tasas de bits objetivo	Modo	PCR	PCAM
		PSNR (dB)	PSNR (dB)
2 bpp	A	19,774	19,774
	B	18,208	18,072
	BS	17,828	17,602
3 bpp	A	20,137	20,137
	B	19,281	19,678
	BS	19,476	19,599
4 bpp	A	20,416	20,416
	B	20,266	21,115
	BS	20,446	20,949

Tabla 5.11 Resultados de calidad de imagen reconstruida (PSNR) en el control de la tasa de bits para la imagen HYPERION Erta Ale

Es destacable el hecho que, en el Modo A, la falta de realimentación desde la etapa de codificación entrópica produce que los resultados en cuanto a la calidad en la imagen reconstruida sean idénticos entre los algoritmos PCR y PCAM. A nivel general, se observa que el codificador de rango proporciona resultados ligeramente superiores al codificador entrópico adaptativo por muestras en cuanto a la calidad de la imagen reconstruida. Por el contrario, el codificador entrópico adaptativo proporciona una menor complejidad en su implementación.

Los resultados de las Tabla 5.7 a Tabla 5.11 pueden ser combinados con los presentados de la Tabla 5.2 a la Tabla 5.6 para obtener una representación gráfica de la ratio tasa de bits – distorsión del cada algoritmo en sus distintas implementaciones para cada una de las imágenes del conjunto seleccionado. De la Figura 5.17 a la Figura 5.21 se presentan gráficamente los resultados donde, en el eje de abscisas, se muestran las tasas de bits, en unidades de bpp, mientras que, en el eje de ordenadas, se representan los valores de PSNR, en dB, obtenidos en cada imagen para cada una de las configuraciones de cada algoritmo presentadas.

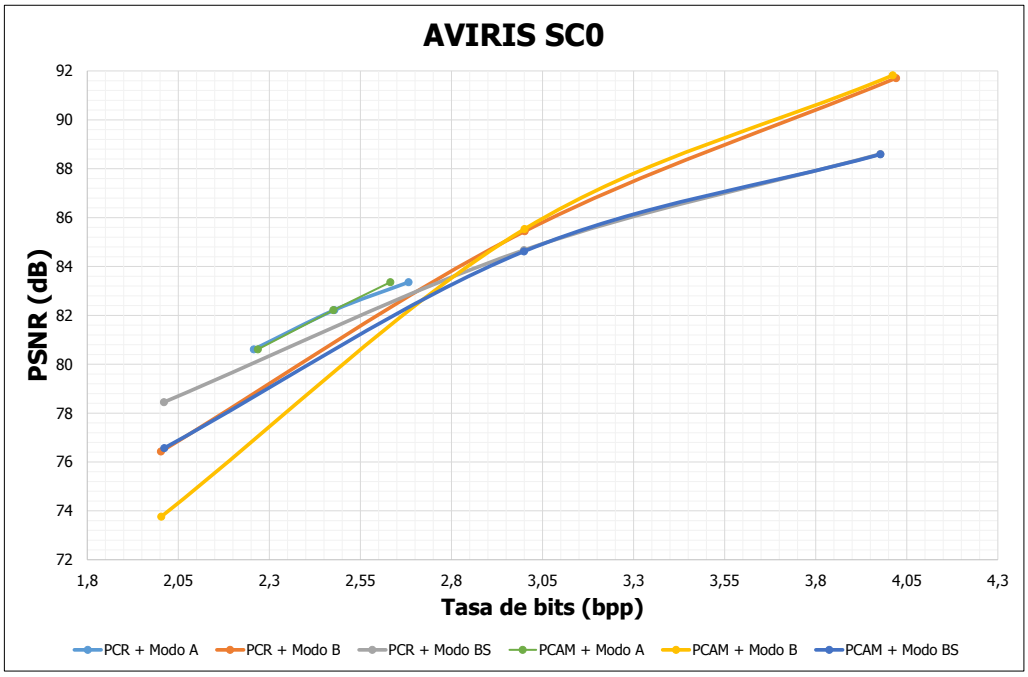


Figura 5.17 Resultados ratio tasa de bits - distorsión para la imagen AVIRIS SC0

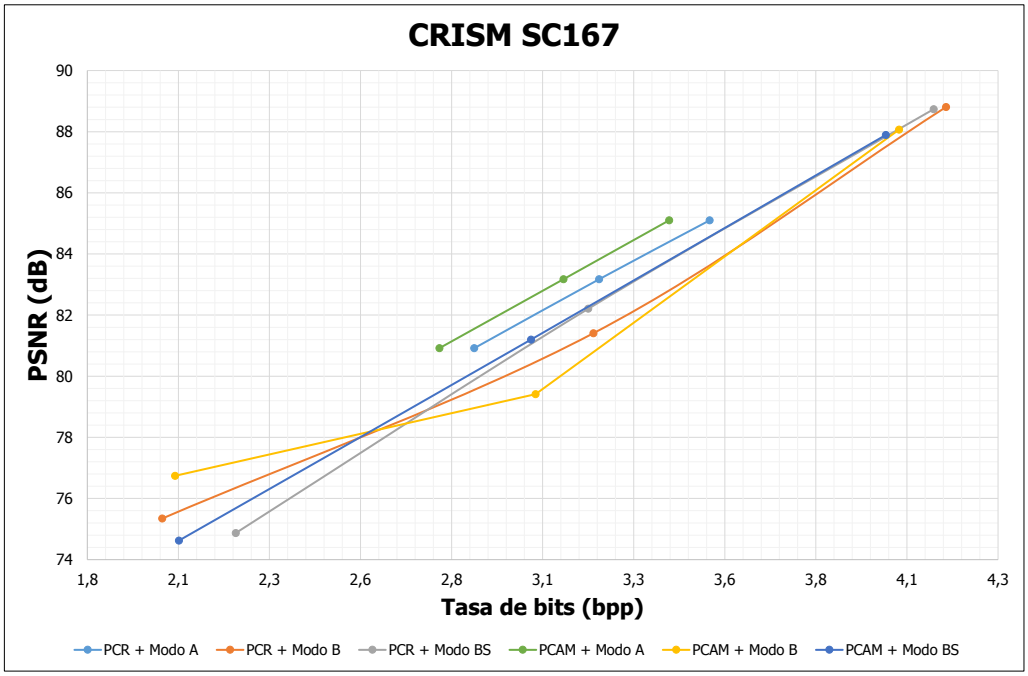


Figura 5.18 Resultados ratio tasa de bits - distorsión para la imagen CRISM SC167

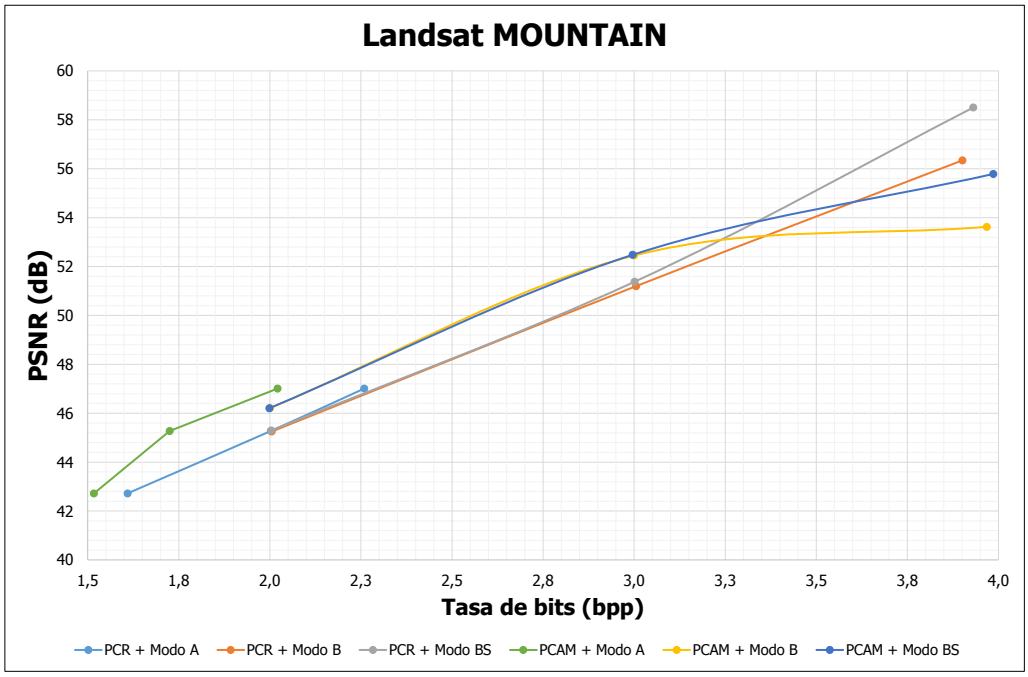


Figura 5.19 Resultados ratio tasa de bits - distorsión para la imagen Landsat MOUNTAIN

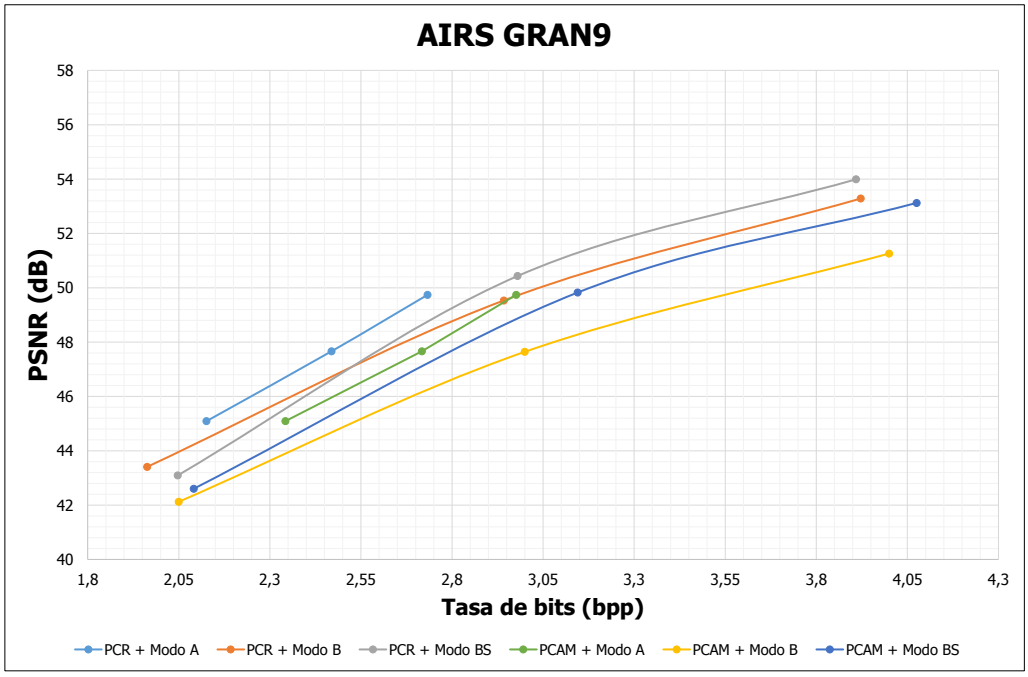


Figura 5.20 Resultados ratio tasa de bits - distorsión para la imagen AIRS GRAN9

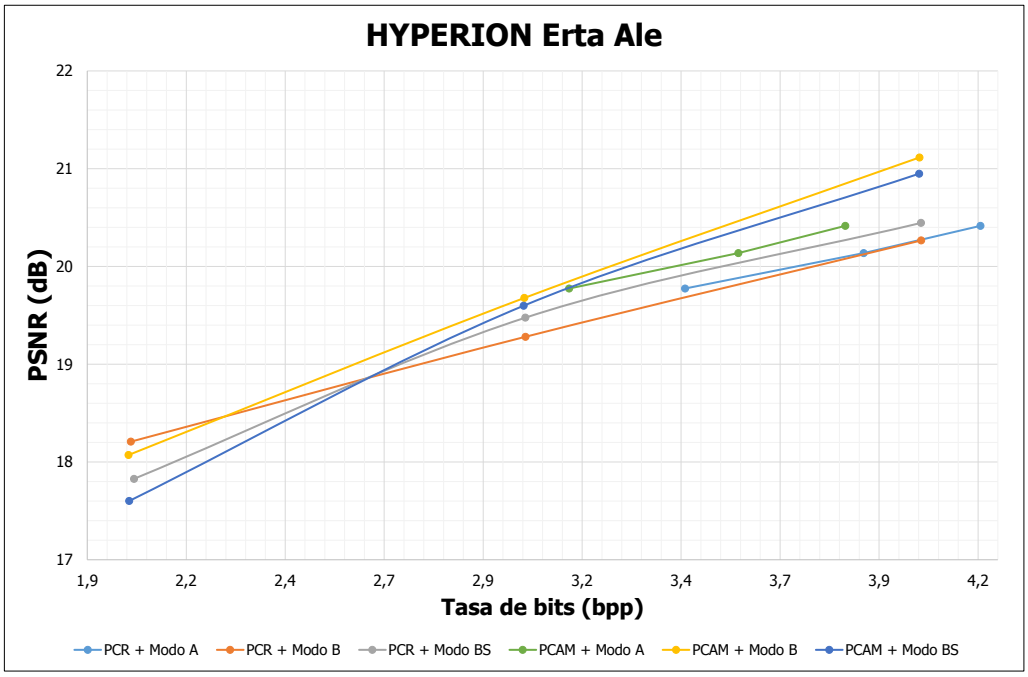


Figura 5.21 Resultados ratio tasa de bits - distorsión para la imagen HYPERION Erta Ale

Los resultados presentados desde la Figura 5.17 hasta la Figura 5.21, ponen de manifiesto las desviaciones en cuanto a la ratio tasa de bits - distorsión mediante el modo A. Especialmente, se observa que el rango de funcionamiento del control de la tasa de bits en este modo resulta muy limitado. Los modos B y BS de control de la tasa de bits implementados en el algoritmo presentan resultados similares si se realiza la comparativa de la ratio tasa de bits - distorsión. El modo BS presenta, en el peor caso, una pérdida promedio en cuanto a calidad de la imagen reconstruida de 0,617 dB comparado con el modo B original para la imagen SC0 del sensor AVIRIS. Por el contrario, el modo BS alcanza, para la imagen GRAN9 del sensor AIRS, una mejora promedio en la calidad de la imagen reconstruida de 1,51 dB si se compara con el modo B original.

5.5 Resultados obtenidos

Siguiendo la metodología de diseño que emplea la herramienta de síntesis CatapultC detallada en el capítulo 3 para la generación de código RTL, se ha realizado la síntesis de cada uno de los módulos que implementan las funciones descritas del algoritmo propuesto de compresión con pérdidas y control de la tasa de bits. El algoritmo de entrada a la herramienta es el descrito a nivel de pseudocódigo en la Figura 5.16 y evaluado en la sección 3.4.

La síntesis de los módulos diseñados ha sido realizada sobre la FPGA XQR5VFX130 de la familia Virtex-5QV del fabricante Xilinx. Esta FPGA se encuentra calificada para operar en el espacio y tiene experiencia de uso en vuelo. En la Tabla 5.12 se presentan los recursos disponibles en este dispositivo.

	Dispositivo
	XQR5VFX130
Celdas lógicas (LUTs)	81.920
Bloques de lógica configurable (CLBs)	20.480
D-Flip-Flops (DFF) o Latches	81.920
Bloques RAM (36 kb)	298
Bloques DSP	320
Bancos de Entrada/Salida	24
Entradas/Salidas disponibles para el usuario (máximo)	836

Tabla 5.12 Recursos disponibles en la FPGA Xilinx Virtex-5QV XQR5VFX130

La síntesis de cada uno de los módulos independientes se ha realizado con distintas configuraciones gracias a las modificaciones que se incluyen en el algoritmo propuesto. En este sentido, se han obtenido resultados de ocupación de recursos, frecuencia de operación y tasa de datos máxima para implementaciones hardware que emplean el operador divisor clásico de C/C++ "/", la función *div()* disponible dentro de la librería *math.h* proporcionada por CatapultC, los diferentes modos de control de la tasa de bits y para el

codificador de rango y el codificador adaptativo por muestras. Además, se ha optimizado cada bucle presente en el diseño mediante el ajuste de los valores de intervalo de iniciación (II) y el desenrollado de bucles (U). Para la síntesis del codificador entrópico adaptativo por muestras se ha empleado la implementación a nivel RTL presentada en [27] que ha sido proporcionada por el IUMA. Esta implementación ha sido específicamente realizada a nivel RTL con el objetivo de ser desplegada sobre FPGAs calificadas para espacio. Por lo tanto, su uso resulta idóneo en el contexto de la implementación propuesta en este trabajo.

Los módulos que componen el algoritmo, y que formarán parte de las implementaciones realizadas, son los siguientes:

- **Predictor sin pérdidas:** Este módulo implementa el predictor presente en el estándar CCSDS 123.0 descrito en la sección 3.2.1.1.
- **Predictor con cuantificador:** En el interior de este módulo se ha incluido un cuantificador escalar uniforme dentro del predictor del estándar CCSDS 123.0 como se describe en la sección 5.2.2.
- **Módulo *LWRC_Rows()*:** En este módulo se implementa la función que realiza el cálculo de la mediana de L-1 píxeles de tipo A como se describe en la sección 5.3.1.1.
- **Módulo *LWRC_Bands()*:** El cálculo de las medianas de cada una de las bandas espectrales descrito en la sección 5.3.1.1 se realiza en este módulo.
- **Control de la tasa de bits – Modo A:** El control de la tasa de bits en el modo A presentado en la sección 5.3.1.1 se implementa en el interior de este módulo.
- **Control de la tasa de bits – Modo B:** Este módulo implementa el Modo B de control de la tasa de bits detallado en la sección 5.3.1.1.

- **Control de la tasa de bits – Modo BS:** En este módulo se implementa el modo BS de control de la tasa de bits propuesto en este trabajo y detallado en la sección 5.3.1.2.

Los resultados obtenidos de síntesis sobre la FPGA XQR5VFX130 de cada uno de los módulos y para cada una de las configuraciones sintetizadas se presentan desde la Tabla 5.13 hasta la Tabla 5.19:

	Predictor sin pérdidas	
	Ocupación	**%**
LUTs	23839	29,10
CLBs	5960	29,10
DFF o Latches	12385	15,12
Bloques RAM	0	0,00
Bloques DSP	16	5,00
Frecuencia	82,939 MHz	
Tasa de datos	25,520 Mmuestras/s	

Tabla 5.13 Resultados de síntesis del módulo predictor sin pérdidas sobre la FPGA Xilinx Virtex-5QV XQR5VFX130

	Predictor con cuantificador – USQ (división "/")		**Predictor con cuantificador – USQ (bucle div. II=1)**		**Predictor con cuantificador – USQ (bucle div. U)**	
	Ocupación	**%**	**Ocupación**	**%**	**Ocupación**	**%**
LUTs	26425	32,26	**23938**	**29,22**	25143	30,69
CLBs	6607	32,26	**5985**	**29,22**	6286	30,69
DFF o Latches	12621	15,41	**12602**	**15,38**	13204	16,12
Bloques RAM	0	0,00	**0**	**0,00**	0	0,00
Bloques DSP	16	5,00	**16**	**5,00**	16	5,00
Frecuencia	9,749 MHz		**83,043 MHz**		79,409 MHz	
Tasa de datos	2,785 Mmuestras/s		**34,363 Mmuestras/s**		22,819 Mmuestras/s	

Tabla 5.14 Resultados de síntesis del módulo predictor con cuantificador sobre la FPGA Xilinx Virtex-5QV XQR5VFX130

	LWRC_Rows	
	Ocupación	**%**
LUTs	1815	2,22
CLBs	454	2,22
DFF o Latches	717	0,88
Bloques RAM	0	0,00
Bloques DSP	0	0,00
Frecuencia	174,642 MHz	
Tasa de datos	87,151 Mmuestras/s	

Tabla 5.15 Resultados de síntesis del módulo LWRC_Rows sobre la FPGA Xilinx Virtex-5QV XQR5VFX130

	LWRC_Bands	
	Ocupación	**%**
LUTs	391	0,48
CLBs	98	0,48
DFF o Latches	197	0,24
Bloques RAM	0	0,00
Bloques DSP	0	0,00
Frecuencia	179,662 MHz	
Tasa de datos	59,885 Mmuestras/s	

Tabla 5.16 Resultados de síntesis del módulo LWRC_Bands sobre la FPGA Xilinx Virtex-5QV XQR5VFX130

	Control tasa de bits - Modo A	
	Ocupación	**%**
LUTs	1571	1,92
CLBs	393	1,92
DFF o Latches	732	0,89
Bloques RAM	114	38,26
Bloques DSP	0	0,00
Frecuencia	136,612 MHz	
Tasa de datos	68,374 Mmuestras/s	

Tabla 5.17 Resultados de síntesis del módulo Control tasa de bits - Modo A sobre la FPGA Xilinx Virtex-5QV XQR5VFX130

	Control tasa de bits - Modo B (división "/")		Control tasa de bits - Modo B (bucle div. II=1)		Control tasa de bits - Modo B (bucle div. U)	
	Ocupación	**%**	**Ocupación**	**%**	**Ocupación**	**%**
LUTs	12698	15,50	**4510**	**5,51**	26077	31,83
CLBs	3175	15,50	**1128**	**5,51**	6520	31,84
DFF o Latches	2083	2,54	**2992**	**3,65**	13391	16,35
Bloques RAM	114	38,26	**114**	**38,26**	114	38,26
Bloques DSP	15	4,69	**14**	**4,38**	14	4,38
Frecuencia	3,944 MHz		**60,779 MHz**		55,869 MHz	
Tasa de datos	1,974 Mmuestras/s		**30,420 Mmuestras/s**		27,961 Mmuestras/s	

Tabla 5.18 Resultados de síntesis del módulo Control tasa de bits – Modo B sobre la FPGA Xilinx Virtex-5QV XQR5VFX130

	Control tasa de bits - Modo BS (división "/")		Control tasa de bits - Modo BS (bucle div. II=1)		Control tasa de bits - Modo BS (bucle div. U)	
	Ocupación	**%**	**Ocupación**	**%**	**Ocupación**	**%**
LUTs	6408	7,82	**1790**	**2,19**	3626	4,43
CLBs	1602	7,82	**448**	**2,19**	907	4,43
DFF o Latches	992	1,21	**1072**	**1,31**	1662	2,03
Bloques RAM	114	38,26	**114**	**38,26**	114	38,26
Bloques DSP	6	1,88	**5**	**1,56**	5	1,56
Frecuencia	4,997 MHz		**135,888 MHz**		90,588 MHz	
Tasa de datos	2,501 Mmuestras/s		**68,012 Mmuestras/s**		45,567 Mmuestras/s	

Tabla 5.19 Resultados de síntesis del módulo Control tasa de bits – Modo BS sobre la FPGA Xilinx Virtex-5QV XQR5VFX130

Nótese que, se resalta en negro en la tabla de resultados correspondiente, aquellas configuraciones que comparativamente presentan mejores resultados cuando se han evaluado diferentes opciones de implementación para un mismo módulo. Estas configuraciones se emplearán posteriormente en la composición de cada uno de los algoritmos de compresión que se derivan de la implementación realizada.

De forma análoga, en la Tabla 5.20 se muestran los resultados obtenidos de la síntesis de las dos alternativas en cuanto a codificación entrópica implementadas en el algoritmo:

- **Codificador de rango:** Este módulo implementa el codificador de rango que se describe en la sección 5.2.2.
- **Codificador adaptativo por muestras:** El codificador entrópico adaptativo por muestras contenido en el estándar CCSDS 123.0 se implementa en este módulo (ver sección 5.3.1.3).

	Codificador de rango		**Codificador adaptativo por muestras**	
	Ocupación	**%**	**Ocupación**	**%**
LUTs	18036	22,02	1527	1,86
CLBs	4512	22,03	381	1,86
DFF o Latches	8480	10,35	473	0,58
Bloques RAM	0	0,00	0	0,00
Bloques DSP	24	7,50	0	0,00
Frecuencia	57,016 MHz		134,9 MHz	
Tasa de datos	42,773 Mmuestras/s		134,9 Mmuestras/s	

Tabla 5.20 Resultados de síntesis a nivel de módulo sobre la FPGA Xilinx Virtex-5QV XQR5VFX130 de los codificadores entrópicos implementados

De los resultados presentados, se extrae que la implementación del operador divisor mediante la función *div()* obtiene mejores resultados en cuanto a ocupación de recursos, frecuencia de funcionamiento y tasa de datos si es comparado con el operador divisor "/". Asimismo, en la Tabla 5.20, se constata que los recursos empleados para la implementación de los cuatro modelos necesarios en el codificador de rango son claramente superiores a los empleados por el codificador adaptativo por muestras. Este hecho confirma lo expuesto en la sección 5.2.2, en donde se presenta al codificador de rango como aproximación sub-óptima de codificación entrópica si es comparado con los codificadores entrópicos propuestos en el estándar CCSDS 123.0 en cuanto a su implementación hardware dado que un único modelo no es capaz procesar las diferentes estadísticas de los residuos predichos en cada banda.

Por otra parte, se observa, en la Tabla 5.13, que el módulo limitante en cuanto a prestaciones dentro de los módulos sintetizados es el predictor. Esto se debe a que en su interior se realiza el cálculo de la muestra predicha

escalada, en el que, es necesaria la multiplicación vectorial del vector de diferencias locales y el vector de pesos, tal y como se describe en la ecuación (5.9). Realizar una multiplicación vectorial es una operación con una alta carga computacional, por lo tanto, los resultados obtenidos ponen de manifiesto esta complejidad de forma que la frecuencia de operación máxima del algoritmo queda limitada a 82,939 MHz con una tasa de datos de 25,52 Mmuestras/s.

Mediante los datos de síntesis a nivel de módulo, pueden obtenerse diferentes implementaciones hardware que a su vez se derivan en diferentes algoritmos de compresión. De este modo, es posible, por ejemplo, obtener resultados de implementación hardware para el predictor sin pérdidas CCSDS 123.0 acompañado del codificador de rango (PCR) o acompañado del codificador adaptativo por muestras (PCAM). Los resultados en cuanto a ocupación de recursos, frecuencia máxima de operación y tasa de datos, se presentan desde la Tabla 5.21 hasta la Tabla 5.25. En ella, se muestran los datos para las siguientes configuraciones que pueden obtenerse mediante los módulos sintetizados:

- **PCR sin pérdidas:** Predictor sin pérdidas CCSDS 123.0 con etapa de codificación entrópica basada en el codificador de rango.
- **PCAM sin pérdidas:** Predictor sin pérdidas CCSDS 123.0 con etapa de codificación entrópica basada en el codificador adaptativo por muestras.
- **PCR con pérdidas (USQ):** Predictor CCSDS 123.0 con cuantificador escalar uniforme con etapa de codificación entrópica basada en el codificador de rango.
- **PCAM con pérdidas (USQ):** Predictor CCSDS 123.0 con cuantificador escalar uniforme con etapa de codificación entrópica basada en el codificador adaptativo por muestras.
- **PCR con pérdidas (USQ) y control de la tasa de bits (Modo A):** Predictor CCSDS 123.0 con cuantificador escalar uniforme y control de la tasa de bits en Modo A y codificador de rango.

- **PCAM con pérdidas (USQ) y control de la tasa de bits (Modo A):** Predictor CCSDS 123.0 con cuantificador escalar uniforme y control de la tasa de bits en Modo A y codificador adaptativo por muestras.
- **PCR con pérdidas (USQ) y control de la tasa de bits (Modo B):** Predictor CCSDS 123.0 con cuantificador escalar uniforme y control de la tasa de bits en Modo B y codificador de rango.
- **PCAM con pérdidas (USQ) y control de la tasa de bits (Modo B):** Predictor CCSDS 123.0 con cuantificador escalar uniforme y control de la tasa de bits en Modo B y codificador adaptativo por muestras.
- **PCR con pérdidas (USQ) y control de la tasa de bits (Modo BS):** Predictor CCSDS 123.0 con cuantificador escalar uniforme y control de la tasa de bits en Modo BS y codificador de rango.
- **PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS):** Predictor CCSDS 123.0 con cuantificador escalar uniforme y control de la tasa de bits en Modo BS y codificador adaptativo por muestras.

	PCR sin pérdidas		**PCAM sin pérdidas**	
	Ocupación	**%**	**Ocupación**	**%**
LUTs	41875	51,12	25366	30,96
CLBs	10472	51,13	6341	30,96
DFF o Latches	20865	25,47	12858	15,70
Bloques RAM	0	0,00	0	0,00%
Bloques DSP	40	12,50	16	5,00%
Frecuencia	57,016 MHz		82,939 MHz	
Tasa de datos	25,520 Mmuestras/s		25,520 Mmuestras/s	

Tabla 5.21 Resultados de síntesis a nivel de módulo sobre la FPGA Xilinx Virtex-5QV XQR5VFX130 de los algoritmos PCR y PCAM sin pérdidas implementados

	PCR con pérdidas (USQ)		PCAM con pérdidas (USQ)	
	Ocupación	%	Ocupación	%
LUTs	41974	51,24	25465	31,09
CLBs	10497	51,25	6366	31,08
DFF o Latches	21082	25,73	13075	15,96
Bloques RAM	0	0,00	0	0,00
Bloques DSP	40	12,50	16	5,00
Frecuencia	57,016 MHz		83,043 MHz	
Tasa de datos	34,363 Mmuestras/s		34,363 Mmuestras/s	

Tabla 5.22 Resultados de síntesis a nivel de módulo sobre la FPGA Xilinx Virtex-5QV XQR5VFX130 de los algoritmos PCR y PCAM con pérdidas (USQ) implementados

	PCR con pérdidas (USQ) y control de la tasa de bits (Modo A)		PCAM con pérdidas (USQ) y control de la tasa de bits (Modo A)	
	Ocupación	%	Ocupación	%
LUTs	45751	55,85	29242	35,70
CLBs	11442	55,87	7311	35,70
DFF o Latches	22728	27,74	14721	17,97
Bloques RAM	114	38,26	114	38,26
Bloques DSP	40	12,50	16	5,00
Frecuencia	57,016 MHz		83,043 MHz	
Tasa de datos	34,363 Mmuestras/s		34,363 Mmuestras/s	

Tabla 5.23 Resultados de síntesis a nivel de módulo sobre la FPGA Xilinx Virtex-5QV XQR5VFX130 de los algoritmos PCR y PCAM con pérdidas (USQ) y control de la tasa de bits (Modo A) implementados

	PCR con pérdidas (USQ) y control de la tasa de bits (Modo B)		PCAM con pérdidas (USQ) y control de la tasa de bits (Modo B)	
	Ocupación	**%**	**Ocupación**	**%**
LUTs	48690	59,44	32181	39,28
CLBs	12177	59,46	8046	39,29
DFF o Latches	24988	30,50	16981	20,73
Bloques RAM	114	38,26	114	38,26
Bloques DSP	54	16,88	30	9,38
Frecuencia	57,016 MHz		60,779 MHz	
Tasa de datos	34,363 Mmuestras/s		30,420 Mmuestras/s	

Tabla 5.24 Resultados de síntesis a nivel de módulo sobre la FPGA Xilinx Virtex-5QV XQR5VFX130 de los algoritmos PCR y PCAM con pérdidas (USQ) y control de la tasa de bits (Modo B) implementados

	PCR con pérdidas (USQ) y control de la tasa de bits (Modo BS)		PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS)	
	Ocupación	**%**	**Ocupación**	**%**
LUTs	45970	56,12	29461	35,96
CLBs	11497	56,14	7366	35,97
DFF o Latches	23068	28,16	15061	18,39
Bloques RAM	114	38,26	114	38,26
Bloques DSP	45	14,06	21	6,56
Frecuencia	57,016 MHz		83,043 MHz	
Tasa de datos	34,363 Mmuestras/s		34,363 Mmuestras/s	

Tabla 5.25 Resultados de síntesis a nivel de módulo sobre la FPGA Xilinx Virtex-5QV XQR5VFX130 de los algoritmos PCR y PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) implementados

Nótese que, cada configuración sintetizada, resulta en un algoritmo de compresión de imágenes hiperespectrales completo. Por lo tanto, mediante esta aproximación se obtienen implementaciones desde algoritmos de compresión sin pérdidas, algoritmos de compresión con pérdidas con cuantificador escalar uniforme (USQ) sin control de la tasa de bits y algoritmos de compresión con pérdidas con diferentes mecanismos para el control de la tasa de bits.

De forma gráfica, en la Figura 5.22, Figura 5.23 y Figura 5.24 se representan los resultados obtenidos. Comparativamente, los resultados de implementación de los algoritmos que emplean el codificador entrópico adaptativo por muestras (PCAM) presentan menores valores de ocupación de recursos y valores mayores de frecuencia de funcionamiento que las implementaciones que emplean el codificador de rango (PCR). El valor de la tasa de datos procesada viene determinado por el módulo predictor ya que es el que presenta una mayor complejidad computacional. Por lo tanto, se observa que si se comparan las implementaciones de ambos codificadores entrópicos en cuanto a la tasa de bits los resultados son de 25,52 Mmuestras/s para el predictor sin pérdidas y de 34,42 Mmuestras/s para las implementaciones que hacen uso del predictor con pérdidas.

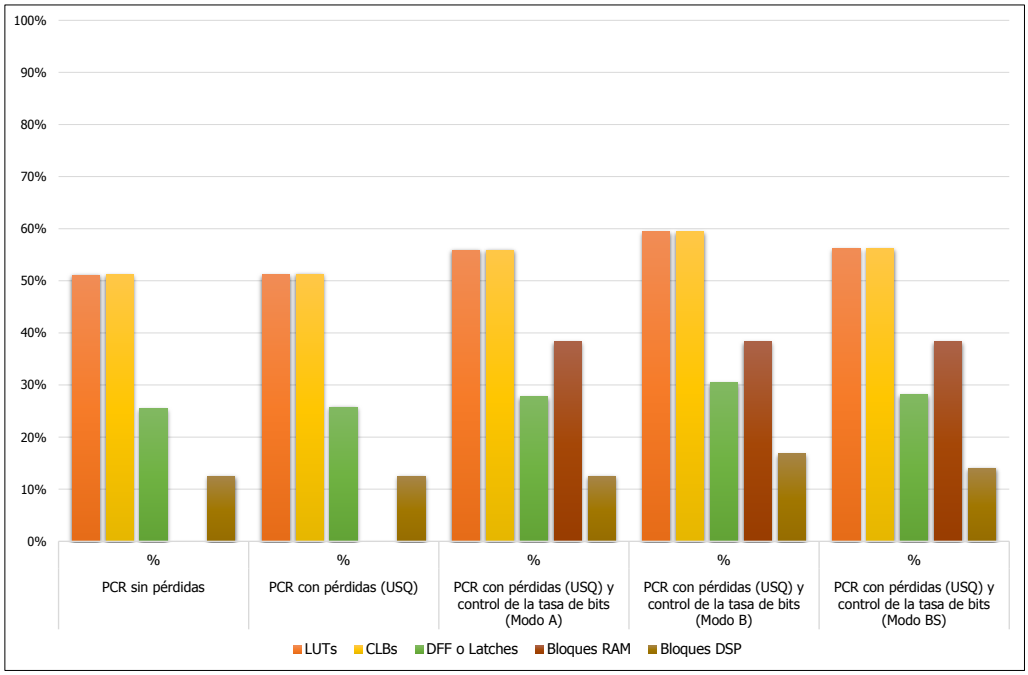


Figura 5.22 Ocupación de recursos de las implementaciones hardware de los algoritmos Predictivos con Codificador de Rango (PCR)

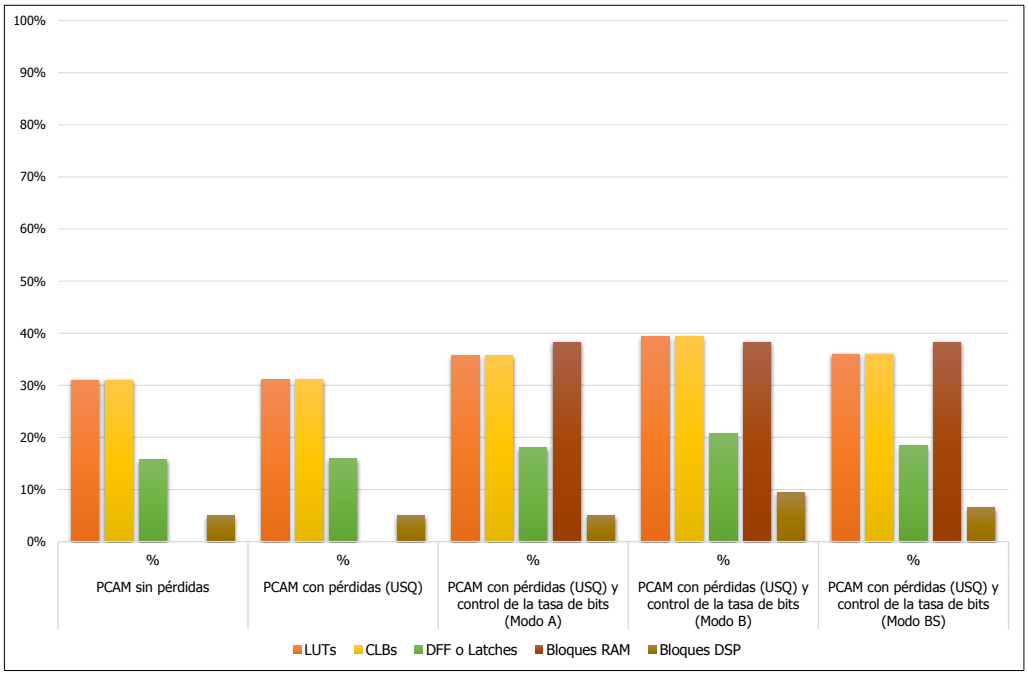


Figura 5.23 Ocupación de recursos de las implementaciones hardware de los algoritmos Predictivos con Codificador Adaptativo por Muestras (PCAM)

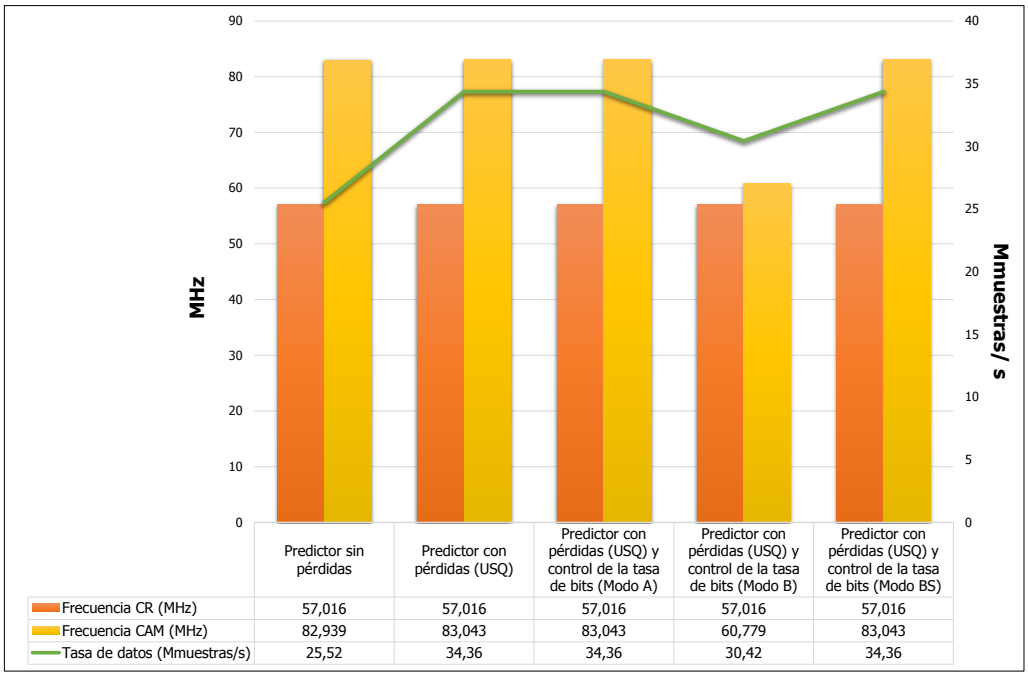


Figura 5.24 Comparativa entre el codificador de rango y el codificador adaptativo por muestras en términos de máxima frecuencia de operación y tasa de datos

Si se evalúan los resultados de las implementaciones fijando el codificador entrópico a emplear, ya sea el codificador de rango o bien el codificador adaptativo por muestras, pueden compararse las diferentes implementaciones realizadas tanto aquellas con diferentes modos de control de la tasa de bits como la implementación del algoritmo de compresión sin pérdidas. En la Figura 5.22 y en la Figura 5.23 se observa que la implementación del modo B de control de la tasa de bits resulta en la implementación más demandante en términos de recursos ocupados llegando a ocupar en términos de bloques de lógica configurable (CLB) el 59,46 % en el caso de emplear el predictor y el codificador de rango y el 37,43 % para una implementación mediante el predictor y el codificador adaptativo por muestras.

La implementación del modo BS propuesto, presenta resultados comparables a los obtenidos para el control de la tasa de bits mediante el modo A en términos de ocupación de recursos, tal y como se presenta en la Figura 5.22 y la Figura 5.23. Este hecho resulta especialmente significativo ya que, como se describe en la sección 5.4.1, las prestaciones del algoritmo empleando el modo BS resultan netamente superiores en cuanto a control de la tasa de bits de la imagen comprimida y en cuanto a la calidad de la imagen reconstruida. Dado que las prestaciones del algoritmo de compresión mediante el modo BS son comparables a las que presenta el modo B original (tal y como se observa en la evaluación realizada en la sección 3.4) y que, además, su ocupación de recursos es menor resulta una alternativa atractiva a ser utilizada en un compresor de imágenes hiperespectrales que se ejecute a bordo de un satélite.

5.6 Comparativa entre los resultados obtenidos y los disponibles en el estado del arte

En este trabajo de tesis doctoral se ha estudiado la implementación de diferentes algoritmos de compresión con pérdidas de imágenes hiperespectrales sobre dispositivos FPGA calificados para el espacio. Se ha realizado la implementación de la transformada POT (ver capítulo 4), del algoritmo LCE (ver capítulo 3) y del algoritmo basado en el estándar CCSDS 123.0 con pérdidas y control de la tasa de bits presentado en este capítulo. En esta sección se analizan comparativamente los resultados obtenidos de implementación sobre dispositivos de tipo FPGA de algoritmos de compresión con pérdidas predictivos con los resultados disponibles en el estado del arte presentados a lo largo del capítulo 2.

Es importante remarcar el hecho de que no resulta representativo realizar comparativas de los resultados de implementación obtenidos para la transformada POT. Como se describe en el capítulo 4, esta transformada no representa la totalidad de un algoritmo de compresión de imágenes hiperespectrales sino la primera etapa de eliminación de la correlación espectral de la imagen, necesitando de una segunda etapa para la eliminación de la correlación espacial para completar el algoritmo de compresión. Por lo tanto, a lo largo de esta sección se realizarán únicamente comparativas entre algoritmos de compresión completos.

Se han obtenido diferentes resultados de implementación del algoritmo CCSDS 123.0 con pérdidas y control de la tasa de bits fruto de las diferentes configuraciones desarrolladas. Para todas las comparaciones, se ha seleccionado la configuración PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) al ser la implementación que presenta mayores prestaciones funcionales en conjunto con una ocupación de recursos moderada.

En la Tabla 5.26 se comparan los resultados de implementación de los algoritmos predictivos estudiados. Se presentan los resultados del algoritmo de compresión con pérdidas LCE y la configuración PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS). Estos resultados han sido obtenidos para

la FPGA calificada para el espacio XQR5VFX130 de la familia Virtex-5QV del fabricante Xilinx.

	Algoritmo de compresión con pérdidas (LCE)		**PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS)**	
	Ocupación	**%**	**Ocupación**	**%**
LUTs	7746	12,1	29461	35,96
CLBs	1937	12,11	7366	35,97
DFF o Latches	4281	5,23	15061	18,39
Bloques RAM	4	0,88	114	38,26
Bloques DSP	25	9,77	21	6,56
Frecuencia	86,96 MHz		83,043 MHz	
Tasa de datos	27,93 Mmuestras/s		34,363 Mmuestras/s	

Tabla 5.26 Resultados sobre la FPGA XQR5VFX130 de los algoritmos LCE y PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS)

Se observa que la ocupación de recursos del algoritmo LCE es menor que la proporcionada por el algoritmo PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS). Este hecho es debido principalmente a la mayor complejidad en la etapa de predicción de este algoritmo. Las causas de la mayor complejidad del algoritmo PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) son: el predictor contenido en el estándar CCSDS 123.0 y la inclusión del algoritmo para el control de la tasa de bits. El predictor incluido en el algoritmo LCE requiere de una pequeña cantidad de muestras vecinas para su operación y carece de mecanismos de control de la tasa de bits. De este modo, el algoritmo LCE presenta una menor complejidad lo que deriva en un menor uso de recursos necesarios para su implementación. En cuanto a frecuencia de operación y tasa de datos ambos algoritmos presentan resultados similares, observándose una ligera mejora en la tasa de bits obtenida en el algoritmo PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS).

Del análisis del estado del arte se deriva que, en el momento de escribir este documento, existen dos implementaciones de algoritmos de compresión con pérdidas predictivos. Por un lado, el algoritmo de compresión

de imágenes hiperespectrales FLEX [28], que supone una evolución casi-sin pérdidas del algoritmo de compresión sin pérdidas FL [57]. Por otro lado, el algoritmo de compresión con pérdidas y control de tasa de bits Hydra [90]. La comparación de los resultados de las implementaciones disponibles en la literatura con las implementaciones aportadas en este trabajo, se presenta en la Tabla 5.27 y la Tabla 5.28. La comparativa se ha realizado en términos de ocupación de recursos, frecuencia de operación y tasa de datos de las implementaciones sobre la FPGA XQR5VFX130 del fabricante Xilinx.

	Algoritmo de compresión con pérdidas (LCE)		**PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS)**		***Fast Lossless Extended* (FLEX)**	
	Ocupación	**%**	**Ocupación**	**%**	**Ocupación**	**%**
LUTs	7746	12,1	29461	35,96	16185	20
CLBs	1937	12,11	7366	35,97	5911	29
DFF o Latches	4281	5,23	15061	18,39	18648	23
Bloques RAM	4	0,88	114	38,26	27	9
Bloques DSP	25	9,77	21	6,56	40	13
Frecuencia	86,96 MHz		83,043 MHz		82,5 MHz	
Tasa de datos	27,93 Mmuestras/s		34,363 Mmuestras/s		3,4 Mmuestras/s	

Tabla 5.27 Comparativa de resultados sobre la FPGA XQR5VFX130 de los algoritmos LCE, PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) y FLEX

	Algoritmo de compresión con pérdidas (LCE)		**PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS)**		***Hydra***	
	Ocupación	**%**	**Ocupación**	**%**	**Ocupación**	**%**
LUTs	7746	12,1	29461	35,96	19957	24
CLBs	1937	12,11	7366	35,97	8496	42
DFF o Latches	4281	5,23	15061	18,39	4296	5
Bloques RAM	4	0,88	114	38,26	158	53
Bloques DSP	25	9,77	21	6,56	71	22
Frecuencia	86,96 MHz		83,043 MHz		-	
Tasa de datos	27,93 Mmuestras/s		34,363 Mmuestras/s		20 Mmuestras/s	

Tabla 5.28 Comparativa de resultados sobre la FPGA XQR5VFX130 de los algoritmos LCE, PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) e Hydra

De la Tabla 5.27 y la Tabla 5.28, se concluye que el algoritmo de compresión con pérdidas LCE presenta una menor ocupación de recursos frente al resto de algoritmos comparados gracias a su menor complejidad. En la comparativa de resultados de implementación de los algoritmos LCE y PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) frente al algoritmo FLEX, Tabla 5.27, destaca la tasa de datos alcanzada por las aportaciones presentadas en esta tesis doctoral. Los algoritmos LCE y PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) proporcionan tasas de datos en torno a 30 Mmuestras/s mientras que, el algoritmo FLEX presenta una tasa de datos de 3,4 Mmuestras/s.

El algoritmo de compresión con pérdidas de imágenes hiperespectrales Hydra, presenta el mecanismo de control de la tasa de bits presentado en la sección 5.2.2. De este modo, resulta interesante la comparativa de resultados de implementación entre este algoritmo y el algoritmo propuesto PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS). En la Tabla 5.28, se observa que la ocupación de recursos y tasa de datos de ambos algoritmos es similar. Destaca el menor número de bloques DSP empleados en el algoritmo PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS). Este hecho confirma la menor complejidad del algoritmo de control de la tasa de bits implementado en este algoritmo frente al algoritmo de "dieta selectiva" implementado en el algoritmo Hydra. Por otro lado, la tasa de datos proporcionada por la implementación del algoritmo PCAM con pérdidas (USQ) y control de la tasa de bits (Modo BS) es de 34,363 Mmuestras/s, valor que mejora la tasa de datos de 20 Mmuestras/s proporcionada por la implementación del algoritmo Hydra.

5.7 Conclusiones

La compresión de imágenes hiperespectrales mediante codificación predictiva es abordada en este capítulo desde el punto de vista del control de la tasa de bits. Se ha tomado como partida el predictor descrito en el estándar recomendado CCSDS 123.0 sobre el que han incluido, en lenguaje C/C++, las modificaciones necesarias para implementar el control de la tasa de bits.

El código original, descrito en la sección 5.2.2, incluye un codificador de rango como codificador entrópico que se complementa con un codificador adaptativo por muestras con el fin de disponer de un algoritmo resultante que sea acorde con el estándar CCSDS 123.0. Se han implementado diferentes estrategias para el control de la tasa de bits, proporcionando la posibilidad de disponer de un algoritmo que, dependiendo de los módulos empleados, pueda operar como un algoritmo de compresión si pérdidas, con pérdidas mediante un cuantificador escalar uniforme y con pérdidas y control de la tasa de bits. Se ha propuesto un nuevo modo para el control de la tasa de bits que simplifica la carga computacional del modo B original alcanzando resultados similares en cuanto a exactitud y precisión en el control de la tasa de bits y a calidad de la imagen reconstruida.

Se ha realizado la evaluación del algoritmo mediante la compresión de un conjunto de imágenes hiperespectrales heterogéneo combinando imágenes de diferentes tipos de sensores. Esto permite la obtención de resultados en términos de exactitud y precisión en el control de la tasa de bits, así como, en cuanto a la ratio distorsión – tasa de bits. El entorno de evaluación permite la realización de comparativas de los resultados obtenidos de las diferentes funcionalidades implementadas. Se ha comprobado que, el modo BS propuesto alcanza resultados similares al modo B original en cuanto a la exactitud y precisión del control de la tasa de bits y en cuanto a calidad de imagen reconstruida.

La implementación hardware ha sido llevada a cabo mediante la herramienta de síntesis de alto nivel CatapultC. De esta forma, se ha procedido a la generación de código RTL a partir del algoritmo propuesto en

C/C++. La implementación se ha desarrollado sobre la FPGA XQR5VFX130 de la familia Virtex-5QV del fabricante Xilinx, ya que este dispositivo se encuentra cualificado para operar en el entorno espacial. Se han obtenido resultados de síntesis de cada uno de los módulos del algoritmo con el fin de poder realizar diferentes implementaciones hardware. Se ha comprobado que, todas las configuraciones sintetizadas, pueden implementarse en la FPGA objetivo, alcanzando una ocupación máxima del 59,46 % en bloques de lógica configurable y del 59,44 % de ocupación en LUTs. El mejor caso arroja resultados de ocupación del 30,96 % y del 29,10 % de bloques de lógica configurable y LUTs respectivamente. Los resultados indican que la frecuencia máxima alcanzada es de 83,043 MHz, lo que produce una tasa de datos de 34,363 Mmuestras/s.

Se ha realizado un análisis comparativo de los resultados obtenidos en las diferentes implementaciones extendiéndose a las implementaciones disponibles en el estado del arte. Se concluye que tanto el algoritmo LCE como las diferentes configuraciones de algoritmos predictivos con pérdidas y control de la tasa de bits presentados en este capítulo, presentan mejores o similares resultados de ocupación de recursos, frecuencia de funcionamiento y tasa de datos que las implementaciones disponibles en la literatura.

Los resultados presentados confirman el objetivo propuesto en esta tesis doctoral centrado en el estudio e implementación del algoritmo de compresión predictivo de imágenes hiperespectrales con pérdidas y con control de la tasa de bits para ser empleado en dispositivos FPGA cualificados para el entorno espacial.

Capítulo 6

Conclusiones y líneas futuras

En este capítulo se presentan las principales contribuciones esta Tesis Doctoral. Se destacan las implementaciones hardware de algoritmos de compresión con pérdidas de imágenes hiperespectrales aportadas sobre dispositivos de tipo FPGA calificados para espacio. Estas implementaciones hacen uso de metodologías de diseño de síntesis de alto nivel demostrando la posibilidad de emplear este tipo de metodologías en implementaciones a bordo de futuras misiones espaciales. Asimismo, se aportan líneas futuras de investigación que extiendan los trabajos realizados.

6.1 Conclusiones y contribuciones

La observación de la tierra ha experimentado una amplia evolución en desde el inicio de la era espacial. Los instrumentos disponibles en la carga útil a bordo de los satélites se encuentran altamente optimizados para alcanzar los objetivos de la misión científica para la que son diseñados. Estos objetivos de misión presentan necesidades que se incrementan de una generación de satélites a la siguiente. Este hecho deriva en la necesidad de sensores a bordo que proporcionen mayores resoluciones y prestaciones. Los sensores, actuales y futuros, generan un gran volumen de datos que es necesario procesar, almacenar y transmitir a tierra. De entre todos ellos, destacan los sensores hiperespectrales por la gran cantidad de información que generan tanto espacial como espectral.

Debido al limitado, y en ocasiones insuficiente, ancho de banda de comunicaciones desde el segmento espacial al segmento terreno y a las limitaciones en cuanto a capacidad de almacenamiento, existe un esfuerzo creciente en proporcionar algoritmos de compresión a bordo con el fin de optimizar tanto el canal de comunicaciones como el almacenamiento disponible. En este sentido, es igualmente importante la eficiencia del algoritmo de compresión propuesto como el de disponer implementaciones sobre los dispositivos calificados para operar en el espacio. Los sistemas de procesados de datos disponibles a bordo se basan en tecnologías que presentan capacidades limitadas ya que han de operar en el entorno espacial, hecho que implica la necesidad de emplear estrategias de mitigación para evitar los efectos producidos por la radiación ionizante presente en el espacio.

La compresión de imágenes hiperespectrales, tanto sin pérdidas como con pérdidas, no es ajena a estas necesidades siendo necesaria la evaluación de las prestaciones del algoritmo, en términos de compresión, conjuntamente con la viabilidad de su implementación sobre dispositivos capaces de operar en el entorno espacial. Entre las tecnologías calificadas disponibles, los dispositivos de tipo FPGA destacan por su flexibilidad y capacidad de procesado paralelo, todo ello combinado con un consumo de potencia inferior a tecnologías como GPUs.

Se ha realizado un análisis de los diferentes algoritmos de compresión de imágenes hiperespectrales disponibles en la literatura. Este análisis forma parte fundamental de los trabajos desarrollados en esta tesis doctoral ya que ha servido para detectar las carencias de los algoritmos propuestos así como para identificar aquellos algoritmos que, posteriormente, han sido implementados. Se observó que, tradicionalmente, los algoritmos de compresión sin pérdidas han sido los empleados y que, además, presentan implementaciones capaces de ser desplegadas a bordo de satélites. Este hecho se debe principalmente a que este tipo de algoritmos conservan la totalidad de la información en la imagen reconstruida. Sin embargo, existe una tendencia actual marcada por los algoritmos de compresión con pérdidas de imágenes hiperespectrales fruto del incesante incremento de información generada a bordo por los sensores hiperespectrales. Por lo tanto, proporcionar implementaciones que demuestren la posibilidad del empleo de este tipo de algoritmos con pérdidas resulta parte fundamental para completar su desarrollo y que se postulen como alternativas reales para ser operados a bordo. Los trabajos desarrollados en esta tesis doctoral aportan soluciones en el ámbito de la implementación sobre FPGAs de algoritmos de compresión con pérdidas de imágenes hiperespectrales para ser desplegados en futuras misiones espaciales.

Dentro de los algoritmos de compresión con pérdidas de imágenes hiperespectrales destaca el algoritmo predictivo LCE. Este algoritmo ha sido especialmente diseñado para su despliegue en la misión ExoMars cumpliendo con requisitos de baja complejidad computacional y tolerancia a errores. Se ha realizado su implementación mediante metodologías de diseño de síntesis de alto nivel. Estas metodologías suponen un paso adelante en cuanto a la implementación de algoritmos complejos acortando los tiempos de desarrollo y aumentando el nivel de abstracción desde el que se realiza la implementación. Se ha realizado la implementación teniendo como objetivo los dispositivos FPGA calificados para espacio Xilinx Virtex-5QV (XQR5VFX130) y Microsemi RTAX (RTAX2000S). Los resultados obtenidos confirman la viabilidad en la implementación hardware del algoritmo de compresión de imágenes hiperespectrales LCE de naturaleza predictiva y con pérdidas para

ser empleado en dispositivos FPGA cualificados para el entorno espacial. De este modo, se en esta tesis se aporta, por un lado, una implementación que valida el despliegue en un satélite del algoritmo LCE y, por otra, la validez del empleo de metodologías de síntesis de alto nivel para implementaciones centradas en el sector espacial.

Los algoritmos de compresión basados en transformadas, son conocidos por proporcionar buenas ratios de compresión gracias a su capacidad para eliminar la correlación existente en los datos. La principal desventaja de este tipo de algoritmos de compresión reside en la complejidad de las operaciones involucradas en su cálculo, imposibilitando su despliegue sobre los dispositivos calificados para espacio. En esta tesis doctoral se aporta la implementación de las operaciones aritméticas de la transformada POT demostrando la posibilidad de emplear esta transformada como etapa de eliminación de la correlación espectral para la compresión de imágenes hiperespectrales en futuras misiones espaciales. La implementación se ha realizado a nivel RTL mediante el lenguaje de descripción VHDL, demostrando los beneficios del empleo de dispositivos de tipo FPGA para la compresión de imágenes hiperespectrales a bordo de satélites. Se han empleado en el desarrollo dos dispositivos FPGA calificados para operar en el espacio, la RTAX2000S y RTAX2000S-DSP del fabricante Microsemi. Asimismo, se han implementado hasta ocho configuraciones diferentes demostrando que, cualquiera de ellas, puede implementarse en las FPGAs objetivo. Los resultados muestran que, comparativamente, la FPGA RTAX2000S-DSP ofrece mejores resultados de implementación proporcionando un menor uso de recursos y una mayor frecuencia de operación gracias a la presencia de bloques DSP dedicados.

El algoritmo de compresión basado en transformada POT alcanza unas prestaciones de en términos de compresión que le han permitido postularse como candidato para ser empleado como extensión en la compresión de imágenes hiperespectrales del estándar de compresión de imágenes CCSDS 122.0. Sin embargo, no existían, hasta la realización de los trabajos presentados en esta tesis doctoral, implementaciones de la transformada POT que demostraran su viabilidad para ser empleado en una misión espacial. Por lo tanto, las aportaciones realizadas, en términos de implementación sobre

FPGAs calificadas para espacio, dan continuidad a que la transformada POT forme parte del nuevo estándar de compresión de imágenes hiperespectrales basado en transformada a ser publicado por el comité CCSDS.

La compresión predictiva con control de la tasa de bits, si bien es una característica deseable en un algoritmo de compresión de imágenes hiperespectrales, resulta un reto ya que no existe una relación matemática simple entre la tasa de bits y el residuo predicho cuantificado. Se han estudiado las principales aportaciones algorítmicas disponibles en la literatura, destacando los algoritmos presentados en [81] y [30]. El algoritmo presentado en [30] presenta una menor complejidad computacional, siendo el seleccionado para su implementación en esta tesis doctoral. Estos algoritmos hacen uso del predictor perteneciente al estándar de compresión sin pérdidas de imágenes hiperespectrales CCSDS 123.0, extendiendo su funcionalidad mediante la inclusión de pérdidas y diferentes estrategias para el control de la tasa de bits. Partiendo de la implementación en C/C++ del algoritmo presentado en [30], se ha propuesto un nuevo modo para el control de la tasa de bits que simplifica la carga computacional alcanzando resultados similares a los propuestos en cuanto a exactitud en el control de la tasa de bits y a calidad de la imagen reconstruida. Asimismo, se incluye el codificador adaptativo por muestras perteneciente al estándar CCSDS 123.0 en la etapa de codificación entrópica como alternativa al codificador de rango propuesto [30]. Las modificaciones realizadas sobre el código original originan la necesidad de la validación funcional del algoritmo resultante. Esta validación se ha realizado sobre un conjunto de imágenes hiperespectrales de diferentes sensores comprobándose el correcto funcionamiento del algoritmo en todas sus posibles configuraciones.

La implementación del algoritmo de compresión con pérdidas predictivo y con control de la tasa de bits se ha realizado mediante el empleo de metodologías de síntesis de alto nivel, acortando de esta forma su desarrollo. Se ha realizado la implementación del algoritmo teniendo como objetivo el dispositivo FPGA calificado para espacio Xilinx Virtex-5QV (XQR5VFX130). Los resultados obtenidos concluyen que tanto el algoritmo de

compresión desarrollado así como su implementación pueden ser empleados en el entorno espacial.

Finalmente, se han comparado los resultados aportados en esta tesis doctoral con los resultados disponibles en el estado del arte. Se concluye que las implementaciones propuestas resultan alternativas viables para ser desplegadas a bordo de la siguiente generación de misiones espaciales que requieran de la compresión con pérdidas de imágenes hiperespectrales. Asimismo, los resultados obtenidos en términos de ocupación de área y tasa de datos, son comparables con las implementaciones disponibles en el estado del arte pese a la menor funcionalidad implementada en algoritmos como el denominado FLEX [28]. Puede concluirse que, si bien los algoritmos predictivos con pérdidas continúan presentando una menor complejidad computacional que los algoritmos de compresión basados en transformadas, esta tesis doctoral proporciona implementaciones capaces de operar a bordo de futuras misiones espaciales de ambos tipos de algoritmos de algoritmos de compresión de imágenes hiperespectrales. Por lo tanto, la decisión del empleo de un tipo de algoritmo sobre otro queda abierto a las necesidades de cada misión.

Los trabajos presentados satisfacen los objetivos propuestos en este trabajo de tesis doctoral. Se ha estudiado la viabilidad de los algoritmos de compresión con pérdidas para la compresión a bordo de imágenes hiperespectrales. Es destacable el hecho de que, en este trabajo, se ha abordado el estudio e implementación de algoritmos de compresión predictivos y basados en transformadas. Se han aportado soluciones a nivel de implementación sobre dispositivos calificados para espacio de tipo FPGA para los algoritmos estudiados. Además, se han empleado metodologías de síntesis de alto nivel y diseño RTL en las distintas implementaciones realizadas, demostrando la viabilidad de ambas aproximaciones para la implementación de algoritmos de compresión de imágenes hiperespectrales sobre dispositivos capaces de operar bordo de satélites. Los trabajos realizados en esta tesis doctoral pretenden servir en el despliegue de los algoritmos de compresión con pérdidas de imágenes hiperespectrales a bordo de futuras misiones espaciales.

6.2 Líneas futuras

Los resultados obtenidos motivan la exploración de nuevas líneas de investigación que complementen y extiendan los trabajos realizados en esta tesis doctoral.

La necesidad de evolución de las capacidades de procesado a bordo, se ha identificado por parte de la agencia espacial europea (ESA) como aspecto esencial en el despliegue de futuras misiones espaciales. La armonización y elaboración de la estrategia futura de desarrollo de tecnología espacial en Europa se recoge en los dosieres de armonización para las diferentes tecnologías desplegadas en el sector espacial. El dosier técnico de armonización de tecnologías centradas en el procesado de datos a bordo vigente para el periodo 2016-2024, establece como requisito técnico estratégico el desarrollo de nuevos y avanzados algoritmos de procesados de datos, selección y compresión como soporte de los sistemas de procesados de datos y cargas útiles [11]. Por lo tanto, en los próximos años es previsible la aparición tanto de nuevos dispositivos como de algoritmos y estándares de compresión de imágenes hiperespectrales. Estos algoritmos necesitarán de la continuidad en términos de trabajos de implementación que confirmen su viabilidad para ser desplegados en los nuevos dispositivos disponibles.

Con respecto al algoritmo LCE, los resultados de implementación obtenidos hacen posible el funcionamiento en paralelo de varias entidades del algoritmo si se emplea la FPGA Xilinx Virtex-5QV (XQR5VFX130). Este hecho se ve potenciado por el procesado de la imagen hiperespectral en bloques independientes característico de este algoritmo. El interés de la inclusión de varias entidades del algoritmo en paralelo reside en la mejora en términos de tasa de bits proporcionado por el algoritmo. Teóricamente es posible la introducción de 8 implementaciones del algoritmo LCE en la FPGA trabajando en paralelo lo que se traduciría en un incremento en la misma magnitud tanto de los recursos empleados como de la tasa de datos, siendo capaz de procesar aproximadamente 200 Mmuestras/s con una ocupación máxima del 97% de bloques lógicos configurables. Sin embargo, tanto la gestión del

funcionamiento en paralelo de las entidades como los accesos a memoria presentan un reto con el fin de optimizar el flujo de los datos procesados.

La implementación del algoritmo de compresión basado en la transformada POT presenta unos resultados atractivos para su despliegue a bordo de un satélite. Esta transformada forma parte de la totalidad del sistema de compresión de una imagen hiperespectral siendo empleado como etapa de eliminación de la correlación espectral. Por lo tanto, requiere de una etapa posterior de eliminación de la redundancia espacial en la imagen, como puede ser el estándar de compresión de imágenes CCSDS 122.0. Actualmente, existen implementaciones en dispositivos tipo FPGA y ASIC de este estándar. Por lo tanto, la mejor solución disponible a implementar en una misión real en este momento estaría basada en la implementación de la transformada POT en un dispositivo tipo FPGA y el estándar CCSDS 122.0 en un ASIC. En este sentido, se plantea iniciar la investigación de la implementación de ambos algoritmos sobre un mismo dispositivo de tipo FPGA calificado para espacio una vez sea publicado el estándar de compresión de imágenes hiperespectrales en el que la transformada POT sea adoptada. La implementación resultante es de gran interés en términos de planificación de una misión ya que se mejorarían las prestaciones en cuanto a tamaño, peso y consumo de potencia de la unidad de compresión dentro del sistema de procesado de datos.

En el caso del algoritmo de compresión con pérdidas predictivo y con control de la tasa de bits, se pueden explorar distintas líneas en el futuro que pueden mejorar las prestaciones proporcionadas por el algoritmo. El cuantificador empleado en el algoritmo implementado se basa en un cuantificador escalar uniforme (USQ). La etapa de cuantificación es la etapa fundamental en la introducción de pérdidas dentro de un algoritmo de compresión con pérdidas. Por lo tanto, pese a que el cuantificador USQ empleado proporciona buenos resultados en cuanto a la calidad de la imagen reconstruida, pueden explorarse cuantificadores más sofisticados como los basados en umbral, no lineales o incluso la cuantificación vectorial con el fin de optimizar el binomio tasa de bits – distorsión en la imagen reconstruida. Con respecto a la etapa de codificación entrópica, se concluye que la

complejidad en la implementación del codificador de rango, lo convierte en una solución menos atractiva para ser desplegada a bordo que otras soluciones como el codificador adaptativo por muestras perteneciente al estándar CCSDS 123.0. Una tendencia actual es el empleo de codificadores entrópicos híbridos en donde se implementan varios tipos de codificadores y se selecciona cuál de ellos ha de funcionar en función de la entropía presente en los datos [28]. Resulta atractiva la exploración de los diferentes codificadores entrópicos en compresores con pérdidas especialmente con tasas de bits muy bajas, por debajo de 1 bpp, ya que los codificadores actuales centrados en estas tasas de bits presentan una alta carga computacional dificultando su implementación.

6.3 Publicaciones

En esta sección, se resumen en orden cronológico las publicaciones en revistas y congresos realizadas como parte de los trabajos de esta tesis doctoral.

6.3.1 Revistas

[R1] Santos, L., Blanes, I., García, A., Serra-Sagristà, J., López, J.F., Sarmiento, R. (2015). On the hardware implementation of the arithmetic elements of the pairwise orthogonal transform. Journal of Applied Remote Sensing 9(1), 097496, 2015.

6.3.2 Congresos

[C1] García, A., Santos, L., López, S., Callicó, G.M., López, J.F., Sarmiento, R. (2013). High level modular implementation of a lossy hyperspectral image compression algorithm on a FPGA. In IEEE Workshop on Hyperspectral Image and Signal Processing Evolution in Remote Sensing (WHISPERS), 2013.

[C2] García, A., Santos, L., López, S., Callicó, G. M., Lopez, J. F., and Sarmiento, R. (2014). Efficient lossy compression implementations of hyperspectral images: tools, hardware platforms, and comparisons. In SPIE Sensing Technology+ Applications (pp. 912408-912408).

[C3] García, A., Santos, L., López, S., Callicó, G. M., López, J.F., Sarmiento, R. (2014). FPGA implementation of the hyperspectral Lossy Compression for Exomars (LCE) algorithm. Proceedings SPIE 9247, High-Performance Computing in Remote Sensing IV, 924705, 2014.

[C4] Santos, L., Blanes, I., García, A., Serra-Sagristà, J., López, J.F., Sarmiento, R. (2014). On the hardware implementation of the complex elements of the pairwise orthogonal transform. On-Board Payload Data Compression Workshop (OBPDC), 2014

[C5] Santos, L., García, A., López, J.F., Sarmiento, R. (2014). Highly parallel hardware architectures for lossy hyperspectral image compression on-board satellites. Proceedings of the 2014 conference on Big Data from Space (BiDS'14), 2014

Bibliografía

[1] G. W. Swenson, «Looking back: Sputnik», *IEEE Potentials*, vol. 16, n.º 1, pp. 36-40, mar. 1997.

[2] N. Aeronautics y S. A. (NASA), «Landsat Science», 2014. [En línea]. Disponible en: http://landsat.gsfc.nasa.gov/. [Accedido: 18-may-2017].

[3] C. N. d'Etudes S. (CNES), «SPOT Earth Observation Mission. Eyes in Space.», 2013. [En línea]. Disponible en: http://smsc.cnes.fr/SPOT/. [Accedido: 20-may-2017].

[4] N. Aeronautics y S. A. (NASA), «Airborne Visible/Infrared Imaging Spectometer (AVIRIS)», 2014. [En línea]. Disponible en: http://aviris.jpl.nasa.gov/. [Accedido: 05-jul-2017].

[5] F. Collomb, «Copernicus». [En línea]. Disponible en: http://www.copernicus.eu/. [Accedido: 18-may-2017].

[6] R. Vitulli, P. Armbruster, y D. Merodio, «Big Data Starts On-board», *Big Data Space BiDS*, mar. 2016.

[7] P. Soille *et al.*, *Proceedings of the 2014 conference on Big Data from Space (BiDS'14)*. Luxembourg: Publications Office, 2014.

[8] P. Soille, P. G. Marchetti, European Commission, Joint Research Centre, y Big Data from Space (BiDS'16), *Proceedings of the 2016 conference on Big Data from Space (BiDS'16): 15th-17th March 2016 Santa Cruz de Tenerife (Spain)*. Luxembourg: Publications Office, 2016.

[9] F. Miranda, «Recent Efforts in Advanced High Frequency Communications at the Glenn Research Center in Support of NASA Mission», *Forum Electromagn. Res. Methods Appl. Technol. FERMAT*, mar. 2015.

[10]F. Heine, G. Muhlnikel, H. Zech, S. Philipp-May, y R. Meyer, «The European Data Relay System, high speed laser based data links», presentado en Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), 2014, 2014, pp. 284-286.

[11]M. Suess, R. Trautner, R. Vitulli, D. Thurnes, D. Jameux, y J. Both, *Technical Dossier on On-Board Payload Data Processing*. IPC TECHNOLOGY HARMONISATION ADVISORY GROUP, 2016.

[12]D. Landgrebe, «Hyperspectral image data analysis», *IEEE Signal Process. Mag.*, vol. 19, n.º 1, pp. 17-28, ene. 2002.

[13]M. Zortea y A. Plaza, «A Quantitative and Comparative Analysis of Different Implementations of N-FINDR: A Fast Endmember Extraction Algorithm», *IEEE Geosci. Remote Sens. Lett.*, vol. 6, n.º 4, pp. 787-791, oct. 2009.

[14]J. Plaza, A. Plaza, D. Valencia, y A. Paz, «Massively parallel processing of remotely sensed hyperspectral images», presentado en Proc. SPIE 7455, Satellite Data Compression, Communication, and Processing V, 74550O, 2009, p. 74550O-74550O-11.

[15]A. Plaza, P. Martinez, R. Perez, y J. Plaza, «Spatial/spectral endmember extraction by multidimensional morphological operations», *IEEE Trans. Geosci. Remote Sens.*, vol. 40, n.º 9, pp. 2025-2041, sep. 2002.

[16]L. O. Jimenez, J. L. Rivera-Medina, E. Rodriguez-Diaz, E. Arzuaga-Cruz, y M. Ramirez-Velez, «Integration of spatial and spectral information by means of unsupervised extraction and classification for homogenous objects applied to multispectral and hyperspectral data», *IEEE Trans. Geosci. Remote Sens.*, vol. 43, n.º 4, pp. 844-851, abr. 2005.

[17]B. Penna, T. Tillo, E. Magli, y G. Olmo, «Transform Coding Techniques for Lossy Hyperspectral Data Compression», *Geosci. Remote Sens. IEEE Trans. On*, vol. 45, n.º 5, pp. 1408-1421, may 2007.

[18]S. Lopez, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos, y A. Plaza, «The Promise of Reconfigurable Computing for Hyperspectral Imaging

Onboard Systems: A Review and Trends», *Proc. IEEE*, vol. 101, n.º 3, pp. 698-722, mar. 2013.

[19]R. E. Roger y M. C. Cavenor, «Lossless compression of AVIRIS images», *IEEE Trans. Image Process.*, vol. 5, n.º 5, pp. 713-719, may 1996.

[20]J. S. Mielikainen, A. Kaarna, y P. J. Toivanen, «Lossless hyperspectral image compression via linear prediction», presentado en Proc. SPIE 4725, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery VIII, 600, 2002, pp. 600-608.

[21]I. Blanes, E. Magli, y J. Serra-Sagrista, «A Tutorial on Image Compression for Optical Space Imaging Systems», *IEEE Geosci. Remote Sens. Mag.*, vol. 2, n.º 3, pp. 8-26, sep. 2014.

[22]B. Aiazzi, L. Alparone, y S. Baronti, «Near-lossless compression of 3-D optical data», *IEEE Trans. Geosci. Remote Sens.*, vol. 39, n.º 11, pp. 2547-2557, nov. 2001.

[23]M. Cabral, R. Trautner, R. Vitulli, y C. Monteleone, «Efficient Data Compression for spacecraft including Planetary Probes», *IPP7 Barc.*, 2010.

[24]L. Santos, «Hyperspectral Image Compression Onboard Next-Generation Satellites: Implementation Solution on GPUs and FPGAs», Universidad de Las Palmas de Gran Canaria, 2014.

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

[26]A. Kiely, M. Klimesh, N. Aranki, y M. Burl, «Multispectral & Hyperspectral Image Compression Development at the Jet Propulsion Laboratory», *Board Payload Data Compression Workshop*, sep. 2016.

[27]L. Santos, L. Berrojo, J. Moreno, J. F. López, y R. Sarmiento, «Multispectral and Hyperspectral Lossless Compressor for Space Applications (HyLoC): A Low-Complexity FPGA Implementation of the CCSDS 123 Standard», *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 9, n.º 2, pp. 757-770, feb. 2016.

[28]D. Keymeulen, H. Luong, D. Dolman, C. Holyoake, K. Crocker, y P. Thang, «FPGA Implementation of Space-based Lossless and Lossy Multispectral and Hyperspectral Image Compression», *Board Payload Data Compression Workshop*, sep. 2016.

[29]A. Abrardo, M. Barni, y E. Magli, «Low-complexity predictive lossy compression of hyperspectral and ultraspectral images», en *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011, pp. 797-800.

[30]D. Valsesia y E. Magli, «Fast and Lightweight Rate Control for Onboard Predictive Coding of Hyperspectral Images», *IEEE Geosci. Remote Sens. Lett.*, vol. 14, n.º 3, pp. 394-398, mar. 2017.

[31]I. Blanes y J. Serra-Sagrista, «Pairwise Orthogonal Transform for Spectral Image Coding», *tgrs*, vol. 49, n.º 3, pp. 961–972, 2011.

[32]I. Blanes, M. Hernandez-Cabronero, F. Auli-Llinas, J. Serra-Sagrista, y M. W. Marcellin, «Isorange Pairwise Orthogonal Transform», *IEEE Trans. Geosci. Remote Sens.*, vol. 53, n.º 6, pp. 3361-3372, jun. 2015.

[33]*Image data compression recommended standard CCSDS 122.0-B-1*. The Consultative Committee for Space Data Systems, 2005.

[34]P. Carballo, «Aportaciones a la metodología de diseño basada en Síntesis de Alto Nivel. Aportaciones al diseño de IPs para procesado de eventos complejos y codificación de vídeo», Universidad de Las Palmas de Gran Canaria (ULPGC), 2016.

[35]«Catapult C/C++/SystemC HLS». [En línea]. Disponible en: https://www.mentor.com/hls-lp/catapult-high-level-synthesis/c-systemc-hls. [Accedido: 07-may-2017].

[36]A. Skodras, C. Christopoulos, y T. Ebrahimi, «The JPEG 2000 still image compression standard», *IEEE Signal Process. Mag.*, vol. 18, n.º 5, pp. 36-58, sep. 2001.

[37]D. Valsesia, M. De Nino, y E. Magli, «Compression algorithm and implementation for the PRISMA mission», *5th Int. Workshop Board Payload Data Compression OBPDC 2016*, oct. 2016.

[38]E. Christophe, D. Leger, y C. Mailhes, «Quality criteria benchmark for hyperspectral imagery», *IEEE Trans. Geosci. Remote Sens.*, vol. 43, n.º 9, pp. 2103-2114, sep. 2005.

[39]J. Reichel, G. Menegaz, M. J. Nadenau, y M. Kunt, «Integer wavelet transform for embedded lossy to lossless image compression», *IEEE Trans. Image Process.*, vol. 10, n.º 3, pp. 383-392, mar. 2001.

[40]A. Said y W. A. Pearlman, «An image multiresolution representation for lossless and lossy compression», *IEEE Trans. Image Process.*, vol. 5, n.º 9, pp. 1303-1310, sep. 1996.

[41]Lei Wang, Jiaji Wu, Licheng Jiao, y Guangming Shi, «Lossy-to-Lossless Hyperspectral Image Compression Based on Multiplierless Reversible Integer TDLT/KLT», *IEEE Geosci. Remote Sens. Lett.*, vol. 6, n.º 3, pp. 587-591, jul. 2009.

[42]C. E. Shannon, «A Mathematical Theory of Communication», *SIGMOBILE Mob Comput Commun Rev*, vol. 5, n.º 1, pp. 3–55, ene. 2001.

[43]N. S. Jayant y P. Noll, *Digital coding of waveforms: principles and applications to speech and video*, 5. print. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[44]J. Mielikainen, «Lossless compression of hyperspectral images using lookup tables», *IEEE Signal Process. Lett.*, vol. 13, n.º 3, pp. 157-160, mar. 2006.

[45]B. Huang y Y. Sriraja, «Lossless compression of hyperspectral imagery via lookup tables with predictor selection», presentado en Proc. SPIE 6365, Image and Signal Processing for Remote Sensing XII, 63650L, 2006, p. 63650L.

[46]B. Aiazzi, P. Alba, L. Alparone, y S. Baronti, «Lossless compression of multi/hyper-spectral imagery based on a 3-D fuzzy prediction», *IEEE Trans. Geosci. Remote Sens.*, vol. 37, n.º 5, pp. 2287-2294, sep. 1999.

[47]J. Mielikainen y P. Toivanen, «Clustered DPCM for the lossless compression of hyperspectral images», *Geosci. Remote Sens. IEEE Trans. On*, vol. 41, n.º 12, pp. 2943-2946, 2003.

[48]X. Wu y N. Memon, «Context-based lossless interband compression-extending CALIC», *Image Process. IEEE Trans. On*, vol. 9, n.º 6, pp. 994-1001, 2000.

[49]E. Magli, G. Olmo, y E. Quacchio, «Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC», *Geosci. Remote Sens. Lett. IEEE*, vol. 1, n.º 1, pp. 21-25, 2004.

[50]F. Rizzo, B. Carpentieri, G. Motta, y J. A. Storer, «Low-complexity lossless compression of hyperspectral imagery via linear prediction», *IEEE Signal Process. Lett.*, vol. 12, n.º 2, pp. 138-141, feb. 2005.

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

[52]R. Rice, P.-S. Yeh, y W. Miller, «Algorithms for high-speed universal noiseless coding», presentado en 9th AIAA Computing in Aerospace Conference, 1993.

[53]A. G. Villafranca, S. Mignot, J. Portell, y E. Garcia-Berro, «Hardware implementation of the FAPEC lossless data compressor for space», presentado en 2010 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2010, pp. 164-170.

[54]L. Li, B. Fiethe, H. Michalik, y O. Björn, «Efficient Implementation of the CCSDS 122.0-B-1 Standard on Space-qualified FPGAs», en *Proceedings of 2012 ESA workshop on Onboard Payload Data Compression (OBPDC)*, 2012.

[55]*Lossless Multispectral and Hyperspectral Image Compression Informational Report CCSDS 120.2-G-1*. The Consultative Committee for Space Data Systems, 2015.

[56]M. Klimesh, «Low-Complexity Lossless Compression of Hyperspectral Imagery via Adaptive Filtering», *Interplanet. Netw. Prog. Rep.*, nov. 2005.

[57]M. Klimesh, «Low-complexity adaptive lossless compression of hyperspectral imagery», *Proc. SPIE*, vol. Vol. 6300, 63000N, 2006.

[58]Sunghyun Lim, Kwanghoon Sohn, y Chulhee Lee, «Compression for hyperspectral images using three dimensional wavelet transform», presentado en IEEE 2001 International Geoscience and Remote Sensing Symposium (IGARSS '01), 2001, vol. 1, pp. 109-111.

[59]A. Kaarna y J. Parkkinen, «Comparison of compression methods for multispectral images», *Nord. Signal Process Symp*, vol. 2, pp. 251–254, 2000.

[60]G. P. Abousleman, M. W. Marcellin, y B. R. Hunt, «Compression of hyperspectral imagery using the 3-D DCT and hybrid DPCM/DCT», *IEEE Trans. Geosci. Remote Sens.*, vol. 33, n.º 1, pp. 26-34, ene. 1995.

[61]D. Markman y D. Malah, «Hyperspectral image coding using 3D transforms», presentado en International Conference on Image Processing, 2001, vol. 1, pp. 114-117.

[62]K. A y K. M, «The ICER Progressive Wavelet Image Compressor», en *Interplanetary Network Progress Report 42-155*, 2003, pp. 1-46.

[63]K. A, K. M, X. H, y A. N, «ICER-3D: A Progressive Wavelet-based Compressor for Hyperspectral Images», en *Interplanetary Network Progress Report 42-164*, 2006, pp. 1-21.

[64]Q. Du y J. E. Fowler, «Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis», *Geosci. Remote Sens. Lett. IEEE*, vol. 4, n.º 2, pp. 201-205, abr. 2007.

[65]A. Karami, M. Yazdi, y G. Mercier, «Compression of Hyperspectral Images Using Discerete Wavelet Transform and Tucker Decomposition», *Sel. Top. Appl. Earth Obs. Remote Sens. IEEE J. Of*, vol. 5, n.º 2, pp. 444-450, abr. 2012.

[66]M. D. Pal, C. M. Brislawn, y S. R. Brumby, «Feature extraction from hyperspectral images compressed using the JPEG-2000 standard», presentado en Fifth IEEE Southwest Symposium on Image Analysis and Interpretation, 2002, pp. 168-172.

[67]X. Tang, S. Cho, y W. A. Pearlman, «3D set partitioning coding methods in hyperspectral image compression», presentado en 2003 International Conference on Image Processing, 2003, vol. 3, p. II-239-42.

[68]X. Tang y W. A. Pearlman, «Three-Dimensional Wavelet-Based Compression of Hyperspectral Images», en *Hyperspectral Data Compression*, G. Motta, F. Rizzo, y J. A. Storer, Eds. Boston: Kluwer Academic Publishers, 2006, pp. 273-308.

[69]Sunghyun Lim, Kwang Hoon Sohn, y Chulhee Lee, «Principal component analysis for compression of hyperspectral images», presentado en IEEE 2001 International Geoscience and Remote Sensing Symposium (IGARSS '01), 2001, vol. 1, pp. 97-99.

[70]J. A. Saghri, A. G. Tescher, y J. T. Reagan, «Practical transform coding of multispectral imagery», *IEEE Signal Process. Mag.*, vol. 12, n.º 1, pp. 32-43, ene. 1995.

[71]L. Chang, C.-M. Cheng, y T.-C. Chen, «An efficient adaptive KLT for multispectral image compression», en *Image Analysis and Interpretation, 2000. Proceedings. 4th IEEE Southwest Symposium*, 2000, pp. 252-255.

[72]P. Hao y Q. Shi, «Reversible integer KLT for progressive-to-lossless compression of multiple component images», en *Image Processing, 2003.*

ICIP 2003. Proceedings. 2003 International Conference on, 2003, vol. 1, p. I-633-6 vol.1.

[73]B. Penna, T. Tillo, E. Magli, y G. Olmo, «Progressive 3-D coding of hyperspectral images based on JPEG 2000», *Geosci. Remote Sens. Lett. IEEE*, vol. 3, n.º 1, pp. 125-129, 2006.

[74]C. Egho y T. Vladimirova, «Adaptive hyperspectral image compression using the KLT and integer KLT algorithms», presentado en 2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2014, pp. 112-119.

[75]C. Egho y T. Vladimirova, «Hardware acceleration of the Integer Karhunen-Loève Transform algorithm for satellite image compression», presentado en 2012 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2012, pp. 4062-4065.

[76]L. Santos, S. Lopez, G. M. Callico, J. F. Lopez, y R. Sarmiento, «Performance Evaluation of the H.264/AVC Video Coding Standard for Lossy Hyperspectral Image Compression», *Sel. Top. Appl. Earth Obs. Remote Sens. IEEE J. Of*, vol. 5, n.º 2, pp. 451-461, abr. 2012.

[77]M. J. Ryan y J. F. Arnold, «The lossless compression of AVIRIS images by vector quantization», *IEEE Trans. Geosci. Remote Sens.*, vol. 35, n.º 3, pp. 546-550, may 1997.

[78]S. Qian, M. Bergeron, I. Cunningham, L. Gagnon, y A. Hollinger, «Near lossless data compression onboard a hyperspectral satellite», *IEEE Trans. Aerosp. Electron. Syst.*, vol. 42, n.º 3, pp. 851-866, jul. 2006.

[79]Shen-En Qian, A. B. Hollinger, D. Williams, y D. Manak, «Vector quantization using spectral index-based multiple subcodebooks for hyperspectral data compression», *IEEE Trans. Geosci. Remote Sens.*, vol. 38, n.º 3, pp. 1183-1190, may 2000.

[80]M. J. Ryan y J. F. Arnold, «Lossy compression of hyperspectral data using vector quantization», *Remote Sens. Environ.*, vol. 61, n.º 3, pp. 419-436, sep. 1997.

[81]D. Valsesia y E. Magli, «A Novel Rate Control Algorithm for Onboard Predictive Coding of Multispectral and Hyperspectral Images», *IEEE Trans. Geosci. Remote Sens.*, vol. 52, n.º 10, pp. 6341-6355, oct. 2014.

[82]A. G. Holmes-Siedle y L. Adams, *Handbook of radiation effects*, 2nd ed, Repr. Oxford: Oxford Univ. Press, 2007.

[83]M. Poizat, F. Sturesson, y C. Poivey, «Single Event Effects (SEE) Mechanism and Effects», *Space Radiat. Its Eff. EEE Compon.*, jun. 2009.

[84]J. L. Poupat, B. Leroy, y T. Helfers, «TCLS ARM FOR SPACE», *DASIA 2016*, may 2016.

[85]«Microsemi». [En línea]. Disponible en: https://www.microsemi.com/. [Accedido: 24-mar-2017].

[86]«NanoXplore». [En línea]. Disponible en: http://www.nanoxplore.com/. [Accedido: 16-may-2017].

[87]«Space-grade Virtex-4QV FPGA». [En línea]. Disponible en: https://www.xilinx.com/products/silicon-devices/fpga/virtex-4qv.html. [Accedido: 16-may-2017].

[88]«Space-grade Virtex-5QV FPGA». [En línea]. Disponible en: https://www.xilinx.com/products/silicon-devices/fpga/virtex-5qv.html. [Accedido: 16-may-2017].

[89]«ESA Science & Technology: JUICE». [En línea]. Disponible en: http://sci.esa.int/juice/. [Accedido: 17-may-2017].

[90]E. Magli y G. Lopez, «New Techniques for lossy multi/hyperspectral compression for very high data rate instruments», *TEC-ED TEC-SW Final Present. Days*, jun. 2015.

[91]G. J. Sullivan, «On embedded scalar quantization», en *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, 2004, vol. 4, p. iv-605-iv-608 vol.4.

[92]*IEEE Standard VHDL Language Reference Manual*. New York, NY, USA: IEEE, 2009.

[93]S. S. Luca Galli, «Lossless hyperspectral compression using KLT», presentado en 2004 IEEE International Geoscience and Remote Sensing Symposium (IGARSS'04), 2004, vol. 1, pp. 313-316.

[94]Pengwei Hao y Qingyun Shi, «Matrix factorizations for reversible integer mapping», *IEEE Trans. Signal Process.*, vol. 49, n.º 10, pp. 2314-2324, oct. 2001.

[95]*ECSS-Q-ST-60-02C - Space product assurance - ASIC and FPGA development*. European Cooperation for Space Standardization (ECSS), 2008.

[96]R. Vitulli, y J. L. Poupat, «CWICOM: the new CCSDS image compression ASIC», *Proc 2012 ESA Workshop Onboard Payload Data Compression OBPDC*, 2010.

[97]D. Valsesia, E. Magli, y M. De Nino, «A Novel Rate-Controlled Predictive Coding Algorithm for Onboard Compression of Multispectral and Hyperspectral Images», *Board Payload Data Compression Workshop*, oct. 2014.