UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Departamento de Arquitectura de Computadores y Automática

TESIS DOCTORAL

Procesamiento a bordo de imágenes hiperespectrales de la superficie terrestre mediante hardware reconfigurable

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Carlos González Calvo

Directoras

Jesús Javier Resano Ezcaray
Daniel Mozos Muñoz
Antonio José Plaza Miguel

Madrid, 2012

Procesamiento a bordo de imágenes hiperespectrales de la superficie terrestre mediante hardware reconfigurable

TESIS DOCTORAL

Carlos González Calvo

Dpto. de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

2011

Procesamiento a bordo de imágenes hiperespectrales de la superficie terrestre mediante hardware reconfigurable

Tesis presentada por

Carlos González Calvo

Dpto. de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

2011

Procesamiento a bordo de imágenes hiperespectrales de la superficie terrestre mediante hardware reconfigurable

Memoria presentada por Carlos González Calvo para optar al grado de Doctor por la Universidad Complutense de Madrid, realizada bajo la dirección de D. Jesús Javier Resano Ezcaray, D. Daniel Mozos Muñoz y D. Antonio José Plaza Miguel.

Madrid, 2011

Este trabajo ha sido financiado por el programa de la Comunidad Europea Marie Curie Research Training Networks con referencia MRTN–CT–2006–035927, Hyperspectral Imaging Network (HYPER–I–NET). Este trabajo también ha sido posible gracias a la Comisión Interministerial de Ciencia y Tecnología, por las ayudas recibidas a través de los proyectos TIN2006–03274, AYA2008–05965–C04–02, AYA2009–13300–C03–02 y TIN2009–09806. También agradecer a la Junta de Extremadura la financiación recibida a través del proyecto PRI09A110.

A mis familiares y amigos, especialmente
a los que ya no verán el resultado

Más que el mastín corre el galgo
pero si el camino es largo
antes llega el mastín que el galgo.

Refrán popular

Agradecimientos

A los que han hipotecado
los momentos a mi lado.

Hace unos años, cuando afronté el reto de hacer esta tesis, me dijeron que era un trabajo personal. Si bien ahora puedo afirmar que, efectivamente, es una labor "básicamente" personal, también he de reconocer que, sin la ayuda y los apoyos recibidos, esta tesis se habría convertido en una empresa imposible. Debo agradecer:

A mi Director de tesis, Javier. Realmente, tu labor en este camino comenzó hace muchos años, cuando yo era un alumno más en búsqueda de un proyecto fin de carrera dentro del Departamento de Arquitectura de Computadores y Automática. Entonces, gracias a tu pasión por la ingeniería y tu actitud de ayuda sin condiciones, sembraste un ingeniero entusiasta de su trabajo y dispuesto a llegar siempre, un poco más lejos. Gracias por la confianza que has puesto en mí.

A mi Director de tesis, Daniel. Debo agradecerte especialmente la ayuda que me has prestado estos años. Las correcciones exhaustivas de las memorias, los acertados análisis técnicos sobre nuestras publicaciones y la aportación de tu experiencia. Tu visión crítica y humana de la vida y de la ingeniería, junto con tu capacidad de dar ánimos en los momentos más difíciles, me han ayudado y me siguen ayudando a afrontar los distintos retos.

A mi Director de tesis, Antonio. Por tu generosidad al brindarme la oportunidad de recurrir a tu capacidad y experiencia científica en un marco de confianza, afecto y amistad, fundamentales para la concreción de este trabajo, y por tus valiosas sugerencias y acertados aportes durante el desarrollo

del mismo. Gracias por ser como eres y por demostrarme que ser un gran científico no está reñido con ser una mejor persona.

A mis compañeros. Tras pasar unos años en la empresa privada, cuando tuve la oportunidad de ingresar en el Departamento no lo dudé. Sabía que me incorporaba a trabajar con excelentes profesionales, pero lo que es más importante para mí, con magníficas personas. Quiero acordarme en primer lugar de David Cuesta, Jesús Fernández, Javier Sánchez y Miguel Peón, a los que no tengo la oportunidad de ver a diario pero que sin duda me han influido notablemente. A mis compañeros de despacho Alberto, Fran, Íñigo, José Luis Lucas, Juanan y Pablo, y a José Luis Vázquez-Poletti y Guillermo por hacerme más llevadero el día a día. He tenido la gran suerte de haber compartido este largo camino con todos vosotros y haberme podido contagiar de vuestro espíritu de superación, sacrificio y esfuerzo; que aún siendo todos estos aspectos inestimables, no son más que un apéndice comparado con vuestra amistad.

A mis amigos, que sabéis apreciar las aventuras en las que me embarco (como esta memoria), en especial a Adolfo, José Javier y Paco. Perdonarme si estos últimos años no os he dedicado todo el tiempo que merecíais.

A mi familia. Hemos avanzado estos años contra viento y marea, sin vuestra fortaleza y cariño, nada de esto hubiera sido posible. Especialmente a mis padres y hermano por el apoyo que me brindaron, por la formación, por fomentar en mí el deseo de saber, de conocer lo novedoso y abrirme las puertas al mundo ante mi curiosidad insaciable. Esto es también vuestro premio.

Gracias a todos.

Resumen

El análisis de imágenes hiperespectrales en observación remota de la Tierra constituye una línea de investigación muy activa, con numerosas contribuciones en la literatura científica reciente. Debido a la resolución espacial disponible en los sensores de observación remota de la Tierra y a la forma en que se presentan los materiales en la naturaleza, la mayor parte de los píxeles registrados por el sensor constituyen una mezcla de diferentes sustancias puras a nivel subpíxel. Para solucionar este problema, una de las técnicas más ampliamente utilizadas es el desmezclado espectral, que comprende dos etapas: 1) extracción de firmas espectrales puras (*endmembers*), y 2) estimación de la abundancia de dichos endmembers a nivel subpíxel. Ambas etapas son muy costosas desde el punto de vista computacional, lo cual supone un importante inconveniente en aplicaciones que requieren una respuesta en tiempo real, tales como monitorización y seguimiento de incendios, prevención y seguimiento de desastres naturales, vertidos químicos y otros tipos de contaminación ambiental, etc.

En este trabajo de Tesis Doctoral, se ha diseñado, implementado y validado la cadena completa de desmezclado espectral para el procesamiento a bordo de imágenes hiperespectrales de la superficie terrestre mediante hardware reconfigurable. El diseño propuesto en este trabajo ha sido desarrollado en plataformas reconfigurables tipo Field Programmable Gate Array (FPGA) utilizando el lenguaje VHDL para su especificación. La implementación propuesta proporciona resultados muy prometedores, tanto desde el punto de vista de su precisión a la hora de identificar endmembers y calcular su abundancia, como desde el punto de vista del rendimiento obtenido, proporcionando factores de aceleración (*speedups*) que nos permiten dar una

respuesta en tiempo real. Estos resultados suponen un incremento muy notable del rendimiento computacional del algoritmo con respecto a su ejecución serie en un PC de última generación.

Palabras clave: Imagen Hiperespectral, Hardware Reconfigurable, Extracción de Endmembers, Pixel Purity Index (PPI), N-finder (N–FINDR), Desmezclado Espectral, Image Space Reconstruction Algorithm (ISRA).

Abstract

Remotely sensed hyperspectral imaging is a very active area of research, with numerous contributions in recent scientific literature. Due to the available spatial resolution of the sensors in remote sensing of the Earth and how the materials appear in nature, most of the pixels collected by hyperspectral imaging instruments are in fact a mixture of different underlying substances. To solve this problem, one of the most widely used approaches for analyzing hyperspectral images is spectral unmixing, which comprises two stages: 1) extraction of pure spectral signatures (*endmembers*), and 2) estimation of endmember fractional abundances at sub–pixel level. Both stages are computationally complex, which is a serious drawback in applications which require a response in near real–time, such as forest fire monitoring and tracking, disaster management and prevention, oil spill detection, etc.

In this Thesis work, the complete chain of spectral unmixing has been designed, implemented and validated to process onboard hyperspectral images of the earth's surface using reconfigurable hardware. The design proposed in this work has been developed in reconfigurable platforms like Field Programmable Gate Arrays (FPGAs) using the VHDL language for its specification. Our proposed implentation exhibits very promising results in terms of endmember extraction and fractial abundances accuracy and performance, with speedups that allow us to respond in near real–time. This results represents a tremendous increase of performance with regards to the serial version of the same algorithm, implemented in a latest-generation desktop PC.

Keywords: Hyperspectral Image, Reconfigurable Hardware, Endmember Extraction, Pixel Purity Index (PPI), N-finder (N–FINDR), Spectral

Unmixing, Image Space Reconstruction Algorithm (ISRA).

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Motivaciones y objetivos

Si supiese qué es lo que estoy haciendo,
no lo llamaría investigación, ¿verdad?

Albert Einstein.

Los avances en la tecnología de los sensores están revolucionando la forma en que los datos obtenidos en observación remota de la Tierra son recogidos, gestionados y analizados. La incorporación de sensores de última generación en plataformas aéreas y en satélites está produciendo un flujo casi continuo de datos de alta dimensionalidad. Como consecuencia de este enorme aumento en la cantidad de información recopilada han surgido nuevos desafíos en el procesamiento de estos datos. En particular, muchas aplicaciones actuales y futuras de observación remota en campos como Ciencias de la Tierra, Ciencia Espacial y próximamente en Exploración Espacial, requieren capacidades de procesamiento en tiempo real. Algunos ejemplos son los estudios medioambientales, las aplicaciones militares, el seguimiento y la vigilancia de peligros tales como incendios forestales, vertidos tóxicos y otros tipos de contaminación química y biológica. Uno de los ejemplos más relevantes de datos de alta dimensionalidad en observación remota de la Tierra son las imágenes hiperespectrales, en el que un espectrómetro de imagen recoge cientos o incluso miles de mediciones (en varios canales de longitud de onda) para la misma área de la superficie de la Tierra. Las escenas proporcionadas por estos sensores se denominan a menudo "cubos de datos", para denotar la

extremadamente alta dimensionalidad de los datos.

La imaginería hiperespectral se refiere a la medición, análisis e interpretación de los espectros obtenidos de un determinado lugar (u objeto específico) a corta, media o larga distancia por un sensor aerotransportado o en satélite. El concepto de imagen hiperespectral se originó en el Jet Propulsion Laboratory de la NASA[1] (National Aeronautics and Space Administration) en California, que desarrolla instrumentación como el Airborne Imaging Spectrometer (AIS), llamado después AVIRIS por Airborne Visible Infra-Red Imaging Spectrometer. Actualmente este sistema permite cubrir un ancho de banda desde 0.4 a 2.5 μm con más de doscientos canales espectrales, con una resolución espectral nominal de 10 nm. Como resultado, cada píxel (considerado como un vector) recogido por un sensor hiperespectral puede ser visto como una *firma espectral* o "huella digital" de los materiales subyacentes en el píxel.

Recientemente se han desarrollado varias herramientas de análisis para el procesamiento de datos hiperespectrales, que abarcan temas como reducción de la dimensionalidad, clasificación, compresión de datos o análisis de mezclas espectrales. El supuesto básico que rige a las técnicas de clasificación es que cada píxel contiene el espectro de un único material. Sin embargo, si la resolución espacial del sensor no es lo suficientemente grande para separar los distintos materiales, estos pueden ocupar conjuntamente un solo píxel y el resultado de la medición del espectro será un píxel mezcla, es decir, una composición de espectros puros individuales.

Para hacer frente a este problema, las técnicas de análisis de mezcla espectral primeramente identifican una colección de componentes con firmas espectralmente puras, llamados *endmembers* en la terminología del análisis hiperespectral y a continuación expresan el espectro de cada píxel mezcla como una combinación lineal de endmembers ponderados por *fracciones de abundancia* que indican la proporción de cada endmember en el píxel. El problema de desmezclado se formula en términos de un sistema determinado de ecuaciones que, partiendo del conjunto correcto de endmembers, permite la determinación de las fracciones de abundancia aparentes de cada endmember a través de un proceso de inversión numérica.

[1] http://www.jpl.nasa.gov

La ejecución de este tipo de algoritmos es lenta. Para atender las necesidades de cómputo establecidas por muchas aplicaciones de tiempo crítico, recientemente se ha investigado la incorporación de la computación de alto rendimiento en los modelos de misiones de observación remota de la Tierra. Hasta la fecha, las técnicas tradicionales en la literatura para abordar este problema han optado por soluciones basadas en el uso de clusters y sistemas multiprocesador. La computación cluster, a pesar de su adaptabilidad al problema del tratamiento de datos hiperespectrales (especialmente, cuando dichos datos se encuentran almacenados en un repositorio de datos en tierra), presenta problemas en cuanto al procesamiento de los datos en tiempo real dado el alto coste y elevados requerimientos en cuanto a espacio, peso y consumo (denominado *payload* en misiones de observación remota de la Tierra). Por otra parte, la instalación de un cluster suele llevar asociada la disponibilidad de un número elevado de ordenadores interconectados entre sí para que compartan el procesamiento de datos a través de sus procesadores, lo cual hace incrementar la velocidad de ejecución y procesamiento de las aplicaciones; sin embargo, cada nodo (u ordenador) lleva ligado un precio y una serie de requerimientos en cuanto a espacio y consumo que alejan esta aproximación de las características requeridas en cuanto a *payload* en misiones reales de observación remota de la Tierra.

La flexibilidad también es un parámetro importante a tratar en aplicaciones aeroespaciales, ya que el espacio disponible es muy limitado. En este sentido, Las unidades de procesamiento gráfico domésticas (GPUs) ofrecen una solución de bajo peso que cumple con los requisitos de *payload* de las misiones, aunque el consumo de energía se considera elevado para este tipo de aplicaciones, motivo principal por el que no están certificadas para operar en el espacio.

La reconfiguración, el tamaño compacto y la elevada potencia de cálculo de las Field Programmable Gate Arrays (FPGAs), las hacen particularmente atractivas para su explotación en aplicaciones de observación remota de la Tierra que requieran una respuesta en tiempo real. Además, existen FPGAs endurecidas para radiación certificadas para operar en condiciones espaciales. Las FPGAs híbridas desarrolladas recientemente, combinan una mayor área reconfigurable junto con microprocesadores y recursos de memoria empotra-

dos. Estos sistemas hardware/software estrechamente acoplados combinan la flexibilidad de los microprocesadores tradicionales con la potencia de cálculo de un hardware específico, dando lugar a nuevas arquitecturas para misiones de observación remota de la Tierra.

1.1. Aportaciones de esta tesis

La propuesta de Tesis Doctoral que se plantea en esta memoria se centra en el procesamiento a bordo de imágenes hiperespectrales de la superficie terrestre mediante hardware reconfigurable. Trata de resolver dos de los problemas más importantes planteados en este contexto, como son la alta dimensionalidad de los datos y el problema de la mezcla espectral. Tradicionalmente, las técnicas adoptadas para abordar estos problemas han optado por soluciones basadas en el uso de clusters y sistemas multiprocesador, sin embargo, el gran volumen que ocupan junto con el elevado peso y consumo, hacen que esta solución sea inviable para un sistema de procesamiento a bordo. Para solucionar estos problemas relativos a coste, consumo y peso, y ofrecer además mejoras sustanciales en cuanto al tiempo de procesamiento, en la presente memoria de Tesis Doctoral se propone una alternativa basada en un nuevo modelo de tratamiento de imágenes hiperespectrales mediante la utilización de FPGAs. Conviene destacar que, con una sola FPGA, pueden llegar a obtenerse mejoras notables a la hora de procesar cálculos de tipo científico, como es el caso de los algoritmos de tratamiento de imágenes hiperespectrales, a un coste razonable y ocupando además un espacio mínimo.

En este trabajo de investigación se han estudiado, diseñado, implementado y validado dos de los algoritmos más utilizados para la extracción de endmembers (Pixel Purity Index –PPI– y N–FINDR) y uno para la estimación de abundancias (Image Space Reconstruction Algorithm –ISRA–). Además, se aprovecha la reconfigurabilidad de las FPGAs para desarrollar una cadena completa de desmezclado espectral.

La consecución del objetivo general anteriormente mencionado se lleva a cabo en la presente memoria abordando una serie de objetivos específicos, los cuales se enumeran a continuación:

- Adquirir conocimientos previos sobre análisis hiperespectral (imagen hiperespectral, formatos de los datos, representación de datos, presentación de resultados, etc.), necesarios para poder llevar a cabo el estudio de los distintos algoritmos.

- Estudiar en profundidad las principales características de las técnicas objeto del análisis y sus parámetros de entrada. Este estudio comprende la adquisición de los conocimientos necesarios sobre todas las técnicas evaluadas, así como de otras técnicas de clasificación comúnmente utilizadas en el ámbito del análisis hiperespectral.

- Con respecto al algoritmo PPI:

  - Establecer un estudio sobre el funcionamiento del algoritmo PPI para establecer las distintas estrategias de paralelización y utilizar la más adecuada para su diseño.
  - Implementar el algoritmo para su ejecución en FPGA optimizando el tiempo de ejecución, comparándolo con una versión software equivalente con la versión original semisupervisada (disponible en software comercial) y con una implementación en FPGA recientemente desarrollada.
  - Ejecutar el algoritmo sobre una serie de datos reales provenientes de sensores hiperespectrales que permitan conocer la precisión de los endmembers extraídos en la ejecución del algoritmo y el rendimiento obtenido.
  - Realizar un estudio comparativo sobre los resultados obtenidos.

- Con respecto al algoritmo N–FINDR:

  - Establecer un estudio sobre el funcionamiento del algoritmo N–FINDR para encontrar el diseño más adecuado para su implementación.
  - Implementar el algoritmo para su ejecución en FPGA optimizando el tiempo de ejecución, comparándolo con una versión software equivalente.

    - Ejecutar el algoritmo sobre una serie de datos reales provenientes de sensores hiperespectrales que permitan conocer la precisión que se ha logrado en la ejecución del algoritmo y el rendimiento obtenido.
    - Interpretar los resultados experimentales realizados.

- Con respecto al ISRA:

    - Establecer un estudio sobre el funcionamiento del ISRA para extraer el grado de paralelismo inherente que presenta y utilizarlo para su implementación.
    - Implementar el algoritmo para su ejecución en FPGA optimizando el tiempo de ejecución, comparándolo con una versión software equivalente.
    - Ejecutar el algoritmo sobre una serie de datos reales provenientes de sensores hiperespectrales que permitan conocer la precisión que se ha logrado en la ejecución del algoritmo y el rendimiento obtenido.
    - Evaluar la implementación propuesta a partir de los experimentos realizados.

- Realizar una comparativa entre los algoritmos PPI y N–FINDR propuestos para determinar cual de los dos es más conveniente utilizar en la cadena completa de desmezclado espectral.

- Utilizar la reconfigurabilidad de las FPGAs para implementar la cadena completa de desmezclado espectral.

- Obtener conclusiones a partir del estudio cuantitativo y comparativo realizado, y plantear posibles trabajos futuros.

1.2. Organización de esta tesis

Teniendo presentes los anteriores objetivos concretos, se procede a describir la organización del resto de esta memoria, estructurada en una serie de capítulos cuyos contenidos se describen a continuación:

- **Análisis hiperespectral**: En este capítulo se describen los conceptos fundamentales relacionados con las imágenes hiperespectrales, los sensores de adquisición de este tipo de imágenes, el análisis de imágenes hiperespectrales mediante el desmezclado espectral y el procesamiento de este tipo de datos, enfatizando la necesidad de la computación de altas prestaciones en este campo.

- **Hardware reconfigurable**: A lo largo de este capítulo se presenta la evolución sufrida desde los primeros dispositivos FPGAs hasta las plataformas híbridas que integran además, entre otros elementos, microprocesadores de propósito general y Application Specific Integrated Circuits (ASICs) en el mismo chip. También se ha incluido una sección donde se presentan los distintos modos de configuración de los dispositivos reconfigurables, siendo estas distintas modalidades uno de los factores más característicos de los mismos. Finalmente, se expone el papel del hardware reconfigurable en sistemas empotrados y en misiones de observación remota de la Tierra.

- **Algoritmo Pixel Purity Index (PPI)**: En este capítulo se ofrece un estudio del algoritmo PPI para la extracción de endmembers, las distintas estrategias de paralelización con el fin de establecer el diseño adecuado, los detalles sobre la implementación en FPGA y finalmente, los resultados experimentales junto con algunas conclusiones.

- **Algoritmo N–finder (N–FINDR)**: En este trabajo de investigación se ha desarrollado la primera implementación en FPGA del algoritmo N–FINDR para la extracción de endmembers, que ilustra las ventajas y desventajas de la tecnología reconfigurable en el contexto de las misiones aéreas y espaciales de observación remota de la Tierra. Tras un minucioso estudio del algoritmo y de las propiedades de los determinantes, se presenta un diseño para la implementación en FPGA del algoritmo. Finalmente, se describe de forma detalla la implementación propuesta junto con los resultados experimentales.

- **Image Space Reconstruction Algorithm (ISRA)**: En este capítulo se proporciona una descripción detallada del Image Space Reconstruction Algorithm (ISRA), que sigue un modelo lineal de mezcla para

la estimación de las fracciones de abundancia no negativas en píxeles mezcla, su implementación en FPGA y la validación de la arquitectura hardware paralela propuesta mediante los resultados experimentales obtenidos.

- **Cadena completa de desmezclado espectral**: En este capítulo se presenta un diseño para la implementación en FPGA de la cadena completa de desmezclado espectral. Tras una sección en la que se realiza una comparativa entre los algoritmos PPI y N–FINDR propuestos para determinar cuál de los dos es más conveniente utilizar, se muestra la implementación en FPGA de la cadena de desmezclado espectral, utilizando la reconfigurabilidad de estos dispositivos, y los resultados experimentales.

- **Conclusiones y trabajo futuro**: Este capítulo está dedicado a resumir los principales logros alcanzados en la presente memoria y a mostrar las conclusiones derivadas. Además, el capítulo sugiere un conjunto de líneas de trabajo que pueden ser abordadas en futuros trabajos.

- **Publicaciones generadas**: En este capítulo se presentan las publicaciones mediante las que se ha divulgado el trabajo de investigación realizado.

- **Bibliografía**: El trabajo incluye una exhaustiva revisión bibliográfica que permitirá al lector profundizar en mayor detalle en los diferentes aspectos teóricos y relativos a la implementación de los algoritmos descritos en la presente memoria.

Capítulo 2

Análisis hiperespectral

Una imagen vale más que mil palabras.
Pero ocupa mucha más memoria.

Anónimo

Durante gran parte de la década pasada, el análisis de imágenes hiperespectrales ha sido un área activa en investigación y desarrollo, pero dichas imágenes han estado disponibles sólo para los investigadores. Con la reciente aparición de sistemas comerciales aerotransportados de adquisición de este tipo de imágenes y el inminente lanzamiento de satélites, el análisis de las mismas pasará a convertirse en la herramienta principal para la observación remota de la Tierra. El análisis de imágenes hiperespectrales encuentra muchas aplicaciones en manejo de recursos, agricultura, exploración minera y monitoreo ambiental, pero su uso efectivo requiere un entendimiento de la naturaleza, las limitaciones de los datos y de las distintas estrategias para procesarlas e interpretarlas. Este capítulo intenta proveer una introducción a los conceptos fundamentales en el campo del análisis hiperespectral.

2.1. Concepto de imagen hiperespectral

El asentamiento de la tecnología hiperespectral en aplicaciones de observación remota de la superficie terrestre ha dado como resultado el desarro-

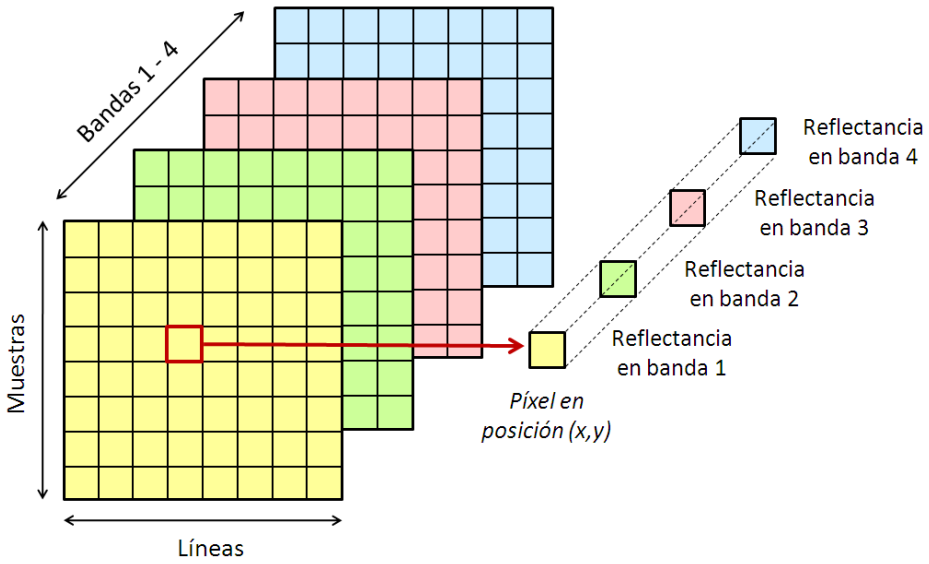


Figura 2.1: Concepto de imagen hiperespectral.

llo de instrumentos de medida de muy elevada resolución en los dominios espacial y espectral, sin olvidarnos del dominio temporal. Los sensores hiperespectrales adquieren imágenes digitales en una gran cantidad de canales espectrales muy cercanos entre sí, obteniendo, para cada porción de la escena o píxel, una *firma espectral* o "huella dactilar" característica de cada material [Lan02].

El resultado de la toma de datos por parte de un sensor hiperespectral sobre una determinada escena puede ser representado en forma de cubo de datos, con dos dimensiones para representar la ubicación espacial de un píxel (generalmente denominadas líneas y muestras), y una tercera dimensión que representa la singularidad espectral de cada píxel en diferentes longitudes de onda [Cha03]. La Figura 2.1 muestra la estructura de una imagen hiperespectral donde el eje X es el indicador de las líneas, el eje Y es el indicador de las muestras y el eje Z es el número de banda, es decir, la longitud de onda de esa banda (canal).

La capacidad de observación de los sensores hiperespectrales permite la obtención de una firma espectral detallada para cada píxel de la imagen, dada por los valores de reflectancia adquiridos por el sensor en diferentes

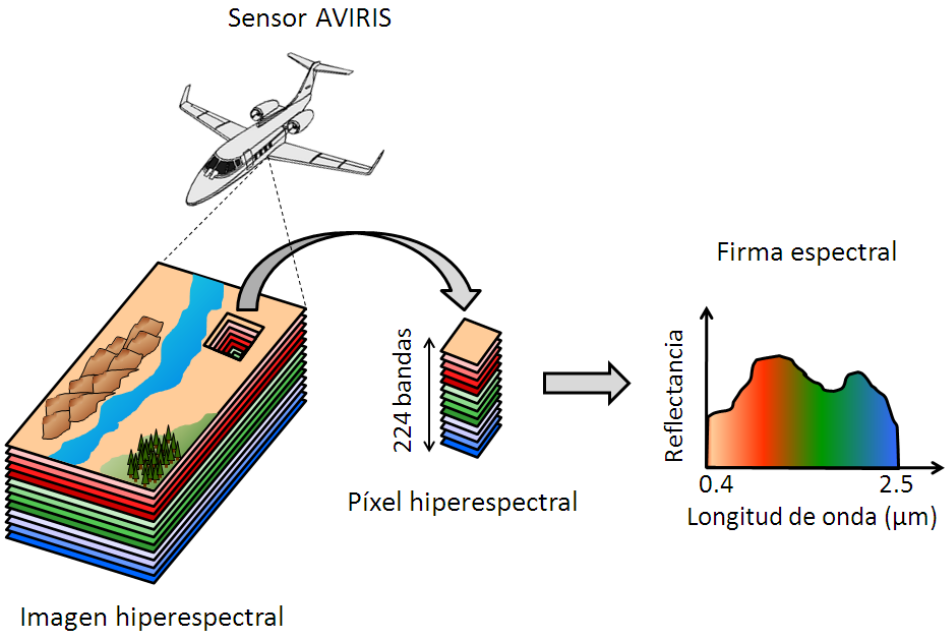


Figura 2.2: Adquisición de una imagen hiperespectral por el sensor AVIRIS.

longitudes de onda, lo cual permite una caracterización muy precisa de la superficie de nuestro planeta [JL98]. Como ejemplo ilustrativo, la Figura 2.2 muestra el procedimiento de análisis hiperespectral mediante un sencillo diagrama, en el que se ha considerado como ejemplo el sensor AVIRIS (Airborne Visible Infra-Red Imaging Spectrometer), desarrollado por el Jet Propulsion Laboratory de la NASA, el cual cubre el rango de longitudes de onda entre 0.4 y 2.5 μm utilizando 224 canales y una resolución espectral de aproximadamente 10 nm [GES+98].

Como puede apreciarse en la Figura 2.2, es el elevado número de bandas de este sensor lo que permite la obtención de una firma espectral detallada del píxel. Conviene destacar que, en este tipo de imágenes, es habitual la existencia de mezclas a nivel de subpíxel, por lo que a grandes rasgos podemos encontrar dos tipos de píxeles en estas imágenes: píxeles puros y píxeles mezcla [PC06]. Se puede definir un píxel mezcla como aquel en el que cohabitan diferentes materiales. Estos tipos de píxeles son los que constituyen la mayor parte de la imagen hiperespectral, en parte, debido a que este fenómeno es independiente de la escala considerada ya que tiene lugar

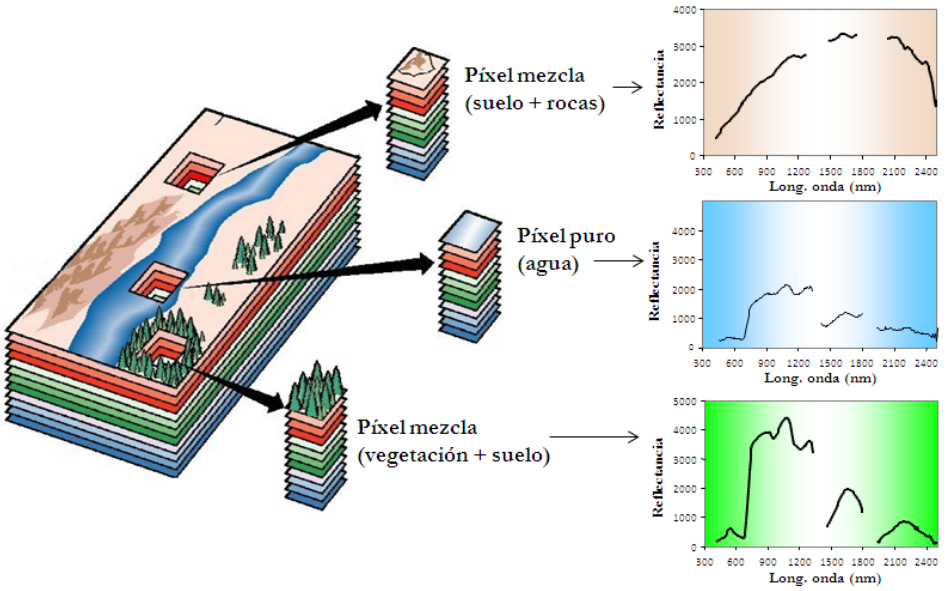


Figura 2.3: Tipos de píxeles en imágenes hiperespectrales.

incluso a niveles microscópicos [PMPP02]. La Figura 2.3 muestra un ejemplo del proceso de adquisición de píxeles puros (a nivel macroscópico) y píxeles mezcla en imágenes hiperespectrales.

El desarrollo tecnológico introducido por la incorporación de sensores hiperespectrales en plataformas de observación remota de la superficie terrestre de última generación ha sido particularmente notable durante los últimos años. En este sentido, conviene destacar que dos de las principales plataformas de tipo satélite que se encuentran en funcionamiento en la actualidad: Earth Observing-1 de NASA[1] y ENVISAT de la Agencia Espacial Europea[2], llevan incorporados sensores de este tipo, permitiendo así la posibilidad de obtener imágenes hiperespectrales de la práctica totalidad del planeta de manera casi continua. A pesar de la gran evolución en los instrumentos de observación remota de la superficie terrestre, la evolución en las técnicas de análisis de los datos proporcionados por dichos sensores no ha sido tan notoria. En particular, el desarrollo de técnicas de análisis hiperespectral avanzadas, capaces de aprovechar totalmente la gran canti-

[1] http://eo1.gsfc.nasa.gov
[2] http://envisat.esa.int

dad de información espacial y espectral presente en imágenes hiperespectrales, constituye un objetivo de gran interés para la comunidad científica [Lan03, JRMRD+05, PMPP02].

2.2. Espectrometría de imágenes

Las imágenes hiperespectrales son producidas por instrumentos llamados espectrómetros de imágenes. El desarrollo de estos sensores complejos ha supuesto la convergencia de dos tecnologías relacionadas pero distintas: la espectroscopia y la observación remota de la Tierra y de la superficie de los planetas.

La espectroscopia es el estudio de la luz que es emitida o reflejada por los materiales y su variación de energía con la longitud de onda [Cla99]. Al ser aplicada al campo óptico de la observación remota de la Tierra, la espectroscopia se basa en el espectro de la luz solar que se refleja difusamente (dispersada) por los materiales terrestres. Los instrumentos llamados espectrómetros (o espectrorradiómetros) son utilizados para realizar mediciones de campo o de laboratorio de la luz reflejada desde un material testigo. Un elemento de dispersión óptica tal como una rejilla o prisma en el espectrómetro separa esta luz dentro de muchas longitudes de onda adyacentes y estrechas, y la energía en cada banda es medida por un detector diferente (ver Figura 2.4). Al usar cientos o miles de detectores, los espectrómetros pueden realizar mediciones espectrales de bandas tan próximas como 0.01 μm sobre un rango amplio de longitudes de onda, típicamente al menos de 0.4 a 2.4 μm (rangos de longitudes de onda del visible al infrarrojo medio).

Los dispositivos de imágenes remotas están diseñados para concentrar y medir la luz reflejada desde varias áreas adyacentes de la superficie terrestre. En muchos dispositivos de imágenes digitales, se realizan mediciones secuenciales de pequeñas áreas siguiendo un patrón geométrico a medida que la plataforma del sensor se mueve y se requiere un procesamiento posterior para juntarlas en una sola imagen. Hasta hace poco, los dispositivos de imágenes estaban restringidos a uno o unos pocos relativamente escasos anchos de banda de longitud de onda por las limitaciones de diseño del detector y los requerimientos de almacenamiento de datos, transmisión y procesamien-

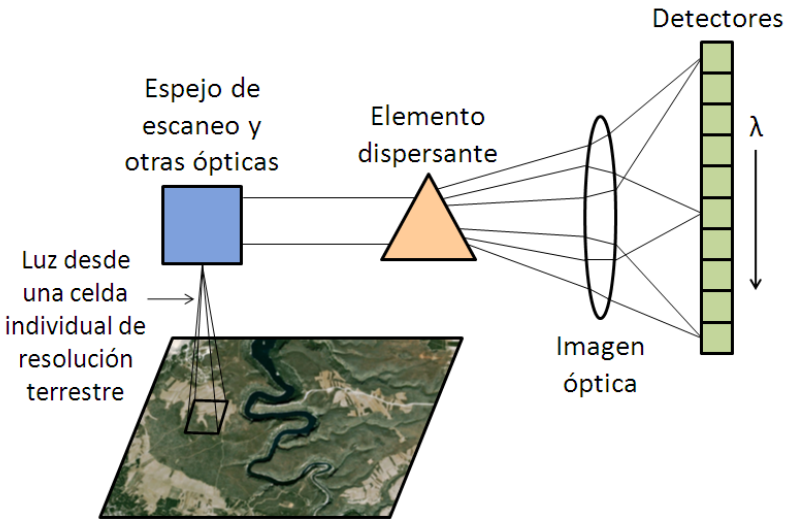


Figura 2.4: Diagrama esquemático de los elementos básicos de espectrometría de imágenes.

to. Este tipo de sensores son conocidos como sensores multiespectrales y sus imágenes tan solo contienen unas pocas decenas de bandas. Los avances recientes en estas áreas han permitido el diseño de dispositivos de imágenes que tienen rangos espectrales y resoluciones comparables con los espectrómetros de campo.

En espectroscopia de luz reflejada la propiedad fundamental que queremos obtener es la *reflectancia espectral*: el porcentaje de la energía reflejada sobre la energía incidente como una función de la longitud de onda. La reflectancia varía con la longitud de onda para la mayoría de los materiales ya que la energía en ciertas longitudes de onda es dispersada o absorbida en diferentes grados. Estas variaciones de reflectancia son evidentes cuando comparamos las curvas de reflectancia espectral (gráficos de reflectancia frente longitud de onda) para diferentes materiales, como en la Figura 2.5. Las desviaciones pronunciadas hacia abajo de las curvas espectrales marcan los rangos de longitud de onda en los cuales los materiales selectivamente absorben la energía incidente. Estas características son comúnmente llamadas *bandas de absorción*. La forma general de una curva espectral y la posición e intensidad de las bandas de absorción en muchos casos pueden ser utilizados

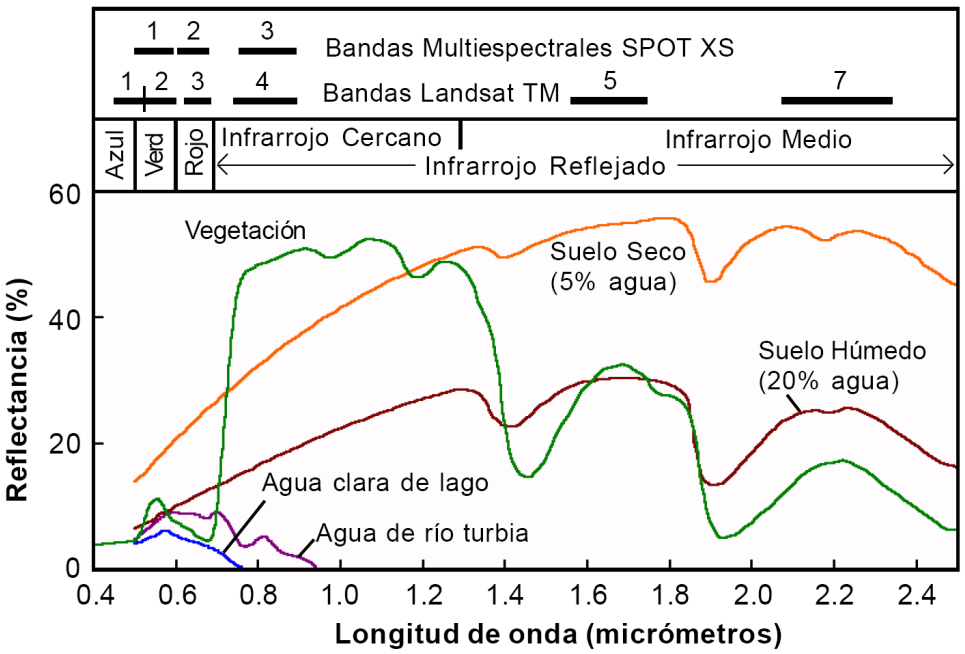


Figura 2.5: Curvas de reflectancia espectral para diferentes materiales.

para identificar y discriminar materiales diferentes. Por ejemplo, la vegetación tiene reflectancias más altas en el rango del infrarrojo cercano y más baja reflectancia en la luz roja que los suelos.

La Figura 2.5 muestra las curvas representativas de reflectancia espectral para varios materiales terrestres comunes desde los rangos de luz visible hasta el espectro infrarrojo reflejado [BDIE99]. Además, se muestran las bandas espectrales usadas por varios sensores remotos de satélites multiespectrales (SPOT XS y Landsat TM) para compararlas. La reflectancia es una cantidad sin unidades con valores en el rango de 0 a 1.0, o también puede expresarse como un porcentaje, como en este gráfico. Cuando se realizan las medidas espectrales de un material de test en el campo o en el laboratorio, se requieren también los valores de la energía incidente para calcular la reflectancia del material. Estos valores son medidos directamente o derivados desde medidas de luz reflejada (bajo las mismas condiciones de iluminación que el material de test) para un material de referencia estándar con una reflectancia espectral conocida.

2.3. Radiancia y reflectancia

De las discursiones de las páginas anteriores, debería quedar claro que la reflectancia espectral es una propiedad de las características del terreno que podría medirse precisa y exactamente utilizando un sensor hiperespectral de avión o satélite. Sin embargo, la reflectancia espectral de los materiales de superficie es sólo uno de los factores que afectan a los valores medidos. Además de la reflectancia de la superficie, la radiancia espectral (cantidad de luz emitida en todo el espectro de frecuencias) medida por un sensor remoto depende del espectro de entrada de energía solar, las interacciones de esta energía durante sus travesías hacia abajo y arriba a través de la atmósfera [GB97], la geometría de iluminación de las áreas individuales sobre el terreno y las características del sistema del sensor. Estos factores adicionales no sólo afectan nuestra habilidad de recuperar valores de reflectancia espectral exactos para los rasgos de la superficie, sino que también introducen variabilidades adicionales dentro de la escena que obstaculizan las comparaciones individuales entre píxeles de la imagen. A continuación, se discuten en mayor detalle estos factores.

La Figura 2.6 muestra una curva típica de irradiancia solar desde el límite de la atmósfera terrestre. La energía solar entrante varía ampliamente con la longitud de onda, con un máximo en el rango de la luz visible. El espectro de la energía solar entrante a la hora en que la imagen fue adquirida debe ser conocida, asumida, o derivada indirectamente desde otras mediciones para convertir los valores de imagen de radiancia a valores de reflectancia.

La cantidad de energía reflejada por un área sobre la superficie depende de la cantidad de energía solar que ilumine el área, que a su vez depende del *ángulo de incidencia*: el ángulo entre la trayectoria de la energía entrante y una recta perpendicular a la superficie terrestre (ver Figura 2.7). Específicamente, la energía recibida en cada longitud de onda E_g varía con el coseno del ángulo de incidencia θ: $E_g = E_o \cdot cos\ \theta$, donde E_o es la cantidad de energía entrante. La energía recibida por cualquier área de superficie por lo tanto varía a medida que la altura del sol cambia con la hora del día y la estación del año. Si el terreno no es plano, la energía recibida también varía instantáneamente a lo largo de la escena por las diferencias en pendiente y

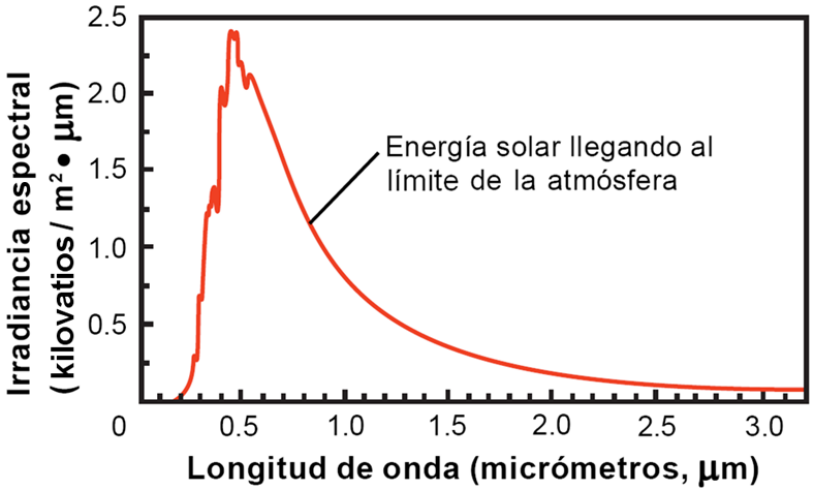


Figura 2.6: Irradiancia solar desde el límite de la atmósfera terrestre frente a longitud de onda.

dirección.

La cantidad de iluminación recibida por un área también puede verse reducida por sombras. Las sombras moldeadas por los rasgos topográficos o nubes pueden afectar a áreas que incluyan muchos píxeles contiguos en la imagen. Los árboles, las hileras de cultivos, afloramientos rocosos u otros objetos pequeños también pueden moldear las sombras que se confinan en un píxel individual. Ambos tipos de sombras tienen el efecto de bajar la cantidad de brillo a lo largo de todas las longitudes de onda de los píxeles afectados.

Incluso una atmósfera relativamente clara interactúa con la energía solar entrante y reflejada. Para ciertas longitudes de onda, estas interacciones reducen la cantidad de energía entrante que alcanzan el terreno y además reducen la cantidad de energía reflejada que alcanza un sensor de avión o de satélite. La transmitancia de la atmósfera se reduce por la absorción de ciertos gases y por la dispersión de moléculas de gas y partículas. Estos efectos combinados producen la curva de transmitancia ilustrada en la Figura 2.8. Los rasgos de absorción pronunciada cerca de 1.4 y 1.9 μm causada por el vapor de agua y el dióxido de carbono, reducen la energía incidente y reflejada casi completamente, esta información útil tan pequeña puede ser obtenida

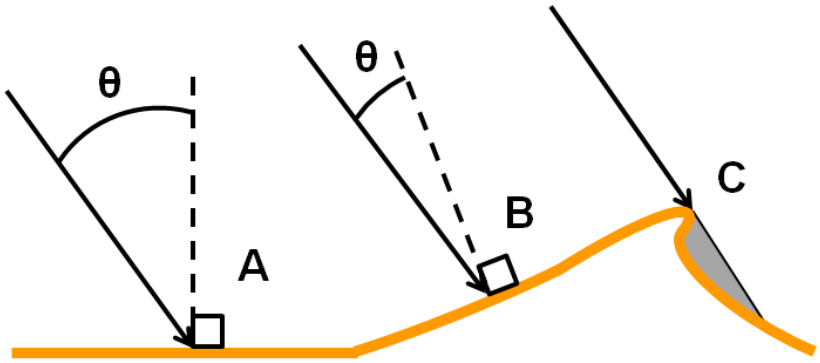


Figura 2.7: Las diferencias de iluminación pueden llegar desde diferentes ángulos de incidencia θ tanto para A y B, o desde el sombreado C.

desde bandas de imagen en estas regiones. En esta curva no se muestra el efecto de la luz dispersada hacia arriba por la atmósfera. Esta luz dispersada es añadida a la radiancia medida por el sensor en las longitudes de onda del visible y del infrarrojo cercano, y se denomina *path radiance* (radiancia de trayectoria). Los efectos atmosféricos también pueden diferir entre áreas en una escena individual si las condiciones atmosféricas varían espacialmente o si hay diferencias significativas de elevación terrestre que varían la longitud de la trayectoria de la radiación a través de la atmósfera.

Un sensor convierte la radiancia detectada en cada canal de longitud de onda a una señal eléctrica que es escalada y cuantificada dentro de valores enteros discretos que representan los valores de radiancia "codificados". Las variaciones entre detectores de una serie, como también los cambios temporales en los detectores, pueden requerir mediciones crudas para ser escaladas y producir valores comparables.

Para comparar directamente la espectrometría de la imagen hiperespectral con la referencia de espectrometría de reflectancia, los valores codificados de radiancia en la imagen deben ser convertidos a reflectancia. Una conversión global debe considerar la fuente solar espectral, los efectos de luminosidad debidos al ángulo solar y la topografía, la transmisión atmosférica, y la ganancia del sensor. En términos matemáticos, la reflectancia espectral del terreno es multiplicada (sobre la base de longitud de onda por longitud de onda) por estos efectos para producir el espectro de radiancia medido.

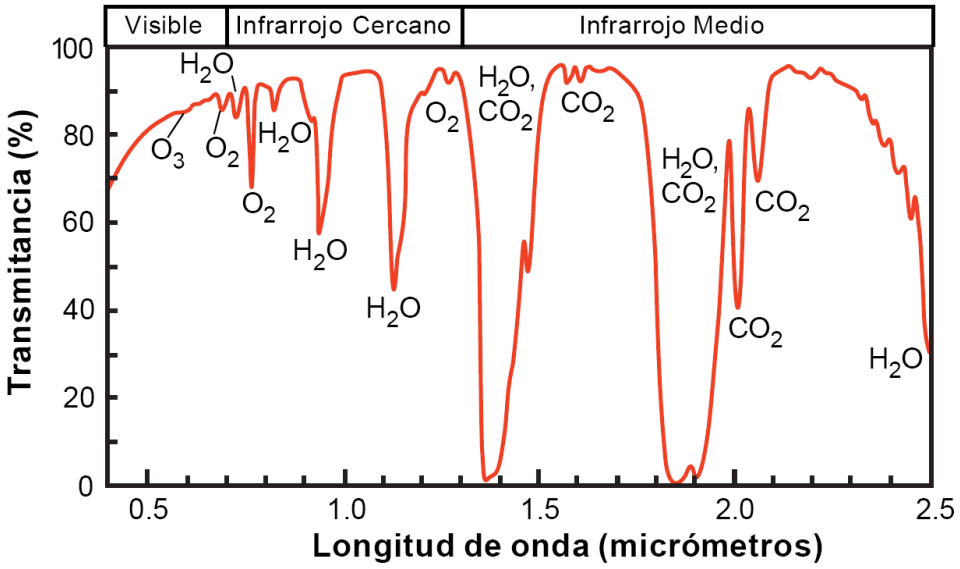


Figura 2.8: Efectos atmosféricos.

Otros dos factores contribuyen de una manera aditiva a la espectrometría de la radiancia: el *offset* del sensor (ruido de instrumentos internos) y la trayectoria de radiancia debida a la dispersión atmosférica. Algunas estrategias sólo usan información sacada desde la imagen, mientras que otras requieren varios grados de conocimiento de las propiedades de reflectancia de superficie y las condiciones atmosféricas en la fecha en que la imagen fue adquirida.

2.4. Bibliotecas espectrales

Varias bibliotecas de reflectancia espectral de materiales naturales o hechos por el hombre están disponibles para el uso público. Estas bibliotecas proveen una fuente de referencia espectral que puede ayudar a la interpretación de imágenes hiperespectrales.

- **Biblioteca espectral USGS**. El laboratorio de espectrometría del United States Geological Survey[3] en Denver, Colorado, ha compilado una biblioteca de cerca de 500 reflectancias espectrales de minerales y

[3]http://www.usgs.gov

unas pocas plantas sobre las longitudes de onda en el rango de 0.2 a 3.0 μm. Esta biblioteca es accesible online en http://speclab.cr.usgs.gov /spectral.lib04/spectral-lib04.html. Se puede realizar la búsqueda de una espectrometría individual online o descargar la biblioteca completa.

- **Biblioteca espectral ASTER**. Esta biblioteca se ha hecho accesible por la NASA como parte del programa instrumental Advanced Spaceborne Termal Emision and Reflectance Radiometer[4] (ASTER). Esta biblioteca incluye compilaciones espectrales del Jet Propulsion Laboratory de la NASA, Johns Hopkins University y United States Geological Survey. La biblioteca espectral ASTER actualmente contiene cerca de 2000 espectrometrías, incluyendo minerales, rocas, suelos, materiales hechos por el hombre, agua, y nieve. La mayoría de las coberturas espectrales comprenden las longitudes de onda entre 0.4 y 14 μm. La biblioteca se encuentra disponible en la web http://speclib.jpl.nasa.gov. Se puede realizar la búsqueda de una espectrometría por categoría, ver un gráfico espectral para cualquier espectro recuperado, y descargar los datos individuales de la espectrometría como archivo de texto.

2.5. Sensores hiperespectrales

Los inicios de la teledección u observación remota de la Tierra se remontan a los primeros sistemas de adquisición de fotografías aéreas de la superficie terrestre a través de cámaras montadas en globos y zepelines, hace más de 150 años. Aunque con posteridad a la Segunda Guerra Mundial la fotografía aérea experimentó un notorio avance tecnológico, no fue sino hasta inicios de la década de los 70 –el gobierno de los EE. UU. desclasificó los sensores multiespectrales y lanzó los primeros satélites con propósitos comerciales– cuando la observación remota de la Tierra dio un paso sustancial a una nueva era de aplicaciones. Durante las décadas siguientes, el rastreo y monitoreo continuo de los recursos naturales de la superficie terrestre fue posible mediante el lanzamiento sucesivo de las misiones satelitales

[4] http://asterweb.jpl.nasa.gov

Landsat[5], NOAA[6] (National Oceanic and Atmospheric Administration) y SPOT[7] (Système Pour l′Observation de la Terre), entre otras, cuyos sensores a bordo mejoraron progresivamente sus características técnicas, conduciendo a nuevos procedimientos de procesamiento digital de imágenes y ampliando ostensiblemente el campo de acción de la observación remota de la Tierra.

A inicios de la década de los 90, los avances en la espectrometría dieron origen a la tecnología de sensores hiperespectrales, cuyas imágenes demostraron en un lapso de pocos años una vasta potencialidad de aplicaciones, así como numerosas ventajas sobre aquellas adquiridas por sensores multiespectrales convencionales.

Mientras un sensor multiespectral es capaz de dividir el espectro electromagnético en unas pocas bandas o porciones espectrales (baja resolución espectral), los sensores hiperespectrales son capaces de registrar cientos de bandas contiguas a través de diversas dimensiones del espectro electromagnético (alta resolución espectral), lo que permite contar con una curva detallada del comportamiento espectral de una gran cantidad de coberturas o elementos de la superficie terrestre, posibilitando su identificación y discriminación con gran precisión.

En la actualidad existen misiones comerciales, tanto aéreas como satelitales, destinadas a adquirir imágenes hiperespectrales de la superficie terrestre, siendo el Jet Propulsion Laboratory de la NASA y el Servicio Geológico de EE. UU., las principales entidades que llevan a cabo estas iniciativas. Sin embargo, la adquisición de imágenes de un área geográfica en particular debe ser requerida con antelación por parte del interesado, con el propósito de que las líneas de vuelo o las rutas orbitales de las misiones sean debidamente programadas. Así, el proceso de adquisición de imágenes queda sujeto a fechas y horas específicas, en las que se arriesga a que durante el momento de paso de la plataforma por el área de interés, las condiciones atmosféricas y de iluminación no sean las más apropiadas para la captura de datos. Por otra parte, estos sensores cuentan con resoluciones espaciales de entre 20 a 30 m, que impiden la visualización de detalles espaciales a escala local.

[5]http://landsat.gsfc.nasa.gov
[6]http://www.noaa.gov
[7]http://www.spot.com

Tabla 2.1: Listado de algunas misiones espaciales de observación remota de la Tierra presentes y futuras, que incluyen sensores hiperespectrales.

	EO-1 Hyperion*	**Prisma†**	**EnMAP‡**	**HyspIRI§**
País de origen	USA	Italia	Alemania	USA
Resolución espacial	30 m	5-30 m	30 m	60 m
Tiempo de ciclo	16 días	3/7 días	4 días	18 días
Rango espectral	0.4-2.5 μm	0.4-2.5 μm	0.42-2.45 μm	0.38-2.5 μm
Resolución espectral	10 nm	10 nm	6.5-10 nm	10 nm
Ancho de barrido	7.7 km	30 km	30 km	120 km
Covertura terrestre	Parcial	Completa	Completa	Completa
Lanzamiento	2000	2010	2012	2018
Tiempo de vida	10 años	$\approx$ 6 años	$\approx$ 6 años	$\approx$ 6 años

*http://eo1.gsfc.nasa.gov †http://www.asi.it/en/flash_en/observing/prisma
‡http://www.enmap.org §http://hyspiri.jpl.nasa.gov

Para solventar estas limitaciones, en la actualidad muchas organizaciones y empresas han optado por desarrollar sus propios sistemas de observación remota hiperespectral de la Tierra, comprando un sensor hiperespectral a algunas de las numerosas empresas privadas que existen en el mercado de la electroóptica (Autovision[8], Itres[9], Geophysical and Environmental Research Corporation) y montándolo en una plataforma aérea, con el fin de otorgarle autonomía y flexibilidad espacial y temporal al proceso de adquisición de imágenes, además de posibilitar aplicaciones de alta precisión a escala local, pues estos instrumentos constan de una alta resolución espacial (<1 m). Algunas de las organizaciones que han desarrollado este tipo de sistemas son: Norwegian Crop Research Institute, para aplicaciones de agricultura de precisión; Canada Centre for Remote Sensing[10], para la valoración de impactos ambientales, e INCO Mine Limited, para la prospección de yacimientos minerales.

La Tabla 2.1 resume las misiones espaciales de observación remota de la superficie terrestre con sensores hiperespectrales que ya están en funcionamiento o que se pondrán en marcha en un futuro próximo. Como indica la

[8]http://www.autovision.net
[9]http://www.itres.com
[10]http://www.ccrs.nrcan.gc.ca

tabla, la proliferación de instrumentos basados en satélites para la observación remota hiperespectral de la Tierra es muy notoria, con una resolución espectral muy alta en todos los casos. A continuación describimos brevemente los sensores de los cuales vamos a tomar imágenes hiperespectrales para la validación de los distintos algoritmos desarrollados.

2.5.1. Sensor AVIRIS

AVIRIS es un sensor hiperespectral aerotransportado con capacidades analíticas en las zonas visible e infrarroja del espectro [GK99, BKG95, GP00]. Este sensor está en funcionamiento desde 1987. Fue el primer sistema de adquisición de imágenes capaz de obtener información en una gran cantidad de bandas espectrales estrechas y casi contiguas. AVIRIS es un instrumento pioreno en el mundo de la teledetección que permitió obtener información espectral en 224 canales espectrales contiguos, cubriendo un rango de longitudes de onda entre 0.4 y 2.5 μm, siendo el ancho entre las bandas muy pequeño, aproximadamente 0.01 μm. En 1989, AVIRIS se convirtió en un instrumento aerotransportado. Desde ese momento, se realizan varias campañas de vuelo cada año para tomar datos mediante AVIRIS. El sensor ha realizado tomas de datos en Estados Unidos, Canadá y Europa, utilizando para ello dos plataformas:

- Un avión ER-2 perteneciente al Jet Propulsion Laboratory de la NASA. El ER-2 puede volar a un máximo de 20 km sobre el nivel del mar, a una velocidad máxima de aproximadamente 730 km/h.
- Un avión denominado Twin Otter desarrollado por la compañía canadiense Havilland Canada, capaz de volar a un máximo de 4 km sobre el nivel del mar, a velocidades de 130 km/h.

Algunas de las características más relevantes en cuanto al diseño interno del sensor AVIRIS son las siguientes:

- El sensor utiliza un explorador de barrido que permite obtener un total de 614 píxeles por cada oscilación.

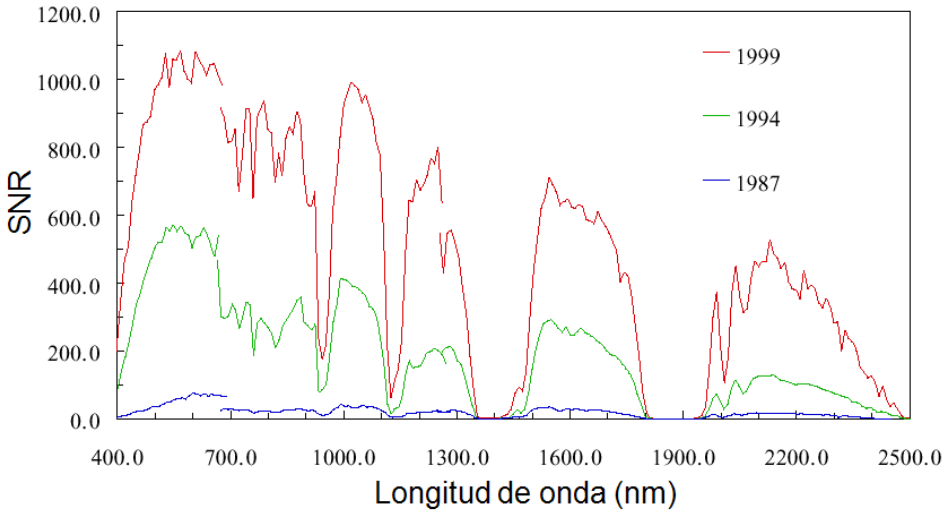


Figura 2.9: Evolución de la relación señal-ruido en el sensor AVIRIS de NASA.

- La cobertura de la parte visible del espectro es realizada por un espectrómetro EFOS-A, compuesto por un array de 32 detectores lineales.

- La cobertura en el infrarrojo es realizada por los espectrómetros EFOS-B, EFOS-C y EFOS-D, compuestos todos ellos por arrays de 64 detectores lineales.

- La señal medida por cada detector se amplifica y se codifica utilizando 12 bits. Esta señal se almacena en una memoria intermedia donde es sometida a una etapa de preprocesado.

- El sensor dispone de un sistema de calibración a bordo, que utiliza una lámpara halógena de cuarzo que proporciona la radiación de referencia necesaria para comprobar el estado de los diferentes espectrómetros.

- El sensor ha ido mejorando sus prestaciones en cuanto a la relación señal-ruido o signal-to-noise ratio (SNR), como se muestra en la Figura 2.9, que describe la evolución de la relación SNR del sensor a lo largo de los últimos años.

2.5.2. Sensor EO-1 Hyperion

El sensor Hyperion es uno de los tres principales instrumentos del satélite Earth Observing 1 (EO-1) que fue lanzado satisfactoriamente el 21 de noviembre de 2000 dentro del programa New Millennium de la NASA. El sensor Hyperion proporciona imágenes hiperespectrales de alta resolución con 220 bandas espectrales (0.4 a 2.5 μm) con una resolución espacial de 30 metros. Las imágenes que proporciona este instrumento abarcan una franja de 7.5 km (cada franja o línea de datos, contiene 256 píxeles) y proporciona información detallada de asignación del espectro en todos los 220 canales con una alta precisión radiométrica. Los componentes principales del instrumento son los siguientes:

- Sistema de óptica basado en el diseño de la misión KOMPSAT COE[11]. El telescopio se compone de dos espectrómetros de imagen de rejilla separados para mejorar la relación señal-ruido.
- Un array de plano focal que proporciona detectores separados para la onda corta para infrarrojos y el visible/infrarrojo cercano basados en el hardware de repuesto del programa LEWIS HSI.
- Un criorefrigerador idéntico al fabricado para la misión LEWIS HSI para la refrigeración del plano focal de onda corta para infrarrojos.

2.6. Principales retos del tratamiento de imágenes hiperespectrales

Ya sabemos que el potencial de estas imágenes es la gran cantidad de información que contienen, lo que permite distinguir materiales y objetos de manera más detallada. Pero esta gran ventaja se convierte también en un inconveniente cuando no se dispone de suficiente capacidad computacional para tratar y almacenar estas cientos de bandas. Nos enfrentamos entonces a los problemas de la alta dimensionalidad de los datos. Esta alta dimensionalidad podemos apreciarla si nos hacemos una idea del tamaño total de una

[11] http://ilrs.gsfc.nasa.gov/satellite_missions/list_of_satellites/kom5_general.html

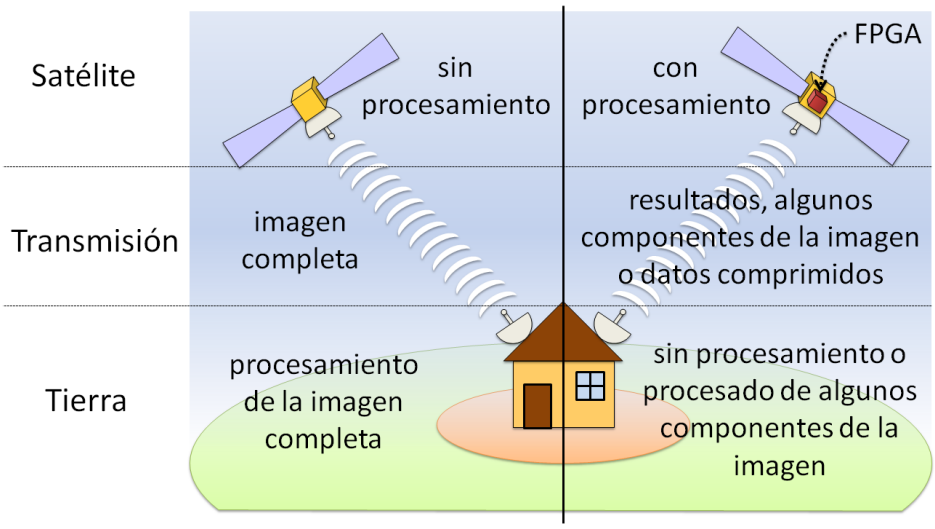


Figura 2.10: Ventajas potenciales del empleo de hardware reconfigurable en el procesado de datos de observación remota de la Tierra.

imagen de este tipo, multiplicando el tamaño del píxel en bits, por el tamaño de una imagen o banda individual, por el número de bandas totales. Otro de los retos a los que nos enfrentamos es la clasificación de los datos hiperespectrales debido a la mezcla espectral. Este fenómeno es muy habitual en el mundo real, independientemente de cuál sea la escala espacial considerada [Che99]. Sobre esta parte, es importante destacar que cualquier cadena de procesamiento de datos en cualquier ámbito científico tiene que ser flexible y adecuarse no sólo a su aplicación sobre distintos escenarios, sino también a los distintos tipos de resolución que proporcionan diversas variaciones espectrales y espaciales de los instrumentos. A continuación, se desarrollan estos retos en mayor detalle.

2.6.1. Alta dimensionalidad de los datos

Los sensores hiperespectrales modernos poseen una moderada resolución espacial y una alta resolución espectral que cubre varios cientos de bandas. Esta información no sólo debe analizarse, modificarse y almacenarse en los ordenadores de los usuarios de este tipo de datos, sino que está marcada por el hecho de que esa información debe ser enviada desde el satélite hasta

las estaciones en tierra. Se estima que los satélites del proyecto EOS (Earth Observing System) de la NASA, generarán más de un terabyte de datos al día. La economía de la transmisión y almacenamiento del gran volumen de datos resultante, recogidos por estos sensores, significa que la compresión de la imagen se convertirá en una característica esencial de los sistemas que incorporen sensores hiperespectrales. Por tanto, serán necesarias técnicas eficientes de compresión para la codificación de imágenes de observación remota de la Tierra, especialmente para aquellas que se transmiten directamente al suelo y se distribuyen a los usuarios.

Para tratar de solventar este problema, existen diferentes alternativas. Algunas de carácter radical implican la toma de información espectral bajo demanda. Es decir, el usuario establece qué bandas le aportan información y, en consecuencia, el satélite al pasar por la zona, únicamente tomará esas bandas. Con ello, se evita el cuello de botella de tener que bajar todas las bandas. Consecuencia de ello, cada país, más o menos, demandará las bandas que más interés le reporten. Por ejemplo, a España le interesarán las que guarden relación con la vegetación o desertificación, mientras que los países nórdicos quizás estén más interesados por el estudio de hielos o evolución de glaciares.

Esta es una primera alternativa, pero existen otras muchas. Una de ellas pasa por la compresión de la información a bordo del satélite (ver Figura 2.10). Es decir, en el lapso de tiempo en que la escena es recogida y es enviada a tierra, puede pasar un periodo de varias horas. En ese tiempo es posible comprimir la información de cara a que su envío consuma menos tiempo, o que en el mismo tiempo se envíe mayor cantidad de información. Ello, sin menosprecio de que en el propio envío se incluyan las compresiones y verificaciones inherentes del traspaso de información digital.

Las técnicas de compresión de imágenes se pueden clasificar en dos grandes categorías: compresión con o sin pérdida. La compresión sin pérdidas se basa en algoritmos que comprimen la información de cara a que su volumen de almacenamiento sea más pequeño, pero que a la hora de utilizarla, debemos descomprimirla, de forma que el resultado sea exactamente el mismo que el original antes de comprimir. Por otro lado, en la compresión con pérdida se descarta información pero a cambio se consiguen unas tasas de

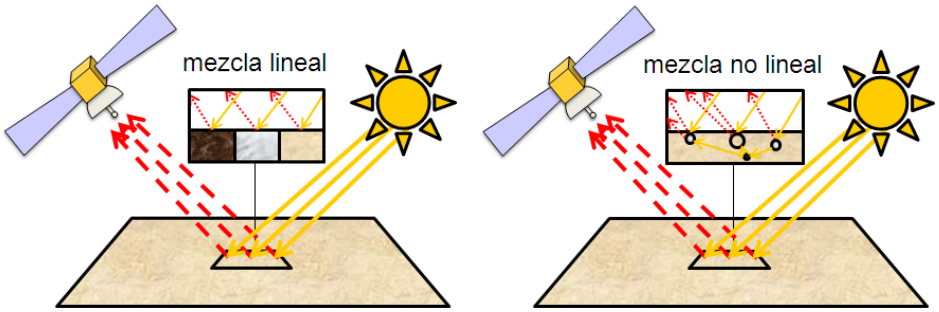
compresión más altas.

Llevada la compresión con pérdida a las imágenes hiperespectrales de observación remota de la Tierra, es claro que nos favorece tener un menor tamaño, pero debemos asegurarnos que en el proceso de compresión no se suprima cierta información importante que nos ayudaría a distinguir unos materiales de otros.

2.6.2. Mezcla espectral

El análisis de mezclas espectral (también conocido como *desmezclado espectral*) ha sido una atractiva meta explorada desde los primeros días de la imaginería hiperespectral [GVSR85] hasta nuestros días [PBB+09, SUP+09]. Sin importar la resolución espacial, los espectros captados en ambientes naturales son inevitablemente una mezcla de los distintos materiales que se encuentran dentro de la extensión espacial visualizada por el sensor hiperespectral [ASJ86]. Por ejemplo, es probable que el píxel captado sobre el área de vegetación en la Figura 2.3 en realidad comprenda una mezcla de vegetación y suelo. En este caso, el espectro medido se puede descomponer en una combinación de firmas espectrales puras de suelo y vegetación, ponderada por los coeficientes zonales que indican la proporción de cada firma *macroscópica* pura en el píxel mezcla [KM02]. La disponibilidad de sensores de imágenes hiperespectrales con un número de bandas espectrales que supera el número de componentes de la mezcla espectral [GES+98], ha permitido formular el problema de desmezclado espectral en términos de un sistema sobredeterminado de ecuaciones en el que, dado un conjunto de firmas espectrales puras (denominadas *endmembers*) permite la determinación de las *fracciones de abundancia* aparentes de cada endmember a través de un proceso de inversión numérica [BBY07].

Una técnica estándar para el análisis espectral de mezclas es el desmezclado espectral *lineal* [HC00, PMPP04], que supone que los espectros captados en el espectrómetro se pueden expresar en forma de una combinación lineal de endmembers ponderados por su abundancia correspondiente. Cabe señalar que el modelo lineal de mezcla asume una cantidad mínima de reflejos secundarios y/o múltiples efectos de dispersión en el procedimiento



(a) Dispersión simple (b) Dispersión múltiple

Figura 2.11: Modelo lineal frente modelo no lineal: dispersión simple frente dispersión múltiple.

de recopilación de datos, y por lo tanto el espectro medido se puede expresar como una combinación lineal de las firmas espectrales de los materiales presentes en el píxel mezcla (ver Figura 2.11(a)). Aunque el modelo lineal tiene ventajas prácticas como la facilidad de implementación y flexibilidad en diferentes aplicaciones [Cha03], el desmezclado espectral *no lineal* puede caracterizar mejor el espectro mezcla resultante para determinadas distribuciones de endmembers, tales como aquellas en las que los componentes de endmembers se distribuyen al azar a lo largo del campo de visión del sensor [GAC01, PPPM09]. En esos casos, los espectros mezcla recogidos por el instrumento de imagen se describen mejor bajo el supuesto de que parte de la radiación de origen se dispersa múltiplemente antes de ser capturados en el sensor (ver Figura 2.11(b)).

En este trabajo, se utiliza el modelo lineal dado que es el más ampliamente utilizado en la literatura relacionada con el análisis de mezclas mediante endmembers. A pesar de que el modelo no lineal puede resultar más preciso en determinadas circunstancias, su correcta aplicación requiere información a priori acerca de la geometría y las propiedades físicas de los objetos observados, lo cual lo hace difícilmente abordable en situaciones en las que no existe dicha información. Además, el modelo no lineal es difícilmente generalizable a diferentes áreas de aplicación (por ejemplo, las propiedades físicas y la geometría de los minerales son completamente diferentes de las propiedades físicas y la geometría de las cubiertas vegetales, e incluso cada cubierta

vegetal podría requerir un modelo específico).

2.6.2.1. Desmezclado espectral lineal: formulación del problema

Supongamos que una escena hiperespectral de observación remota de la Tierra con n bandas se denota por $\mathbf{I}$, en la que un píxel de coordenadas espaciales discretas (i,j) de la escena es representado por un vector $\mathbf{X}(i,j) = [x_1(i,j),\ x_2(i,j),\ ...,\ x_n(i,j)] \in \Re^n$, donde $\Re$ denota el conjunto de los números reales en los que se incluye la respuesta espectral del píxel $x_k(i,j)$ en los canales del sensor $k = 1, ..., n$. Bajo el supuesto de un modelo lineal de mezcla, cada vector de píxel en la escena original puede ser modelado utilizando la siguiente expresión:

$$\mathbf{X}(i,j) = \sum_{z=1}^{p} \Phi_z(i,j) \cdot \mathbf{E}_z + \mathbf{n}(i,j), \tag{2.1}$$

donde $\mathbf{E}_z$ denota la respuesta espectral del endmember z, $\Phi_z(i,j)$ es un valor escalar que designa la fracción de abundancia del endmember z en el píxel $\mathbf{X}(i,j)$, p es el número total de endmembers y $\mathbf{n}(i,j)$ es un vector de ruido. Generalmente al modelo descrito en 2.1 se le imponen dos limitaciones físicas, la restricción de abundancia no negativa, es decir, $\Phi_z(i,j) \geq 0$, y la restricción de suma de abundancias unitaria, es decir, $\sum_{z=1}^{p} \Phi_z(i,j) = 1$ [HC00]. La solución del problema de mezcla espectral lineal totalmente restringido descrito en 2.1 se basa en dos requisitos principales:

1. Una estimación correcta del número de endmembers, p, que se encuentran en la escena hiperespectral de entrada $\mathbf{I}$, y

2. La correcta determinación de un conjunto $\mathbf{E} = \{\mathbf{E}_z\}_{z=1}^{p}$ de endmembers y para cada píxel $\mathbf{X}(i,j)$, sus correspondientes fracciones de abundancias $\Phi(i,j) = \{\Phi_z(i,j)\}_{z=1}^{p}$.

Con el fin de solventar el primer requisito, una técnica exitosa en la literatura ha sido la dimensionalidad virtual (DV) [DC04]. El concepto DV formula la cuestión de si una firma distinta está presente o no en cada una de las bandas espectrales como un problema de prueba de hipótesis binaria,

donde se genera un detector de Neyman-Pearson llamado a servir de toma de decisiones basada en una preestablecida probabilidad de falsa alarma, P_F. A la luz de esta interpretación, la cuestión de determinar un valor apropiado para p es simplificado y reducido a la fijación de un valor específico de P_F. Como se muestra en los experimentos, una elección empírica adecuada es $P_F = 10^{-3}$ o $P_F = 10^{-4}$, donde el método utilizado en este trabajo para estimar la DV es el desarrollado por Harsanyi, Farrand y Chang [DC04] (en lo sucesivo como método de HFC), modificado posteriormente mediante la inclusión de un proceso de blanqueamiento de ruido previo para eliminar la correlación estadística de segundo orden. El propósito es que las fuentes de señal puedan ser decorreladas del ruido para lograr una mejor detección de la señal. El método resultante se conoce como ruido blanqueado HFC.

El segundo requisito para la aplicación exitosa del modelo lineal de mezcla (disponibilidad de extracción endmember y técnicas de estimación de la abundancia) se abordará en los capítulos siguientes. En los Capítulos 4 y 5 se muestra la implementación de dos de los algoritmos más populares utilizados en la extracción de endmembers y en el Capítulo 6 se presenta la implementación de un algoritmo para la estimación de abundancias. Por último, en el Capítulo 7 se realiza una comparativa entre los anteriores algoritmos de extracción de endmembers, para posteriormente implementar la cadena completa de desmezclado espectral.

2.6.3. Cadena estándar de desmezclado espectral

Dado un conjunto de vectores espectrales mezcla, el análisis de mezclas espectrales (o desmezclado espectral) tiene como objetivo estimar el número de materiales de referencia (también llamados endmembers), sus firmas espectrales y sus fracciones de abundancia. La Figura 2.12 muestra la sucesión de etapas de procesamiento involucradas generalmente en la cadena completa de desmezclado espectral: corrección atmosférica, reducción dimensional, extracción de endmembers y estimación de abundancias. En ocasiones, la extracción de endmembers y el proceso de estimación de abundancias se realizan de forma simultánea. A continuación, se ofrece una breve descripción de cada uno de los pasos:

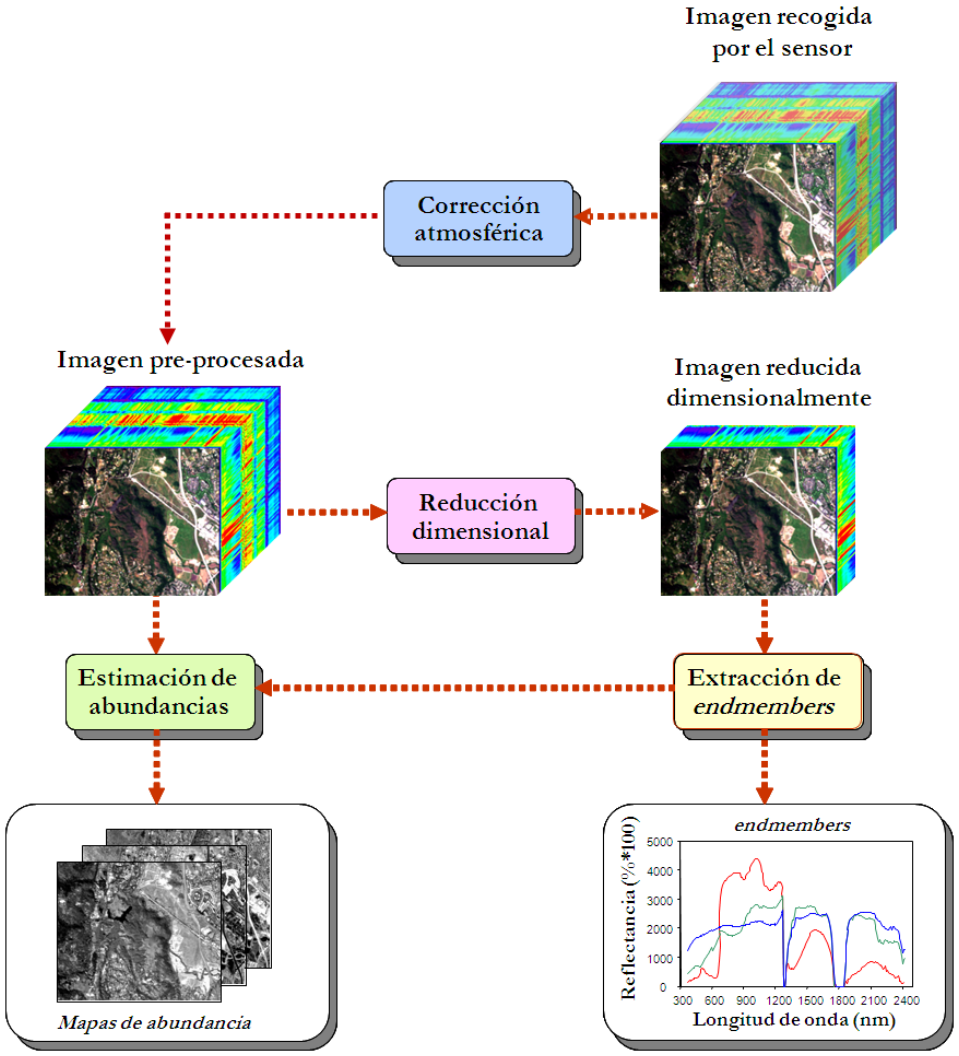


Figura 2.12: Cadena de procesamiento estándar para el análisis de datos hiperespectrales.

1. **Corrección atmosférica**. La atmósfera atenúa y dispersa la luz. Su presencia afecta, por lo tanto, a la radiancia recogida en el sensor. La corrección atmosférica compensa estos efectos mediante la conversión de radiancia a reflectancia, que es una propiedad intrínseca de los materiales. Destacamos, sin embargo, que el desmezclado lineal se podría realizar directamente con los datos en unidades de radiancia siempre y cuanto la atmósfera afectase por igual a todos los píxeles de la escena.

2. **Reducción dimensional**. Frecuentemente los píxeles de una imagen hiperespectral se ubican en un subespacio muy pequeño en comparación con el número de bandas disponibles. La identificación de este subespacio permite una correcta reducción de la dimensionalidad, que se traduce en una mejora en el rendimiento, en la complejidad de los algoritmos y en el almacenamiento de datos. Por otra parte, en el caso de la mezcla lineal, la dimensión del subespacio es igual al número de endmembers, una cuestión crucial en el desmezclado espectral. Sin embargo, dependiendo del algoritmo de análisis hiperespectral que se utilice, este paso no es siempre necesario.

3. **Extracción de endmembers**. El paso de extracción de endmembers consiste en identificar las firmas espectrales puras en la escena. Básicamente, existen dos aproximaciones: la geométrica, que explota el hecho de que los píxeles mezcla están contenidos en un simplex o un cono positivo, y la estadística, que modelan las fracciones de abundancia como variables aleatorias y formulan la determinación de los endmembers como un problema de inferencia estadística. Mientras tanto, está surgiendo una nueva aproximación: la regresión simple. En esta aproximación, el desmezclado espectral se formula como un problema de regresión lineal simple, de manera similar a la de la compresión de detección.

4. **Estimación de abundancias**. Teniendo en cuenta los píxeles espectrales observados y los endmembers identificados, la etapa de estimación de abundancias consiste en la solución de un problema de minimización de distancia con restricciones (las fracciones de abundancia estimadas se someten a menudo a las restricciones de no negatividad y de suma unitaria) que minimiza la diferencia entre los píxeles espectrales observados y los píxeles reconstruidos mediante combinaciones lineales de las firmas espectrales extraídas. Sin embargo, existen enfoques de desmezclado hiperespectral en los que los procesos de extracción de endmembers y de estimación de abundancias se realizan al mismo tiempo.

Capítulo 3

Hardware reconfigurable

No te establezcas en una forma, adáptala y construye la tuya propia, y déjala crecer, sé como el agua. Vacía tu mente, se amorfo, moldeable, como el agua. Si pones agua en una taza se convierte en la taza. Si pones agua en una botella se convierte en la botella. Si la pones en una tetera se convierte en la tetera. El agua puede fluir o puede chocar. Sé agua amigo mío.

Bruce Lee

Los métodos tradicionales para realizar computación son dos. El primero de ellos consiste en el procesamiento hardware utilizando circuitos interconexionados de forma fija, bien mediante la integración en un circuito de aplicación específica (Application Specific Integrated Circuit –ASIC–) o bien conectando componentes individuales en una placa [CH02]. El segundo método, denominado procesamiento software, se basa en el uso de microprocesadores que ejecutan un conjunto de instrucciones para realizar la computación.

El primer sistema se caracteriza por su rapidez y eficiencia para la aplicación concreta para la que ha sido diseñado, pero el circuito no puede ser alterado después de la fabricación, lo que le resta flexibilidad. Usando un microprocesador se incrementa la flexibilidad del sistema para poder cam-

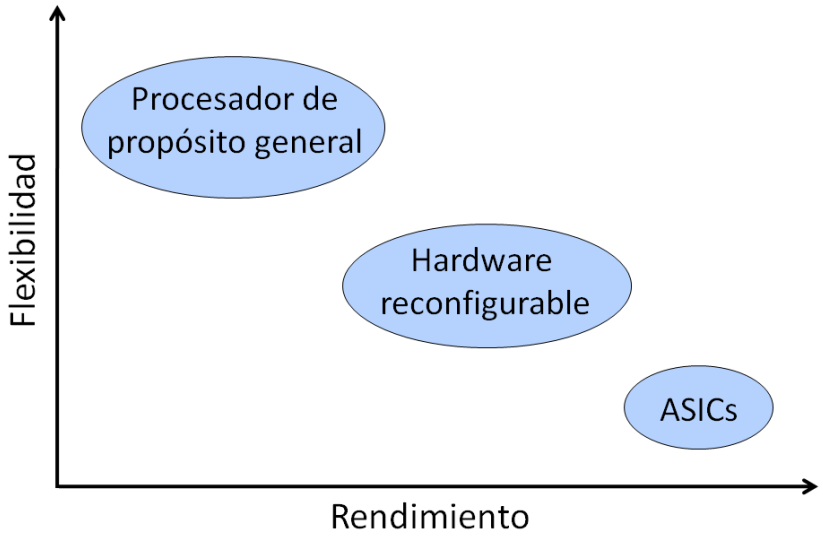


Figura 3.1: Relación flexibilidad–rendimiento de los distintos dispositivos de computación.

biar la funcionalidad empleando otro software, pero se reduce la eficiencia debido a las secuencias necesarias de lectura, decodificación y ejecución de instrucciones.

Los dispositivos reconfigurables vienen a cubrir el espacio existente entre estos dos métodos, de forma que se disponga de la eficiencia del procesamiento hardware y de un alto grado de flexibilidad [Tul97]. La adaptabilidad de las arquitecturas reconfigurables permite explotar el paralelismo existente en muchas aplicaciones de forma que se realice computación específica. En la Figura 3.1 se muestra cómo los sistemas configurables, basados en dispositivos reconfigurables, se sitúan en una zona intermedia en la relación flexibilidad–rendimiento. No son tan flexibles como un procesador de propósito general ni tan específicos, ni tan eficaces, como un ASIC. Sin embargo se benefician de las características positivas de ambos.

Las diferencias más importantes entre la lógica reconfigurable y el procesamiento convencional se pueden resumir en los siguientes aspectos según K. Bondalapati y V. K. Prasanna [BP02]:

- **Computación espacial**: El procesamiento de los datos se realiza dis-

tribuyendo las computaciones de forma espacial, en contraste con el procesamiento secuencial.

- **Ruta de datos (*datapath*) configurable**: Empleando un mecanismo de configuración es posible cambiar la funcionalidad de las unidades de computación y de la red de interconexión.

- **Control distribuido**: Las unidades de computación procesan datos de forma local en vez de estar gobernados por una única instrucción.

- **Recursos distribuidos**: Los elementos requeridos para la computación se encuentran distribuidos por todo el dispositivo, en contraste con una única localización.

Otras características importantes de estos sistemas destacadas por A. DeHon [DeH96] y R. Tessier [TB01] son la capacidad de adaptación, la posibilidad de configuración en tiempo de ejecución y la especialización. Beneficiándose de estas características específicas se han desarrollado sistemas reconfigurables eficientes para aplicaciones como la programación genética [SMP99a], detección de patrones de texto [SMP99b, SP01], criptografía [HW97, LWL00, LAB+04], compresión de datos [HSM00] o procesamiento de imágenes [BDH+97, GCL02] entre otras.

En este capítulo se realizará un repaso de varias arquitecturas reconfigurables que han dado soporte al desarrollo de la disciplina de la Computación Reconfigurable. A lo largo del mismo se presenta la evolución sufrida desde los primeros dispositivos Field Programmable Gate Arrays (FPGAs) hasta las plataformas híbridas que además integran, entre otros elementos, microprocesadores de propósito general y ASICs en el mismo chip. De esta forma la distribución de la computación puede ser repartida entre los distintos componentes del sistema. También se ha incluido una sección donde se presentan los distintos modos de configuración de los dispositivos reconfigurables, siendo estas distintas modalidades uno de los factores más característicos de los mismos. Finalmente, se expone el papel del hardware reconfigurable en sistemas empotrados y en misiones de observación remota de la Tierra.

3.1. Tecnología Reconfigurable

Las FPGAs consisten en una matriz de bloques lógicos (Logic Blocks –LBs–) y una red de interconexión. La funcionalidad de los LBs y las conexiones de la red de interconexión pueden modificarse mediante la descarga de los bits de configuración en el hardware [BP02]. La configuración del dispositivo se realiza empleando dispositivos anti–fuse [AGG+90] o bits de memoria SRAM que controlan la configuración de los transistores [HCJ+90]. El primer modo de configuración tiene menor capacidad de reprogramación mientras que la configuración mediante elementos de memoria SRAM es más versátil admitiendo incluso reconfiguración dinámica y/o parcial.

La Figura 3.2 muestra la estructura interna simplificada de una FPGA. A modo ilustrativo se ha elegido para la representación una distribución de tipo isla empleada en varias familias de Xilinx. Existen otras arquitecturas de interconexión como la basada en filas [Act97], sea–of–gates [Act01], jerárquica o estructuras en una única dimensión como las empleadas en Garp [HW97], Chimaera [HFHK04] o NAPA [RLG+98].

Los LBs interconectados mediante esa red contienen típicamente uno o varios circuitos combinacionales programables Look-Up Table (LUT), uno o varios biestables, lógica adicional y las celdas de memoria SRAM requeridas para la configuración de todos los elementos. Las tareas de entrada/salida se realizan en la periferia del dispositivo, bien mediante LBs o disponiendo de bloques específicos denominados Input–Output–Blocks (IOBs). Actualmente se integran habitualmente otros elementos como son los bloques de memoria RAM dedicada [Xil00], multiplicadores e incluso microprocesadores [Xil09].

Una de las clasificaciones más habituales del hardware reconfigurable se realiza atendiendo a su granularidad. La granularidad de la lógica reconfigurable se define como el tamaño de la menor unidad funcional que es tratada por las herramientas de emplazamiento y rutado [BP02]. Las FPGAs (de grano fino) disponen de elementos funcionales de pequeño tamaño, lo que las dota de una mayor flexibilidad. Sin embargo, sufren retardos elevados cuando se componen circuitos complejos. De forma típica son unidades funcionales de 2 a 6 entradas y una única salida, lo que permite definir una función lógica diferente para cada bit del sistema. El hardware reconfigurable con unidades

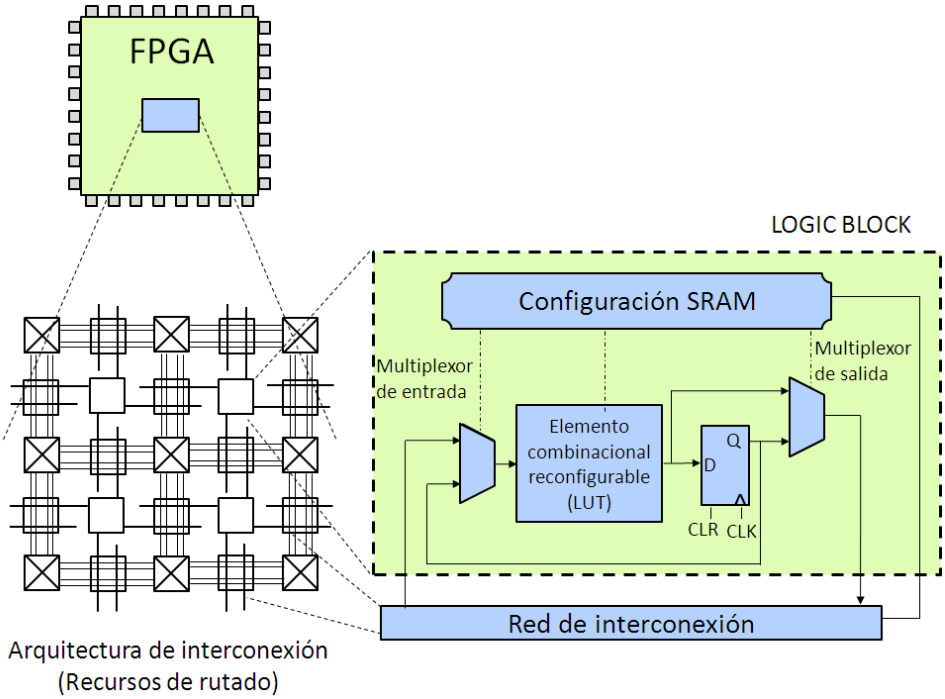


Figura 3.2: Modelo genérico de una FPGA.

funcionales grandes se denomina de grano grueso, existiendo arquitecturas como la Chameleon [Sys00] con elementos aritméticos de 32 bits o la Morphosys compuesta por un componente reconfigurable, un procesador RISC y una interfaz de memoria con un gran ancho de banda.

Con la integración de múltiples elementos arquitecturales (entendidos como procesadores, memoria e interfaces para periféricos) en FPGAs que disponen de una sección de lógica programable por el usuario surge el concepto de Arquitecturas Híbridas [BP02].

El avance tecnológico que ha permitido la integración de sistemas en un único dispositivo ha ido acompañado de diversa terminología, todavía no normalizada, para designar a estos sistemas. A continuación se detallan algunos de estos términos:

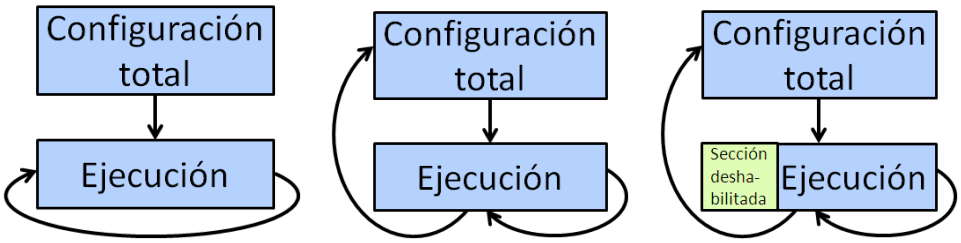
- **System–on–Chip (SoC)**: Circuito integrado formado por diversos módulos VLSI con distinta funcionalidad que interconectados entre sí ofrecen toda o casi toda la funcionalidad específica para una aplicación.

- **System–on–Programable–Chip (SoPC)**: Se aplica este término específicamente cuando el dispositivo utilizado para realizar el sistema en un chip es reconfigurable. En los SoPC no se utiliza la capacidad de reconfiguración dinámica que puedan disponer estos integrados, sino únicamente las facilidades que ofrecen estos dispositivos en la fase de desarrollo y posteriores actualizaciones del sistema.
- **Configurable-System-On-Programable-Chip (CSoPC)**: Con este término se definen los sistemas SoPC en los que se hace uso de la capacidad de reconfiguración de los mismos para aplicaciones de Computación Reconfigurable. Pueden incluirse bajo la denominación CSoPC tanto los sistemas que admiten diferente configuración estática según ciertas condicionantes, como los que utilizan la reconfiguración parcial dinámica para modificar en tiempo de ejecución una sección hardware.
- **Multiprocessor–Configurable–System–On–Programable–Chip (MCSoPC)**: Se aplica esta definición a los sistemas CSoPC que incluyen varios procesadores que ejecutan software, funcionando de forma simultánea.

Dentro de las distintas arquitecturas dinámicamente reconfigurables actuales, las FPGAs dominan ampliamente el mercado. La razón principal es que se basan en una tecnología madura con muchos años de desarrollo y sustentada por un amplio conjunto de herramientas de diseño.

3.2. Tipos de configuración

De forma general un dispositivo reconfigurable se configura cargando en el mismo una secuencia de bits denominada bitstream. El modo de carga varía según la interfaz que éste disponga. Las interfaces de configuración son de tipo serie o paralelo. El tiempo de configuración es directamente proporcional al tamaño del bitstream. Las FPGAs tienen, en general, tiempos de configuración mayores que los dispositivos de grano grueso debido al mayor tamaño de sus bitstreams. Esto es debido a que tienen muchos elementos para ser configurados.



(a) Modelo estático (b) Modelo dinámico (c) Modelo parcial y dinámico

Figura 3.3: Modelos de reconfiguración.

Tradicionalmente, las FPGAs se han utilizado para realizar una determinada función en un único contexto, realizándose una configuración de todo el dispositivo. En el caso de que se desee reconfigurar en tiempo de ejecución, la reconfiguración de todo el dispositivo es un proceso lento y limitado. La lógica que se va a reconfigurar debe parar la computación y continuar tras la nueva configuración. La penalización impuesta por el tiempo de reconfiguración es importante [LCH00, RCGM07, RCG+08], haciendo en muchas aplicaciones inviable la reconfiguración. A continuación se presentan brevemente los modelos de reconfiguración más representativos:

- **Reconfiguración estática**: Implica parar el sistema y reiniciarlo con una nueva configuración. Su utilidad se centra en los procesos de diseño (depuración) y en la actualización de sistemas. Cada aplicación dispone de una configuración que se carga una vez tras el encendido. La mayoría de los sistemas realizados en la actualidad con lógica reconfigurable disponen de este tipo de reconfiguración, también denominada reconfiguración en tiempo de compilación (en el proceso de diseño). La Figura 3.3(a) sintetiza este modo de operación en el que tras la configuración comienza la ejecución de la lógica configurada sin posibilidad de una nueva carga.

- **Reconfiguración dinámica**: Con el objetivo de obtener un equilibrio entre velocidad de ejecución y área de silicio surge el modo de configuración dinámico mediante el cual se modifica la lógica configurable

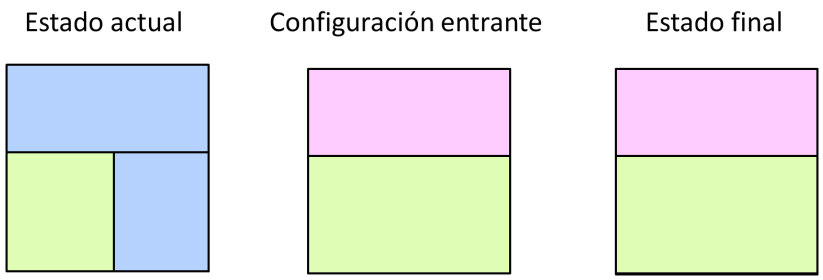
(incluyendo el rutado) en tiempo de ejecución. La reconfiguración dinámica se basa en el concepto de Hardware Virtual [And99] de forma similar a la memoria virtual. Utilizando la capacidad de reprogramación del dispositivo se cambian las configuraciones según se requieren distintas computaciones, reduciendo de esta manera el área de circuito necesaria.

La Figura 3.3(b) representa el modelo de reconfiguración dinámico. De la computación realizada por la lógica configurada (ejecución) se obtiene información que sirve para determinar la nueva configuración. En el caso de que el sistema lo permita, se podría realizar la nueva configuración mientras se mantiene la ejecución. En la práctica, la solución más habitual es la de mantener deshabilitada la sección que se va a reconfigurar mientras continúa la ejecución en la otra sección del dispositivo. Este modo de reconfiguración parcial dinámico se representa en la Figura 3.3(c).

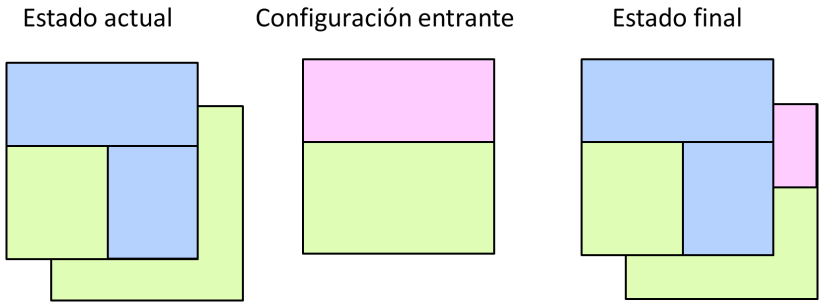
La reconfiguración dinámica tiene diversas variantes según el modo en el que ésta se aplique. Las tres más representativas son:

- **Reconfiguración con contexto único**: Corresponde con el modo de configuración de las FPGAs que únicamente soportan un acceso secuencial a la memoria de configuración. En el caso de realizar una reconfiguración dinámica con estos dispositivos se sufren unas penalizaciones temporales importantes debido a que cada intercambio de funcionalidad requiere una reprogramación completa de los mismos. El modelo de este modo se ha representado en la Figura 3.4(a). La configuración entrante sustituye completamente a la que estaba aplicada sobre la lógica configurable.

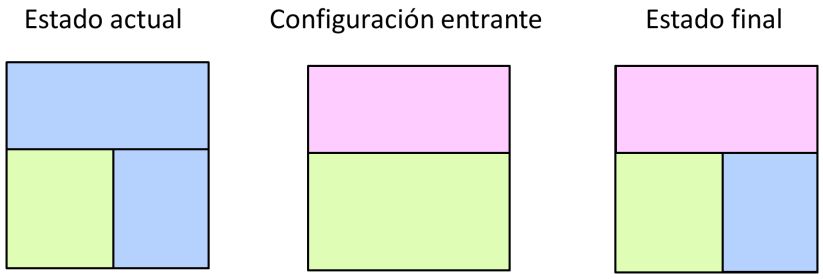
- **Reconfiguración multi-contexto**: Los dispositivos que soportan este tipo de reconfiguración tienen varios bits de memoria de configuración para cada bit de los elementos configurables [TCJW97, DeH96]. En la Figura 3.4(b) se representa un modelo de estos dispositivos donde los bits de memoria pueden considerarse como múltiples planos de información de configuración. Cada plano debe configurarse totalmente, de igual forma que los de contexto único. Sin embargo, el cambio



(a) Modelo contexto único



(b) Modelo multicontexto



(c) Modelo parcialmente reconfigurable

Figura 3.4: Modelos de configuración dinámica.

entre contextos se realiza de forma muy rápida, admitiéndose además la carga de una nueva configuración en un plano no activo mientras otro lo está. Destacar que las FPGAs con este tipo de reconfiguración son teóricas, dado que se requiriría relicar un gran número de bits de configuración, sin embargo, existen dispositivos de grano grueso que utilizan este esquema de reconfiguración.

- **Reconfiguración parcial**: Uno de los avances tecnológicos más im-

portantes en el área de la reconfiguración consiste en la capacidad de algunos dispositivos para admitir la modificación de parte de la configuración mientras el resto del hardware sigue realizando la computación de forma ininterrumpida.

La Figura 3.4(c) muestra este modelo de reconfiguración. En este caso el plano de configuración funciona como una memoria RAM. De este modo se pueden emplear las direcciones para especificar una determinada localización que se desea reconfigurar. La reconfiguración parcial dinámica también permite que se carguen configuraciones diferentes en áreas no usadas del dispositivo con el fin de reducir la latencia en el cambio de contexto, tal y como se propone en el trabajo [CRGM10].

Existen varias plataformas reconfigurables que soportan la reconfiguración parcial como Chimaera [HFHK04], PipeRench [CPSS00], NAPA [RLG+98], Xilinx Virtex [Xil03] y Altera Stratix V [Alt10]. Una variante de la reconfiguración parcial realizada con el fin de reducir la penalización por el tiempo necesario para la carga de los bitstreams parciales es la reconfiguración pipeline. Este sistema está orientado a computaciones de estilo datapath [Sch97a], donde se emplean más etapas pipeline que las que caben simultáneamente en el hardware. El caso mejor sería la situación en la que la ejecución de cada etapa comenzase tras el instante de ser reconfigurada. En este caso la reconfiguración de cada etapa se realizaría justo antes de comenzar a recibir el flujo de datos [CLC+02].

Para aliviar la problemática de la penalización en tiempo impuesta por el proceso de reconfiguración, se han investigado diversas tácticas. La precarga de configuraciones [Hau98, RCGM07] es una de ellas. El objetivo en este caso es cargar la configuración en el dispositivo con anticipación a que ésta se requiera. El carácter especulativo de esta técnica fija la complejidad de la misma en determinar con suficiente antelación cuál va a ser la siguiente configuración requerida. También la compresión de la configuración se ha estudiado como una de las alternativas para la reducción del tiempo de configuración [HLS98].

Otra alternativa abordada con el mismo fin es la del uso de caché de

configuraciones en el dispositivo [CLC+02, PRMC07]. Al retener configuraciones en el integrado, se reduce la cantidad de información de configuración transferida. Al igual que en las cachés de los procesadores de propósito general, se aplican los conceptos de localidad temporal y espacial con el fin de decidir qué configuraciones se mantienen y cuáles se eliminan cuando se produce la reconfiguración. Decisiones incorrectas pueden producir el efecto contrario al deseado, aumentando la penalización temporal producida por la reconfiguración.

Finalmente en [CRGM10], se demuestra como una gestión eficiente de la reconfiguración parcial puede ocultar la mayor parte de la latencia de este proceso, incluso cuando se trabaja con aplicaciones muy dinámicas que necesitan reconfiguraciones frecuentes.

3.3. Diseño con FPGAs mediante lenguajes de descripción hardware

El proceso automático de diseño con diferentes tecnologías hardware es conocido como síntesis de alto nivel y dentro de este contexto, se enmarca el diseño con FPGAs. Un algoritmo o tarea debe ser descrito en un lenguaje de alto nivel que esté directamente relacionado con elementos hardware para su fácil traducción a un circuito. En los últimos 20 años se ha popularizado y extendido el uso de los llamados lenguajes de descripción hardware (HDL, Hardware Description Language), entre los cuales destacan Verilog [Ver08] y VHDL [VHD06].

El primer paso para sintetizar una tarea a hardware es describir el algoritmo en alguno de los HDLs disponibles. El siguiente paso es determinar qué tipo de bloques hardware son necesarios y cómo están conectados entre sí y después, el tercer paso, consiste en asignar bloques básicos configurables concretos de la FPGA (CLBs) y rutado de señales entre ellos. En el cuarto paso del proceso se obtiene el mapa de bits de configuración necesario para que los LBs utilizados en el diseño del circuito realicen cada uno la funcionalidad necesaria. Por último, es necesario escribir dicho mapa de bits en la memoria de configuración del dispositivo. La Figura 3.5 refleja el proceso

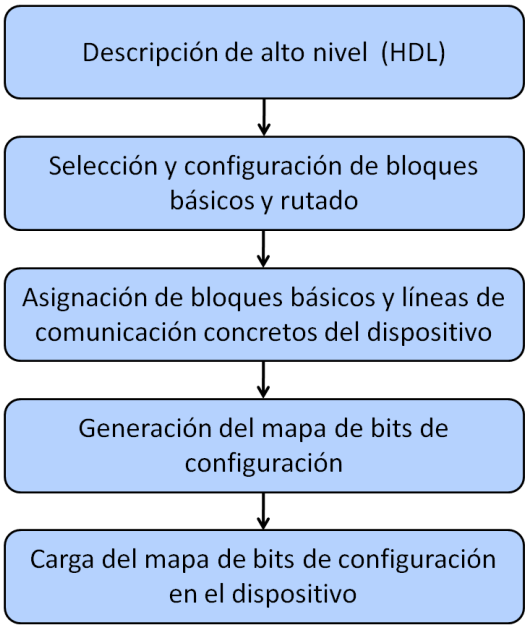


Figura 3.5: Etapas de diseño con FPGAs.

explicado.

El VHDL (Very High Speed Integrate Circuit Hardware Description Language), es un lenguaje de descripción y modelado diseñado para describir la funcionalidad y la organización de sistemas hardware digitales, placas de circuitos y componentes.

La finalidad del modelado es la simulación. La sintaxis amplia y flexible del lenguaje VHDL permite tanto el modelado estructural como el modelado funcional de circuitos. En el primer caso, se describe el circuito indicando los componentes y las conexiones que lo componen (lo cual requiere un conocimiento detallado del circuito). En el segundo caso, se describe el circuito indicando lo que hace y cómo funciona, es decir, describiendo su comportamiento (sin necesidad de conocer su estructura interna).

Esta segunda metodología de modelado resulta muy interesante desde el punto de vista del diseño de sistemas digitales. Más aún teniendo en cuenta que hoy en día otra de las aplicaciones del lenguaje VHDL, con una gran demanda de uso, es la síntesis automática de circuitos. En el proceso de síntesis, se parte de una especificación de entrada con un determinado nivel

de abstracción, y se desciende verticalmente por los niveles de la jerarquía de diseño hasta llegar a una implementación más detallada, menos abstracta. Puesto que el VHDL fue inicialmente concebido para el modelado de sistemas digitales, su utilización en síntesis no es inmediata. Sin embargo, la sofisticación de las actuales herramientas de síntesis es tal que permiten implementar diseños especificados en un alto nivel de abstracción.

Así pues, VHDL permite diseñar, modelar y comprobar un sistema desde un alto nivel de abstracción hasta el nivel de definición estructural de puertas lógicas. Al estar basado en un estándar (IEEE Std. 1076-1987) reduce los errores de comunicación y los problemas de compatibilidad. Finalmente, dada su modularidad, permite dividir o descomponer un diseño hardware y su descripción VHDL en unidades más pequeñas.

Los componentes de un proyecto VHDL son los siguientes:

- **Entity**: Es el más básico de los bloques de construcción en un diseño. Una entidad VHDL especifica el nombre de la entidad, sus puertos, e información relacionada con ella. Todos los diseños son creados usando una o varias entidades. La entidad describe la interfaz en el modelo VHDL.
- **Architecture**: La arquitectura describe la funcionalidad esencial de la entidad y contiene los estados que modelan el funcionamiento de ésta. Una entidad puede tener varias arquitecturas.
- **Configuration**: Permite unir la instancia de un componente a la pareja entidad–arquitectura. Describe el comportamiento a utilizar para cada entidad.
- **Package**: Es una colección de los tipos de datos y subprogramas usados comúnmente en un diseño. Las librerías forman parte de los packages.

3.4. Ventajas e inconvenientes de las FPGAs

A pesar de que no carecen de inconvenientes, las características de estos dispositivos los han convertido en la actualidad en una solución equilibrada para las demandas actuales de los sistemas de computación.

3.4.1. Ventajas

A continuación pasamos a enumerar la lista de los beneficios de utilizar FPGAs para computación de propósito general.

- **Aumento de la velocidad de procesamiento**: ya que con las FPGAs podemos obtener mejores tiempos de ejecución gracias a dos factores:

  1. **Procesamiento hardware**: la velocidad de ejecución de un circuito hardware específicamente diseñado para realizar una tarea es habitualmente mayor que la de un código software de alto nivel ejecutado en un procesador de propósito general. Esto se debe a que el circuito hardware sólo realiza las operaciones necesarias e incluye una ruta de datos a medida, mientras que un procesador convencional debe realizar los cálculos a partir de su repertorio de instrucciones, utilizando una ruta de datos que no esta optimizada para la tarea en cuestión.
  2. **Explotación de paralelismo**: la posibilidad de descomponer una aplicación en tareas o subtareas y de explotar el paralelismo es una realidad en las FPGAs, ya que diferentes tareas pueden ser ejecutadas simultáneamente en el dispositivo si se dispone de suficientes elementos básicos para configurar los circuitos simultáneamente. La multitarea hardware también es factible en los dispositivos actuales, gracias a las técnicas de diseño modular.

- **Reducción de consumo**: ya que se pueden aplicar técnicas que evitan el consumo de energía innecesario, combinadas con otros factores:

  1. **Avances en la tecnología**: el consumo de las FPGAs ha ido disminuyendo desde su aparición en el mercado, alcanzando unos niveles muy aceptables en el presente. Asimismo existen ya propuestas de novedosas tecnologías que podrían fabricarse en un futuro cercano y que según los resultados presentados por [BTAS07] consumen hasta 25 veces menos que un circuito reconfigurable fabricado con tecnología CMOS. Actualmente, Xilinx y Altera fabrican FPGAs utilizando una tecnología de 28 nm.

2. **Eficiencia de diseño**: debido a la gestión multitarea, solamente aquellas partes de una aplicación que se necesiten en cada momento estarán siendo ejecutadas en el dispositivo, y por tanto se reduce el consumo al mínimo necesario para la ejecución de una aplicación en cada momento. Además, al tratarse de diseños hechos a medida, sólo se realizan los cálculos necesarios (desaparece, por ejemplo, la necesidad de buscar y decodificar instrucciones), y con unidades dimensionadas para el tamaño del problema.

- **Flexibilidad**: la posibilidad de reconfigurar los elementos básicos, por partes o en su totalidad, dota a los diseños realizados sobre FPGAs de la flexibilidad necesaria para adaptarse a bajo coste a un mercado que cambia con rapidez.
- **Coste**: el aumento en la popularidad de las FPGAs y los avances en la tecnología de fabricación han permitido que los precios de estos dispositivos para grandes tiradas bajen también a una velocidad que empieza a ser competitiva con respecto a otros dispositivos tradicionales.

A todas estas ventajas podemos sumarle la aparición de herramientas y entornos de desarrollo cada vez más completos y documentados para trabajar con estos dispositivos.

3.4.2. Inconvenientes

Aunque los puntos a favor de utilizar FPGAs son muchos y están sólidamente probados por la literatura, existen sin embargo dos aspectos importantes a tener en cuenta a la hora de su utilización:

- **Rutado de señales**: El rutado obtenido a partir de una compilación automática no es el óptimo y suele ser necesaria una revisión por parte del diseñador y la realización de algunos retoques manuales. No obstante, los fabricantes de FPGAs conocen esta dificultad y están trabajando para ofrecer al mercado arquitecturas de FPGAs donde se mejoren las posibilidades de conexión entre celdas lógicas. Algunos autores han propuesto ya diseños en 3D [DWM+05], que ofrecen grandes posibilidades en este aspecto.

- **Tiempo de reconfiguración**: una de las críticas más usuales al uso de FPGAs dinámicamente reconfigurables para ejecución multitarea hardware es el elevado tiempo de reconfiguración del dispositivo o partes de él. Este tiempo está en la actualidad en el orden de los milisegundos y es proporcional al área de dispositivo reconfigurada (tamaño del mapa de bits de configuración). Puesto que representa el cuello de botella real de los sistemas multitarea hardware, encontramos pruebas de los numerosos esfuerzos dirigidos a mejorar este aspecto, que pasamos a comentar:

  - **Investigación de nuevas tecnologías**: tal es el caso de [WK07], que han implementado una matriz de puertas dinámicamente reconfigurable con tecnología óptica y tiempos de reconfiguración del orden de los nanosegundos.

  - **Propuestas de nuevas arquitecturas para FPGAs**: las propuestas de arquitecturas multicontexto, que permiten cargar una nueva configuración en el chip mientras todavía se está ejecutando una tarea en él, permitirían mejorar la velocidad de reconfiguración, directamente ligada a la lectura del mapa de bits de memoria y su escritura en la FPGA. En trabajos como [MM06] se ha investigado en esta línea.

  - **Desarrollo de técnicas de reconfiguración**: especialmente diseñadas para minimizar el tiempo de reconfiguración. En [LH00] se presentan técnicas de compresión del mapa de bits de configuración y en [LCH00] se ha propuesto utilizar caches para guardarlos y con ello acelerar el proceso de escritura del mapa de bits en la FPGA, así como en [RMC05, CRGM10], donde se han desarrollado técnicas de pre-búsqueda de mapas de configuraciones basadas en una correcta planificación de los módulos a reconfigurar.

3.5. Hardware reconfigurable en sistemas empotrados

El hardware reconfigurable proporciona un medio flexible para implementar circuitos. Este tipo de recursos hardware son configurables (y en general reconfigurables) después de la fabricación, lo que proporciona una base simple de diseño hardware para implementar una enorme variedad de circuitos [GRM+10, GOR09, OGR10].

Los sistemas empotrados tienen a menudo mayores exigencias de rendimiento y consumo de energía, lo que lleva a los diseñadores a incorporar hardware de propósito específico en sus diseños. Las implementaciones hardware dedicadas evitan la penalización de la ejecución de una instrucción (búsqueda/decodificación/ejecución) software tradicional, y utilizan los recursos espaciales para aumentar el paralelismo. En muchas aplicaciones empotradas, tales como multimedia, codificación, comunicación inalámbrica y otros, encontramos cálculos altamente paralelos que se adaptan bien a una implementación en hardware y representan una fracción significativa de la computación total que requiere el sistema [LSL+99, KGV04].

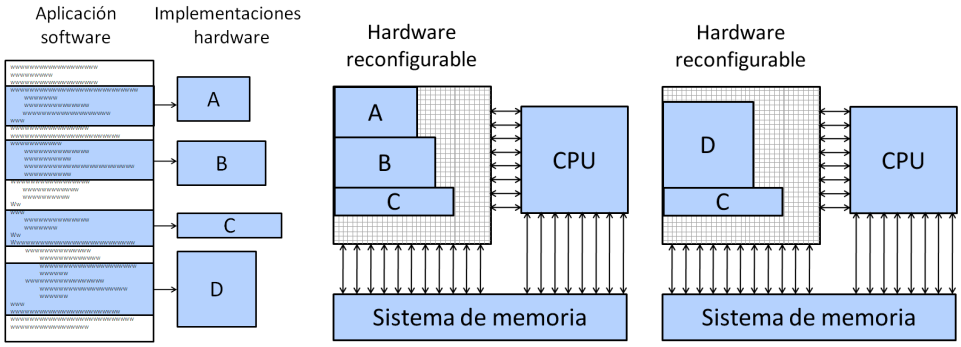
Lamentablemente, la implementación de circuitos integrados específicos para un aplicación (ASIC) no es viable o deseable para todos los circuitos. Un problema clave es que los costes de ingeniería de los ASICs se han incrementado drásticamente. Un conjunto de máscaras para un ASIC con una tecnología de 90 nm costaba aproximadamente un millón de dólares en 2003 [PKP+03]. Anteriormente, el uso de FPGAs como sustitutos de los ASIC sólo era rentable en aplicaciones de baja tirada. Las FPGAs tienen altos costes por unidad, que son esencialmente una amortización de los costes de la ingeniería de la FPGA sobre todos los usuarios de estos chips. Sin embargo, como el coste de ingeniería en ASIC aumenta y las FPGAs se venden en mayor volumen, el coste por unidad de los ASICs empieza a superar el coste por unidad de las FPGAs para aplicaciones de mayor demanda, desplazando el equilibrio hacia el lado de las FPGAs [Act05]. Sobre todo teniendo en cuenta la flexibilidad del hardware reconfigurable para dar cabida a nuevos circuitos para la corrección de errores, actualizaciones en el protocolo implementado o nuevos avances, la cara tecnología del diseño fijo en ASIC se vuelve menos

atractiva.

Además, los dispositivos tradicionalmente clasificados como sistemas empotrados, como las PDAs (Personal Digital Assistants) y teléfonos móviles, son cada vez más polivalentes. Estos sistemas pueden implementar un conjunto muy diverso de aplicaciones que requieren las ventajas de rendimiento y consumo de una implementación hardware, tales como las comunicaciones inalámbricas, criptografía y audio/vídeo digital. Incluir un acelerador hardware fijo a medida para cada tipo de aplicación posible es en general imposible, sobre todo si una o más de las aplicaciones se desconocen en tiempo de diseño. El hardware reconfigurable puede actuar como un acelerador hardware "general", implementando una gran variedad de computaciones dentro de las aplicaciones o implementándolas en su totalidad. Las secciones con cálculo intensivo se pueden ubicar en el hardware según sean necesarias, y posteriormente pueden reemplazarse para dejar sitio para otros cálculos. La Figura 3.6 ilustra un caso donde, después de completarse los cálculos **A** y **B**, pueden ser sustituidos por el cálculo de **D**, potencialmente mientras el cómputo de **C** aún se está ejecutando. En efecto, la reconfiguración en tiempo de ejecución permite al hardware reconfigurable actuar como un acelerador hardware virtual, con capacidades y habilidades más allá de su estructura física. Los principales fabricantes de FPGAs están desarrollando plataformas similares a la de la figura con uno o varios procesadores integrados en el mismo chip que la FPGA.

El funcionamiento a baja potencia es fundamental en muchos sistemas empotrados para alargar la vida de la batería, reducir los costes de operación e incluso mejorar la fiabilidad [Moy01]. Los cálculos realizados en hardware reconfigurable a menudo disipan menos energía que un software equivalente ejecutado en un procesador empotrado, ya que normalmente se pueden implementar con frecuencias de reloj más bajas y evitar la sobrecarga asociada a las instrucciones individuales búsqueda, decodificación, emisión y escritura [ASI+98, MMF98, TB01, LTC03, RSH05]. Sin embargo, tienen una mayor disipación de potencia que las soluciones ASIC fijas [TB01, KR06], principalmente debido al exceso de interconexiones que se incluyen en las FPGAs.

La flexibilidad del hardware reconfigurable también se puede utilizar para aumentar la tolerancia a fallos de los diseños. Este tipo de dispositivos



(a) Implementación hardware de secciones intensivas de cálculo.

(b) Ejecución de las tareas A, B y C.

(c) Ejecución de la tarea D tras la finalización de A y B

Figura 3.6: Codiseño hardware–software en FPGA.

puede ser reconfigurado para evitar fallos hardware [Lap05], como resultado de la fabricación o del medio ambiente. La lógica reconfigurable extra en un diseño se puede utilizar como sustituta en el caso de que se produzca un fallo en un recurso no reconfigurable [ST04]. La nueva configuración, puede ser incluso actualizada de forma remota [Lap05, Bra01] para evitar incomodar a los consumidores o permitir actualizar un dispositivo que no puede ser físicamente accesible (sistemas desplegados en el espacio, en el fondo del océano, o en otros lugares remotos o peligrosos). La tolerancia a fallos del hardware reconfigurable, incluso se puede extender a fallos de diseño, permitiendo corrección de errores o mejoras para nuevos estándares lo que aumentará la vida útil del dispositivo.

La reconfiguración hardware en tiempo de ejecución permite que se aceleren en hardware un mayor número de cálculos que de cualquier otra manera, pero introduce una penalización de reconfiguración ya que se deben cargar los nuevos valores en la SRAM de configuración en cada reconfiguración. Para chips de FPGA, este proceso puede tardar del orden de milisegundos [GVPR04], posiblemente eclipsando los beneficios de la computación hardware. Técnicas tales como la precarga [LH02] o la planificación [CGRM08] pueden reducir la penalización de reconfiguración mediante la predicción de las configuraciones necesarias y la carga por adelantado, así como la retención de configuraciones (en un dispositivo parcialmente re-

configurable) que puedan ser necesarias en el futuro próximo [CRGM10]. Si el funcionamiento del sistema está bien definido y es conocido de antemano, la partición temporal y la programación estática pueden ser suficientes [MKB+99, PB99]. Para otros sistemas, el método más sencillo es cargar configuraciones a medida que se necesitan, eliminando una o más configuraciones del hardware reconfigurable si es necesario para liberar recursos suficientes [Arn05, LCH00, Bre96, RDVC05].

En la siguiente sección, se expondrán en detalle las ventajas de utilizar hardware reconfigurable en misiones de observación remota de la Tierra.

3.6. Hardware reconfigurable en misiones de observación remota

El coste para bajas tiradas, el rendimento y la flexibilidad hacen que el hardware reconfigurable sea especialmente útil en aplicaciones espaciales, donde se han utilizado para varias misiones, incluidas las Mars Pathfinder y Mars Surveyor [RSBK01, KAT+03]. Estos dispositivos pueden ser reconfigurados para añadir funcionalidad, actualizando así los objetivos de la misión, o corregir errores de diseño sin necesidad de una misión espacial para su reparación. Las naves espaciales requieren dispositivos especiales resistentes a la radiación que no se fabrican en el mismo volumen (debido al mayor coste y menor demanda) que los microchips estándar, lo que lleva a los diseñadores a incorporar la funcionalidad de muchos componentes discretos diferentes en una o unas pocas FPGAs endurecidas para radiación.

Futuras misiones de observación remota de la Tierra, como las mostradas en la Tabla 2.1, requerirán importantes avances tecnológicos para lograr altas tasas de procesamiento de datos. Por ejemplo, recientes estudios internos del Jet Propulsion Laboratory de la NASA, estiman que se puede esperar un volumen de 1 a 5 terabytes de datos en bruto (sin comprimir) al día de los instrumentos de imagen futuros como HyspIRI. Como resultado, será necesaria la implementación eficiente de algoritmos de procesamiento a bordo con el fin de disminuir drásticamente el volumen de datos a transmitir entre los satélites y las estaciones terrestres existentes. La tendencia en el diseño de módulos

hardware para las misiones de observación remota de la Tierra ha sido el uso de dispositivos hardware con reducidas dimensiones y coste, pero flexibles y con gran potencia de cálculo [ND99, FBC+03]. El procesamiento a bordo permite una buena reutilización de los costosos recursos hardware. Además, permite la toma de decisiones autónomas a bordo que pueden reducir potencialmente el retardo entre la captura de imágenes, el análisis y la acción correspondiente. La implementación de algoritmos de procesamiento a bordo para llevar a cabo la reducción de datos puede disminuir drásticamente las tasas de transmisión de datos. Sin embargo, muchos sistemas disponibles se caracterizan por su alto consumo de energía, coste, y el requisito de tarjetas de interfaz adicionales. Una alternativa interesante es incluir las recientemente desarrolladas FPGAs híbridas, como la Xilinx Virtex-6 y Virtex-7. Estas FPGAs no sólo incluyen un mayor área hardware para implementar aceleradores específicos, además incluyen procesadores y recursos de memoria empotrados. Esta opción ofrece una versatilidad en el funcionamiento de diversas aplicaciones software en procesadores empotrados, aprovechando los recursos de hardware reconfigurable, todos ellos dentro del mismo circuito integrado. Estos sistemas hardware/software estrechamente acoplados combinan la flexibilidad de los microprocesadores tradicionales con la potencia y el rendimiento de las implementaciones hardware a medida, dando lugar a nuevas arquitecturas para misiones de observación remota de la Tierra. Existen al menos tres compañías que han desarrollado sistemas con microprocesadores de 8 y 32 bits junto con una parte del circuito integrado que contiene lógica reconfigurable [Tri98, Alt05, Xil09]. La ventaja de esta configuración es que el retardo en la comunicación entre el procesador empotrado y la lógica reconfigurable asociada se reduce gracias a la aparición de buses internos. También tiene la ventaja de que el consumo de energía y el cableado externo de entrada/salida se reducen significativamente.

Cabe señalar que los sistemas de observación remota de la Tierra actuales utilizan ampliamente microprocesadores empotrados y periféricos hardware dedicados. Existe mucha flexibilidad en este tipo de sistemas, pero esta flexibilidad está limitada a la parte software del diseño mientras que la parte hardware permanece inalterable desde el momento de su fabricación. Desafortunadamente, la naturaleza de los microprocesadores empotrados es secuencial y emplean mucho tiempo simplemente moviendo datos de un sitio a otro

y no realizando cálculos útiles. El hardware reconfigurable permite alcanzar el grado de flexibilidad necesaria para evitar este cuello de botella. Mediante el empleo de hardware reconfigurable es posible aplicar la mayor parte de la flexibilidad que se limitaba formalmente sólo al desarrollo software, a recursos hardware altamente paralelos. La idea es que las FPGAs pueden ser reconfiguradas en vuelo. Esta aproximación se denomina *particionado temporal* [TMS+03, RSMM06] o *reconfiguración en tiempo real* [RCG+08]. Básicamente, la FPGA (o una región de la FPGA) ejecuta una serie de tareas unas detrás de otras mediante reconfiguración entre tareas [CGRM10]. El proceso de reconfiguración actualiza la funcionalidad implementada en la FPGA, y una nueva tarea puede ser ejecutada. Esta aproximación multiplexada en el tiempo permite la reducción de los componentes hardware a bordo ya que, un simple módulo reconfigurable puede reemplazar muchos periféricos hardware llevando a cabo distintas funcionalidades durante las diferentes fases de la misión.

Tal y como se ha explicado anteriormente, la flexibilidad ofrecida por el hardware reconfigurable puede ser utilizada también para modificar la funcionalidad del instrumento del satélite durante el vuelo, o para recuperarse automáticamente de un mal funcionamiento. Por otra parte, el ciclo de diseño hardware en FPGAs es mucho más corto que en circuitos integrados, principalmente porque el diseño se puede probar en la plataforma de destino desde los primeros pasos del proceso de diseño, evitando así un complejo proceso de fabricación de chips. Otro resultado importante de la computación reconfigurable es la posibilidad de conseguir procesamiento en tiempo real. Esto se debe a que puede proporcionar un mecanismo para la ejecución determinista. Muchos de los módulos hardware asociados con modernos microcontroladores empotrados (tales como los asociados con el procesamiento de interrupciones y operaciones de cache de memoria) tienen una naturaleza altamente indeterminista. Este comportamiento indeterminista compromete la garantía de que todas las restricciones temporales sean siempre satisfechas. La incorporación de hardware reconfigurable da al diseñador de sistemas de observación remota de la Tierra flexibilidad adicional que podría utilizarse para asignar muchas aplicaciones de tiempo crítico a un hardware propio, haciendo que estas funciones sean más independientes de las operaciones indeterministas.

Tabla 3.1: Listado de algunas FPGAs de Xilinx certificadas para misiones espaciales [Xil].

FPGA	**Slices**	**Logic Cells**	**CLB Flip–Flops**	**Block RAM (Kb)**	**PowerPC**
XQR5VFX130	20.480	130.000	81.920	10.728	–
XQR4VLX200	89.088	200.448	178.176	6.048	–
XQR4VFX60	25.280	56.880	50.560	4.176	2
XQR4VFX140	63.168	142.128	126.336	9.936	2
XQR2V3000	14.336	32.256	28.672	1.728	–

Por otra parte, los instrumentos basados en satélites, tan sólo pueden incluir circuitos integrados que hayan sido certificados para funcionar en el espacio. Esto se debe a que los sistemas utilizados en el espacio deben funcionar en un entorno en el que los efectos de radiación tienen un impacto adverso sobre el funcionamiento del circuito integrado [Tho]. La radiación ionizante puede causar errores en las celdas estáticas que se utilizan para almacenar los datos de configuración. Esto afectará a la funcionalidad del circuito y podría provocar un fallo del sistema. Por ello, se requieren FPGAs especiales que proporcionan circuitos de detección de errores y de corrección. Recientemente, han aparecido FPGAs de alta velocidad con una densidad de millones de puertas para apoyar las necesidades de alto rendimiento para aplicaciones de observación remota de la Tierra. De hecho, las FPGAs endurecidas a radiación están siendo muy demandadas para aplicaciones militares y espaciales. Por ejemplo, empresas como Actel Corporation[1] o Xilinx[2] han fabricado FPGAs tolerantes a la radiación durante varios años, destinadas a sistemas de alta fiabilidad en vuelo espacial. Las FPGAs de Actel han estado a bordo en más de 100 lanzamientos y las FPGAs de Xilinx se han utilizado en más de 50 misiones [Tho]. En este trabajo de Tesis Doctoral, se utiliza una FPGA Xilinx Virtex-II Pro XC2VP30 y una FPGA Xilinx Virtex-4 XC4VFX60 como arquitecturas de referencia, porque son similares a otras FPGAs de Xilinx que han sido certificadas para aplicaciones espaciales (ver Tabla 3.1). Están basadas en las mismas arquitecturas por lo que portar nuestros diseños a estas plataformas es un proceso sencillo.

[1] http://www.actel.com
[2] http://www.xilinx.com

Capítulo 4

Algoritmo Pixel Purity Index (PPI)

Si bien buscas, encontrarás.

Platón

Tal y como se vió en el Capítulo 2 dedicado al análisis hiperespectral, el análisis de mezclas espectral ha sido una atractiva meta explorada desde los primeros días de la imaginería hiperespectral hasta nuestros días, debido principalmente a que los espectros captados en ambientes naturales son inevitablemente una mezcla de los distintos materiales que se encuentran dentro de la extensión espacial visualizada por el sensor hiperespectral. Una técnica estándar para el análisis espectral de mezclas es el desmezclado espectral, que comprende dos etapas: 1) extracción de endmembers, y 2) estimación de la abundancia de dichos endmembers a nivel subpíxel. La correcta búsqueda del conjunto de endmembers determinará el error cometido en la estimación de abundancias, por lo que resulta imprencindible seleccionar el conjunto de endmembers más adecuado.

El algoritmo pixel purity index (PPI) [Boa93] se emplea ampliamente en el análisis de imágenes hiperespectrales para la extracción de endmembers debido a su disponibilidad en el software comercial Environment for Vi-

sualizing Images (ENVI) de ITT Visual Information Solutions (ITTVIS)[1], desarrollado originalmente por Analytical Imaging and Geophysics (AIGs)[2]. El algoritmo busca un conjunto de vértices de una envolvente convexa en el conjunto de datos dado, suponiendo que son firmas puras presentes en los datos. Debido a los derechos de propiedad y a los limitados resultados publicados, los detalles de su implementación nunca se han dado a conocer. Por lo tanto, la mayoría de las personas que utilizan el algoritmo PPI para la extracción de endmembers o bien recurren al software ENVI o implementan sus propias versiones del algoritmo basadas en la información disponible en la literatura. El procedimiento general del algoritmo PPI se puede resumir de la siguiente manera [PC06].

1. En primer lugar, se calcula el índice de pureza para cada píxel $\mathbf{f}$ del cubo de datos hiperespectral de entrada $\mathbf{F}$ mediante la generación de K vectores aleatorios N-dimensionales, llamados *skewers*.

2. Entonces, se proyecta cada uno de los píxeles $\mathbf{f}$ de la imagen hiperespectral de entrada sobre el conjunto de skewers $\{\mathbf{skewer}_j\}_{j=1}^{K}$ generados anteriormente, y se identifican los píxeles extremos en la dirección definida por cada skewer (ver Figura 4.1). Después de calcular un gran número de proyecciones sobre distintos skewers aleatorios, los píxeles que han sido seleccionados como extremos en varias ocasiones durante el proceso se identifican y se colocan en una lista de candidatos a endmembers.

3. Los espectros de los potenciales endmembers se cargan en una herramienta interactiva (como el visualizador N-dimensional disponible en ENVI) y se rotan hasta que visualmente se identifican el número deseado de endmembers como píxeles extremos de la nube de datos.

El algoritmo PPI presenta varias limitaciones [CP06]. La primera y más importante, es que el algoritmo es muy sensible al parámetro K, que denota el número de skewers. Dado que los skewers se generan de manera aleatoria, generalmente se requiere un gran número de proyecciones sobre skewers, a

[1] http://www.ittvis.com

[2] Research Systems, *ENVI User's Guide*. Boulder, CO: Research Systems, Inc., 2001

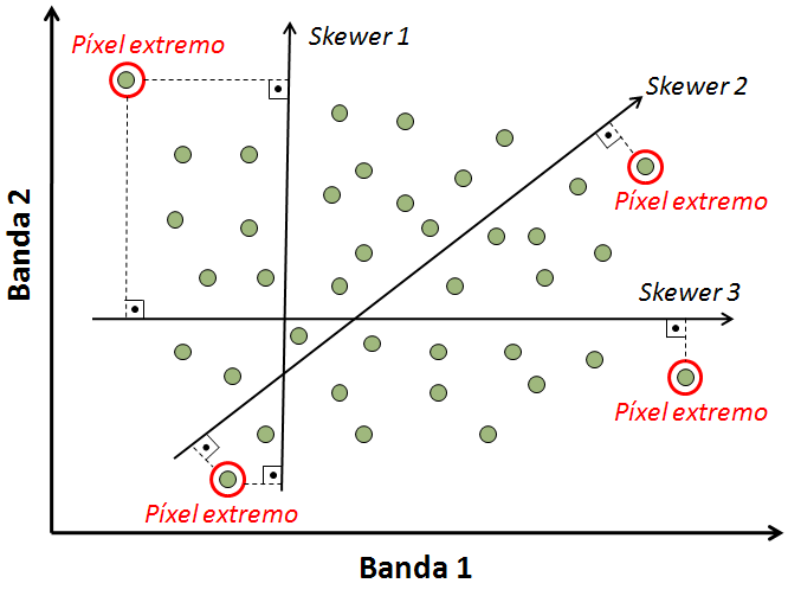


Figura 4.1: Ejemplo básico que ilustra el funcionamiento del algoritmo de extracción de endmembers PPI en un espacio de 2 dimensiones.

fin de llegar a obtener un conjunto de endmembers satisfactorio en términos de pureza de las firmas. Los investigadores recomiendan utilizar el mayor número de skewers aleatorios posible con el fin de obtener resultados óptimos [Boa93]. Como consecuencia, el algoritmo PPI sólo puede garantizar producir resultados óptimos asintóticamente y su complejidad computacional es muy alta, por lo que se requiere una implementación eficiente. Otra carencia del algoritmo PPI es la necesidad de una herramienta interactiva para llevar a cabo la selección final de endmembers. Una alternativa es mantener los píxeles que han sido seleccionados por encima de un umbral predefinido y luego eliminar automáticamente endmembers con espectros redundantes [CP06]. Esto último, se aborda generalmente como una etapa de posprocesamiento externo al algoritmo.

En este capítulo, se presenta un diseño para la implementación en FPGA del algoritmo PPI. Se han escogido las FPGAs Virtex–II Pro XC2VP30 y Virtex–4 XC4VFX60 de Xilinx como arquitecturas representativas para demostrar el rendimiento de la implementación propuesta ya que, existen FPGAs endurecidas para radiación con las mismas arquitecturas y áreas si-

milares que han sido certificadas para aplicaciones de observación remota de la Tierra por varias agencias internacionales [Xil]. El diseño en forma de array sistólico incluye un DMA e implementa una técnica de prebúsqueda para reducir las penalizaciones debidas a las comunicaciones de entrada/salida (E/S). Además, se ha incluido un módulo hardware para la generación de números aleatorios. La implementación propuesta ha sido validada utilizando imágenes hiperespectrales reales captadas por el Airborne Visible Infra–Red Imaging Spectrometer (AVIRIS) de la NASA sobre la región minera de Cuprite en Nevada y sobre la reserva biológica de Jasper Ridge en California, y la captada por el satélite EO–1 Hyperion a lo largo del mismo distrito minero de Cuprite, al igual que la citada escena AVIRIS. Los resultados experimentales revelan que el sistema hardware propuesto es fácilmente escalable y es capaz de proporcionar resultados precisos con un tamaño compacto alcanzando un gran rendimiento, lo que convierte al sistema reconfigurable propuesto en atractivo para el procesamiento a bordo de imágenes hiperespectrales.

4.1. Descripción del algoritmo

Dado que los detalles de la secuencia de pasos específicos para implementar el algoritmo PPI del software ENVI son desconocidos, el algoritmo PPI descrito a continuación se basa en los limitados resultados publicados y en nuestra propia interpretación [CP06]. Sin embargo, excepto un paso final supervisado (incluido en ENVI) que es reemplazado por el paso 4, la aproximación propuesta y el algoritmo PPI de ENVI 4.0 producen resultados muy similares. Las entradas del algoritmo son un cubo de datos hiperespectral $\mathbf{F}$ con N dimensiones; el número de skewers aleatorios que deben ser generados durante el proceso, K; y un valor umbral de corte, t_v, utilizado para seleccionar como endmembers finales sólo aquellos píxeles que han sido seleccionados como píxeles extremos al menos t_v veces durante el proceso PPI.

El algoritmo consta de los siguientes pasos:

1. *Generación de skewers.* Produce un conjunto de K vectores unitarios

generados aleatoriamente $\{\mathbf{skewer}_j\}_{j=1}^{K}$.

2. *Cálculo de proyecciones.* Para cada $\mathbf{skewer}_j$, $j = \{1, \cdots, K\}$, todos los píxeles $\mathbf{f}_i$ del conjunto original de datos $\mathbf{F}$ se proyectan en el $\mathbf{skewer}_j$ mediante dot–products (productos escalares) de $|\mathbf{f}_i \cdot \mathbf{skewer}_j|$ para encontrar los píxeles de la muestra que son extremos (máximo y mínimo), formando así un conjunto de extremos para el $\mathbf{skewer}_j$ que representamos por $S_{extremos}(\mathbf{skewer}_j)$. A pesar del hecho, de que diferentes skewers generen un conjunto distinto de extremos, es muy probable que algunos píxeles de la muestra aparezcan en más de un conjunto de extremos. Para tener en cuenta esto, se define la función $I_S(\mathbf{x})$, para indicar pertenencia de un elemento $\mathbf{x}$ a ese conjunto particular de la siguiente manera:

$$I_S(\mathbf{x}) = \begin{Bmatrix} 1 \text{ si } \mathbf{x} \in S \\ 0 \text{ si } \mathbf{x} \notin S \end{Bmatrix} \tag{4.1}$$

3. *Cálculo de la puntuación PPI.* Utilizando la función siguiente, se calcula la puntuación PPI asociada a cada píxel $\mathbf{f}_i$ (número de veces que un píxel dado ha sido seleccionado como extremo):

$$N_{PPI}(\mathbf{f}_i) = \sum_{j=1}^{k} I_{S_{extremos}(\mathbf{skewer}_j)}(\mathbf{f}_i) \tag{4.2}$$

4. *Selección de endmembers.* Encontrar los píxeles que tengan una puntuación $N_{PPI}(\mathbf{f}_i)$ que esté por encima de t_v y etiquetarlos como endmembers.

La parte del algoritmo PPI que más tiempo consume es el paso 2 (*cálculo de proyecciones*). Por ejemplo, ejecutar este paso en una imagen hiperespectral con 614 x 512 píxeles (el número estándar de píxeles producido por el instrumento AVIRIS de la NASA en un sólo frame), cada uno con 224 bandas espectrales, y utilizando K = 400 skewers requiere el cálculo de más de 2 x 10^{11} operaciones de multiplicación/acumulación (MAC), lo que supone, unas pocas horas de cálculo ininterrumpido en un procesador a 500 MHz con 256 MBytes de SDRAM [LFDW99, LTS+02]. En [VPVRP05], se muestra otro ejemplo en el que el algoritmo PPI disponible en la versión ENVI

4.0 requirió más de 50 minutos de cálculos para proyectar todos los píxeles de la imagen hiperespectral, del tamaño mencionado, en 10^4 skewers en un PC con un procesador AMD Ahtlon 2.6 GHz y una RAM de 512 MB.

Afortunadamente, el algoritmo PPI es muy adecuado para ser paralelizado. El cálculo de las proyecciones de los píxeles en los skewers son independientes y pueden ser ejecutados simultáneamente, dando lugar a diversas formas de paralelización. En [LTS+00], se proponen dos arquitecturas paralelas para la implementación del algoritmo PPI. Ambas se basan en un procesador vectorial 2-D firmemente conectado a una memoria de unos pocos bancos. Mediante la implementación del algoritmo en FPGA en la placa Wildforce (4 Xilinx XC4036EX más 4 bancos de memoria de 512 KBytes) se obtiene un *speedup* de 80 con respecto a la implementación del algoritmo en el lenguaje C y ejecutado en un único procesador a 450 MHz. Sin embargo, este diseño está adaptado a la placa Wildforce y no puede reutilizarse para otra placa sin realizar grandes modificaciones. En [CP06], se presenta un método iterativo PPI rápido (FPPI). La implementación en Matlab del algoritmo FPPI fue más de 24 veces más rápida que el algoritmo PPI del software comercial ENVI 4.0 en el mismo entorno computacional, mientras que la implementación basada en FPGA mostró un aumento significativo en el rendimiento con respecto a las dos versiones software consideradas debido a la implementación hardware de bajo nivel. Si bien se ha demostrado la eficacia de una implementación hardware en una placa reconfigurable, estas soluciones no son escalables.

La implementación en FPGA que se presenta en la siguiente sección tiene por objetivo superar estos inconvenientes. En primer lugar, la arquitectura especificada se puede adaptar fácilmente a diferentes plataformas. En segundo lugar, la arquitectura propuesta es escalable en función de la cantidad de recursos disponibles, ya que los recursos necesarios crecen proporcionalmente con el número de skewers y el ciclo de reloj permanece constante.

4.2. Implementación en FPGA

Esta sección se organiza en los siguientes dos apartados. En el primero de ellos, el apartado 4.2.1, se ofrece un estudio de las distintas estrategias de

paralelización posibles para el algoritmo PPI y se establece así, el diseño a seguir. El segundo, el apartado 4.2.2, describe la implementación basada en el criterio de diseño del apartado anterior, explicando el funcionamiento de cada módulo hardware diseñado.

4.2.1. Estrategias de paralelización para el algoritmo PPI

En el paso que más tiempo consume del algoritmo PPI (*cálculo de proyecciones*), se calculan un gran número de dot–products y todos ellos pueden calcularse simultáneamente, dando lugar a diversas formas de paralelización. Si consideramos la unidad dot–product (dp) mostrada en la Figura 4.2(a) como base para la ejecución en paralelo, entonces podemos paralelizar por píxeles (ver Figura 4.2(b)), por skewers (ver Figura 4.2(c)), o por skewers y píxeles (ver Figura 4.2(d)). Si paralelizamos la ejecución por píxeles, se requiere hardware adicional para comparar todos los máximos y mínimos entre ellos. A medida que aumentamos el número de cálculos en paralelo, se requiere mayor área para calcular los máximos/mínimos y el camino crítico aumenta (menor frecuencia). Otra posible forma de paralelización es calcular K dot–products simultáneamente para el mismo píxel, donde K es el número de skewers (ver Figura 4.2(c)). Si aumentamos el número de skewers, el área requerida crecerá proporcionalmente con el número de unidades dot–product y el ciclo de reloj se mantendrá constante. Por último, la estrategia de paralelización de la Figura 4.2(d) es una solución mixta que no proporciona ninguna ventaja adicional con respecto a la paralelización por skewers y tiene los mismos problemas que la paralelización por píxeles.

Teniendo en cuenta lo expuesto anteriormente, en este trabajo se ha seleccionado la estrategia de paralelización por skewers. A parte de las ventajas mencionadas con respecto a otras posibles estrategias, otra de las razones para esta elección es que esta estrategia se adecua bien a cómo llegan los datos de la imagen al sistema. Nuestro objetivo es hacer un procesamiento a bordo de las imágenes hiperespectrales y hay que tener en cuenta que los sensores hiperespectrales capturan los datos de la imagen píxel a píxel. Por lo tanto, la paralelización basada en skewers es la que mejor se ajusta al mecanismo de entrada de datos donde cada píxel puede ser procesado inmediatamente después de haber sido recogido por el sensor. En concreto, nuestro sistema

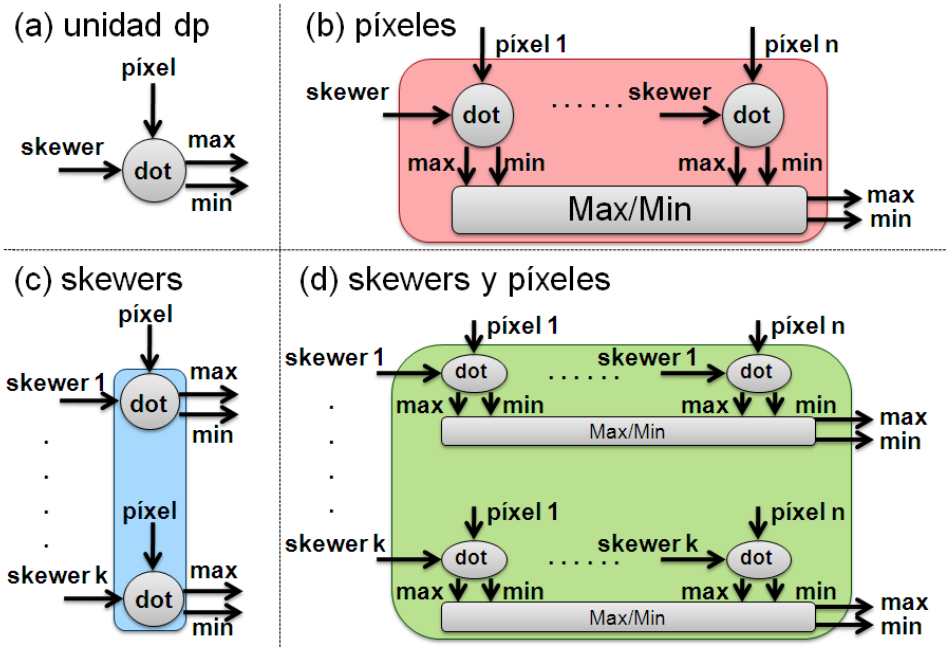


Figura 4.2: Estrategias de paralelización para el algoritmo PPI. (a) Unidad dot–product (dp). (b) Paralelización por píxeles. (c) Paralelización por skewers. (d) Paralelización por skewers y píxeles.

hardware es capaz de calcular K dot–products simultáneamente con el mismo píxel $\mathbf{f}_i$, donde K es el número de skewers. En tal sistema, el paso de *cálculo de proyecciones* del algoritmo PPI (el que más tiempo consume en el proceso PPI) puede ser simplemente escrito como se describe en el Algoritmo 1.

El bucle **par** en el Algoritmo 1 expresa que primero se realizan K dot–products en paralelo, posteriormente se realizan K operaciones para calcular el mínimo (Min) y el máximo (Max) también en paralelo. Ahora bien, si suponemos que no podemos llevar a cabo el cálculo de K dot–products de manera simultánea pero sí una fracción K/P, entonces K/P debería ser el número disponible de unidades de procesamiento en la plataforma paralela, por lo que el *cálculo de proyecciones* debe ser dividido en P iteraciones, cada una calculando F x K/P dot–products, como indica el Algoritmo 2. Desde el punto de vista de la arquitectura, cada procesador recibe sucesivamente los F píxeles, calcula K/P dot–products, y mantiene en memoria el mínimo

Algoritmo 1 Implementación paralela del paso *cálculo de proyecciones*

```
for (f = 0; f < F; f++) { //F denota el número de píxeles
  par (k = 0; k < K; k++) { //K denota el número de skewers
    dp[k]=dot_product(pixels[f],skewers[k]);
    if (dp[k] < Min[k]) { Min[k]=dp[k]; Reg_Min[k]=f; }
    if (dp[k] > Max[k]) { Max[k]=dp[k]; Reg_Max[k]=f; }
  }end par
}end for
```

Algoritmo 2 Implementación paralela del paso *cálculo de proyecciones* (reescrito para ser dividido en P iteraciones)

```
for (p = 0; p < P; p++) { //P es el número de iteraciones del algoritmo
  x = p × (K/P); //K denota el número de skewers
  for (f = 0; f < F; f++) { //F denota el número de píxeles
    par(k = 0; k < K/P; k++) {
     dp[x + k]=dot_product(pixels[f],skewers[x + k]);
     if (dp[x + k] < Min[x + k]) {
       Min[x + k]=dp[x + k]; Reg_Min[x + k]=f; }
     if (dp[x + k] > Max[x + k]) {
       Max[x + k]=dp[x + k]; Reg_Max[x + k]=f; }
    }end par
  }end for
}end for
```

y el máximo dot–product. Con este esquema, cada procesador mantiene un skewer diferente que debe ser recibido antes de cada iteración.

Para concluir esta sección, se vuelve a hacer hincapié en las ventajas de la estrategia de paralelización considerada frente a las otras alternativas posibles. Para este propósito, las Figuras 4.3 y 4.4 comparan las tres estrategias de la Figura 4.2: paralelización por píxeles, paralelización por skewers y paralelización por skewers y píxeles. Las diferentes estrategias de paralelización para el algoritmo PPI se han descrito mediante el lenguaje VHDL utilizando el entorno Xilinx ISE para implementarlas y de esta manera, obtener la cantidad de recursos necesarios (ver Figura 4.3) y el ciclo de reloj (ver Figura 4.4) en cada una de las tres estrategias de paralelización consideradas para

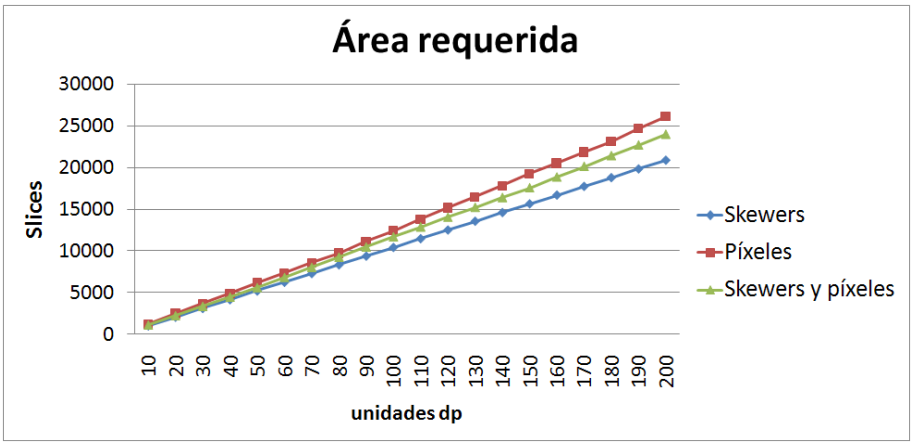


Figura 4.3: Área necesaria (en slices) por unidades dot–products en las diferentes estrategias de paralelización.

el mismo número de unidades dot–product. Como se muestra en las Figuras 4.3 y 4.4, la paralelización por skewers ofrece ventajas significativas con respecto al resto de estrategias consideradas: requiere menos slices de la FPGA, llegando a consumir un 25 % y un 15 % menos de recursos en el caso de 200 unidades dot–product en paralelo con respecto a la paralelización por píxeles y a la paralelización por skewers y píxeles. Por otra parte, la frecuencia de reloj es mayor y esta diferencia se incrementa a medida que aumenta el número de unidades dot–product, al ser más estable que en los otros casos.

4.2.2. Implementación hardware

La Figura 4.5 muestra la arquitectura hardware utilizada para la implementación del algoritmo PPI, junto con las comunicaciones de E/S. Para la entrada de datos se emplea una memoria DDR2 SDRAM y un DMA (controlado por el PowerPC) con una FIFO de escritura para almacenar los datos de los píxeles. Para el envío de datos empleamos un transmisor RS232. Finalmente se incluyen un array sistólico, un módulo de generación aleatoria y una memoria (para almacenar el número de apariciones de cada píxel como extremo) para implementar la versión propuesta del algoritmo PPI.

El diseño propuesto tiene por objetivo mejorar el escalado y proporcionar una implementación eficaz en dispositivos FPGA a través de la utilización

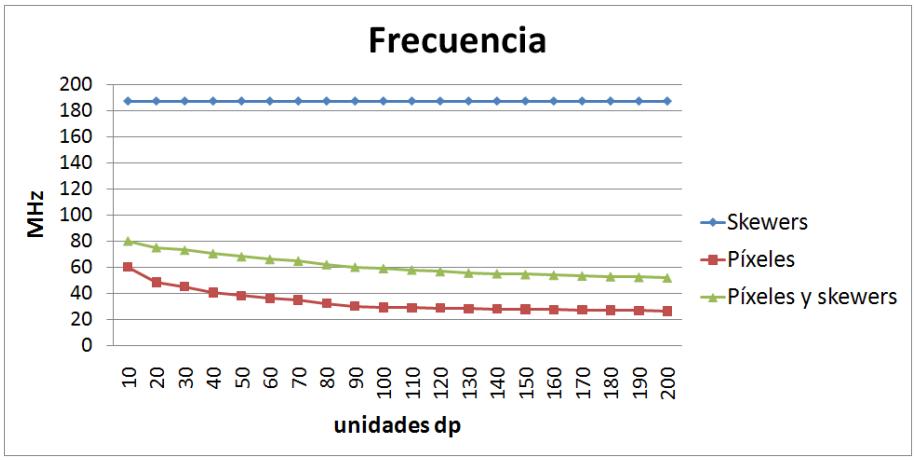


Figura 4.4: Frecuencia (en MHz) por unidades dot–products en las diferentes estrategias de paralelización.

de un array sistólico. Una de las principales ventajas de las implementaciones basadas en un array sistólico, es que proporcionan un procedimiento sistemático para el diseño del sistema que permite la derivación de una estructura bien definida basada en elementos de procesamiento y un patrón de interconexión que puede ser fácilmente adaptado a configuraciones hardware reales.

La Figura 4.6 describe la arquitectura de los procesadores dot–product empleados en el diseño del array sistólico. Básicamente, un ciclo sistólico consiste en calcular el dot–product entre un píxel y un skewer y almacenar el índice del píxel si el resultado es mayor o menor que un valor Max/Min calculado previamente. Recordemos que un píxel es un vector de N valores espectrales, exactamente igual que un skewer. Para calcular el dot–product entre un píxel $\mathbf{f}_i$ y un $\mathbf{skewer}_j$ se utiliza la expresión $\sum_{k=1}^{N} \mathbf{f}_i^{(k)} \times \mathbf{skewer}_j^{(k)}$. Por lo tanto, el cálculo completo de un dot–product requiere N multiplicaciones y N - 1 sumas, donde N es el número de bandas espectrales. Como se ha demostrado en trabajos anteriores [LTS+02], los valores de los skewers pueden limitarse a un conjunto muy pequeño de números enteros cuando N es grande, como es el caso de las imágenes hiperespectrales. Un conjunto particular e interesante es $\{1, -1\}$ ya que evita la multiplicación. Por tanto, el dot–product se reduciría a una acumulación de valores positivos y negativos. Con los supuestos anteriores en mente, cada procesador dot–product

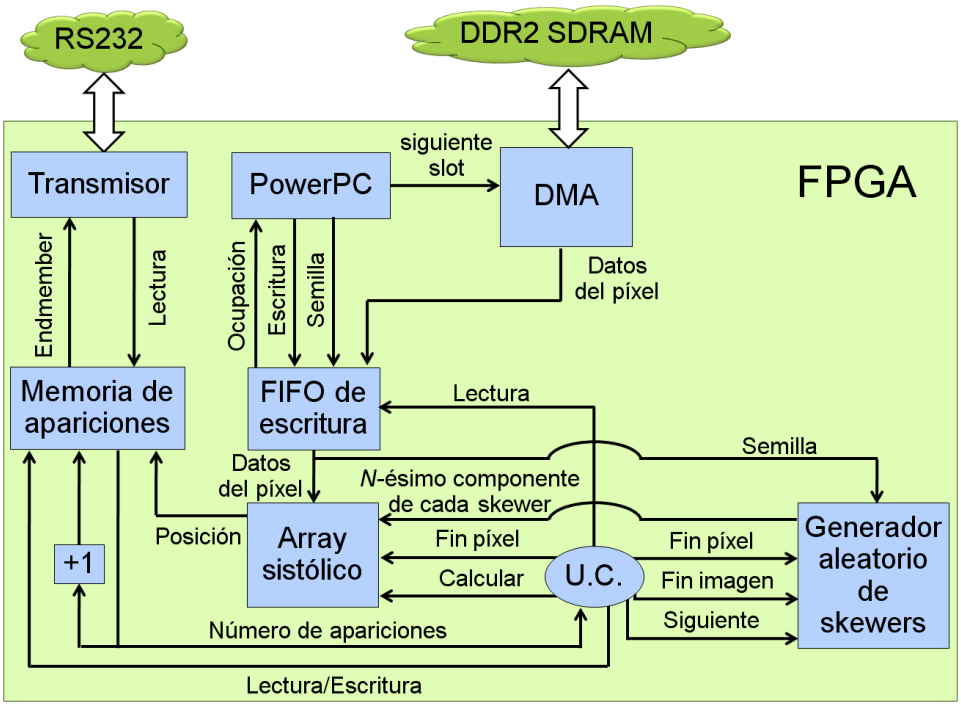


Figura 4.5: Arquitectura hardware para la implementation del sistema PPI completo.

sólo necesita acumular los valores positivos o negativos del píxel según los valores del skewer de entrada. Estas unidades están, por tanto, compuestas solamente por una única unidad de suma/resta y un registro de acumulación. La unidad Min/Max recibe el resultado del dot–product y lo compara con los valores previos de mínimo y máximo. Si el resultado es un nuevo mínimo o máximo, serán almacenados para futuras comparaciones junto con su índice dentro de la imagen correspondiente. Por simplicidad, la parte relacionada con el manejo de índices se ha omitido en la Figura 4.6.

Teniendo en cuenta que la latencia de una suma o una resta es justamente un ciclo de reloj, entonces el cálculo de un dot–product requiere $N + 1$ ciclos de reloj. En cada ciclo, el procesador recibe secuencialmente el dato de cada una de las bandas de un píxel y acumula el resultado, sumando o restando, dependiendo del valor del componente del skewer. El ciclo adicional es necesario para la comparación del mínimo y el máximo y el resultado producido por el píxel. Se han evaluado diferentes alternativas para eliminar

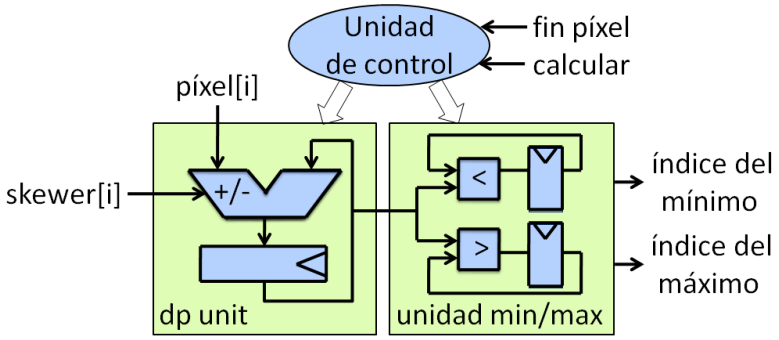


Figura 4.6: Arquitectura hardware de un procesador dot–product.

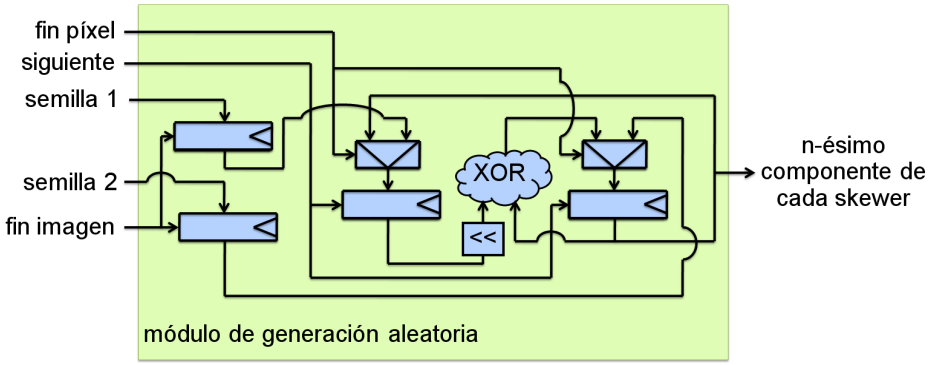


Figura 4.7: Arquitectura hardware del módulo de generación aleatoria.

ese ciclo de reloj adicional, pero finalmente se ha decidido mantenerlo. Una opción, era actualizar los índices del máximo y el mínimo en paralelo con el cálculo del siguiente dot–product, pero esto requiere un mecanismo hardware más complejo (al menos dos registros más) y hace que esta solución sea peor globalmente ya que, se pueden sintetizar menos procesadores sistólicos en la FPGA. También, se puede actualizar el píxel durante el último ciclo de reloj de cada ciclo sistólico, pero esto haría que se incrementase el camino crítico y por tanto la frecuencia de reloj. Por lo tanto, cuando N es un número grande (como en el caso de las imágenes hiperespectrales), se obtiene un mayor tiempo de cómputo.

Una de las principales características del sistema propuesto es la incorporación de un módulo de generación aleatoria hardware que reduce significativamente las comunicaciones de E/S que, en implementaciones anteriores del algoritmo PPI, fueron el principal cuello de botella [VPVRP05, LFDW99,

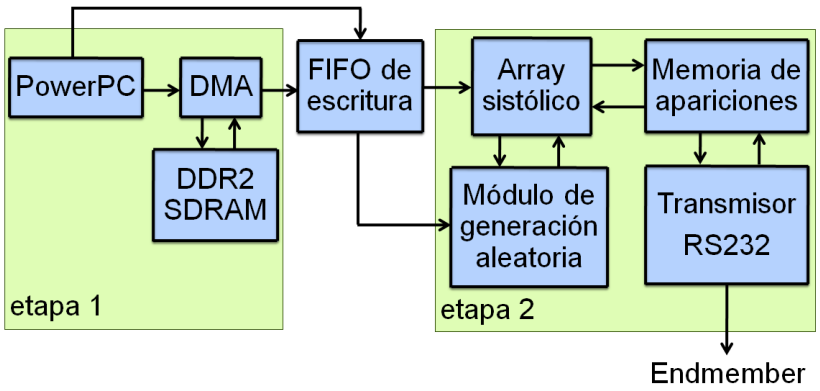


Figura 4.8: Segmentación de la arquitectura para el algoritmo PPI.

LTS+02]. En este trabajo, se ha implementado un módulo generador aleatorio similar al presentado en [Gor06]. Este módulo proporciona secuencias pseudoaleatorias y uniformemente distribuidas utilizando registros y puertas XOR. La Figura 4.7 muestra la estructura del módulo de generación aleatoria que implementa la función $X^{n+1} = X^n \oplus (X^{n-1} << 1)$. Tiene dos registros para almacenar las nuevas semillas. Estas semillas son inicializadas por el sistema cada vez que el algoritmo PPI computa la imagen. Al comienzo de cada ciclo sistólico, estas dos semillas también se almacenan en los otros dos registros. Este generador reduce el número de recursos necesarios, ya que no es necesario almacenar los N bits de K skewers, tan sólo los K bits de las dos semillas. Se requiere una cantidad asequible de espacio (288 slices para 100 skewers) y es capaz de generar el siguiente componente de cada skewer en un solo ciclo de reloj, funcionando a una alta frecuencia de reloj (664 MHz).

Para contabilizar el número de veces que cada uno de los píxeles ha sido seleccionado como extremo se utiliza la memoria de apariciones, que se inicializará a cero en el bitstream de configuración. Una vez que se ha calculado el mínimo y el máximo para cada uno de los skewers, se actualizará el número de apariciones mediante la lectura de la memoria de apariciones en la posición que ocupa ese mínimo o máximo y la posterior escritura de su anterior número de apariciones incrementado en uno. Una vez concluido este proceso, el transmisor leerá el número de apariciones total de cada uno de los píxeles y si supera el valor umbral t_v será enviado como endmember a través del puerto RS232.

La arquitectura propuesta se puede ver como el pipeline que se muestra en la Figura 4.8. Podemos distinguir dos etapas que se comunican utilizando una estructura first-in first-out (FIFO): La primera etapa proporciona los datos necesarios (semillas y datos de la imagen) al sistema y la segunda etapa calcula las proyecciones y envía los endmembers a través de un puerto RS232. Por lo tanto, las dos etapas están trabajando en paralelo.

Para concluir esta sección, se ofrece una descripción paso a paso de cómo la arquitectura propuesta realiza la extracción de un conjunto de endmembers a partir de una imagen hiperespectral:

1. En primer lugar, para inicializar el módulo de generación aleatoria, el PowerPC genera dos semillas de K bits (donde K es el número de skewers) y las escribe en la FIFO de escritura.

2. Posteriormente, la unidad de control lee estas semillas y las envía al módulo de generación aleatoria, donde se almacenan. Desde ahora, el módulo de generación aleatoria puede proporcionar al array sistólico un bit por cada skewer en cada ciclo de reloj como se ha descrito en este apartado.

3. Después de que el PowerPC ha escrito las dos semillas, envía una orden al DMA para comenzar a copiar un trozo de la imagen de la memoria DDR2 SDRAM a la FIFO de escritura. Como se mencionó anteriormente, el principal cuello de botella en este tipo de sistema es con frecuencia la entrada de datos que se aborda en la implementación propuesta con la incorporación de un DMA que elimina la mayor parte de las penalizaciones de E/S. Por otra parte, el PowerPC monitorea la FIFO de entrada y envía una nueva orden al DMA cada vez que se detecta que la FIFO de escritura está medio vacía. Esta vez, el DMA traerá un trozo de la imagen que ocupa la mitad de la capacidad total de la FIFO de escritura.

4. Cuando los datos del primer píxel se han escrito en la FIFO de escritura, el array sistólico y el módulo de generación aleatoria comienzan a trabajar. Cada ciclo de reloj, un nuevo píxel es leído por la unidad de control y es enviado al array sistólico. Al mismo tiempo, el compo-

nente k-ésimo de cada skewer también se envía al array sistólico desde el módulo de generación aleatoria.

5. Durante N ciclos de reloj, los datos de un píxel se acumulan de manera positiva o negativa dependiendo del valor del componente del skewer. En el ciclo de reloj siguiente, la unidad Min/Max actualiza el valor y el índice del máximo y del mínimo y el módulo de generación aleatoria restaura las dos semillas, concluyendo el ciclo sistólico. Con el fin de procesar la imagen hiperespectral, necesitamos tantos ciclos sistólicos como píxeles hay en la imagen.

6. Cuando toda la imagen es procesada, se actualiza el número de apariciones como extremo de todos los mínimos y máximos encontrados, mediante lecturas y escrituras en la memoria de apariciones.

7. Estos pasos se repiten varias veces en función del número de skewers que se puedan paralelizar y del número total de skewers que queramos evaluar.

8. Por último, el transmisor lee de la memoria de apariciones el número de veces que cada píxel ha sido extremo y si supera un umbral preestablecido, se selecciona como endmember y se envía a través de un puerto RS232.

4.3. Resultados experimentales

4.3.1. Plataformas reconfigurables

La arquitectura hardware descrita en la sección 4.2 ha sido implementada utilizando el lenguaje VHDL para la especificación del array sistólico y del módulo de generación aleatoria. Además, se han empleado los entornos Xilinx ISE y Embedded Developed Kit (EDK)[3] para especificar el sistema completo. Este sistema ha sido implementado en las placas XUPV2P (ver Figura 4.9) y ML410 (ver Figura 4.10), placas reconfigurables de bajo coste con

[3]http://www.xilinx.com/ise/embedded/edk pstudio.html

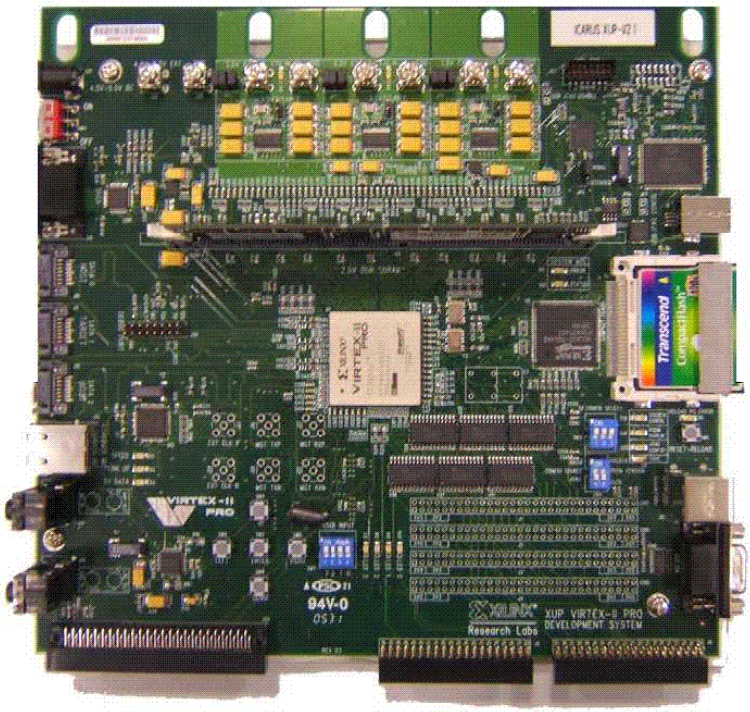


Figura 4.9: Placa XUPV2P con un solo componente FPGA Virtex-II Pro XC2VP30 de Xilinx.

una sola FPGA (Virtex-II Pro XC2VP30 y Virtex-4 XC4VFX60 respectivamente), una ranura DDR2 SDRAM que admite hasta 2 GBytes, un puerto RS232 y algunos componentes adicionales que no se han utilizado en la implementación propuesta. Utilizamos las FPGAs Virtex-II Pro XC2VP30 y Virtex-4 XC4VFX60 de Xilinx, porque se basan en las mismas arquitecturas que otras FPGAs endurecidas para radiación [Xil] que han sido certificadas por diversos organismos internacionales para operar en condiciones espaciales. En concreto, la FPGA Virtex-4 XC4VFX60 es muy parecida a la FPGA Virtex-4QV XQR4VFX60 certificada para el espacio, por lo que el diseño propuesto se puede implementar en ella de manera inmediata.

4.3.2. Conjunto de imágenes hiperespectrales

Cuatro conjuntos diferentes de datos hiperespectrales se han utilizado en los experimentos realizados:

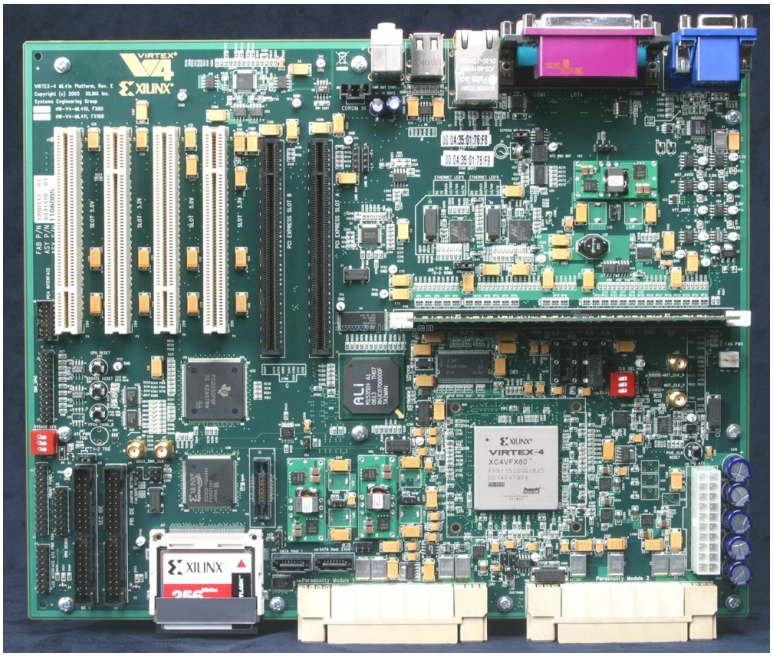


Figura 4.10: Placa ML410 con un solo componente FPGA Virtex-4 XC4VFX60 de Xilinx.

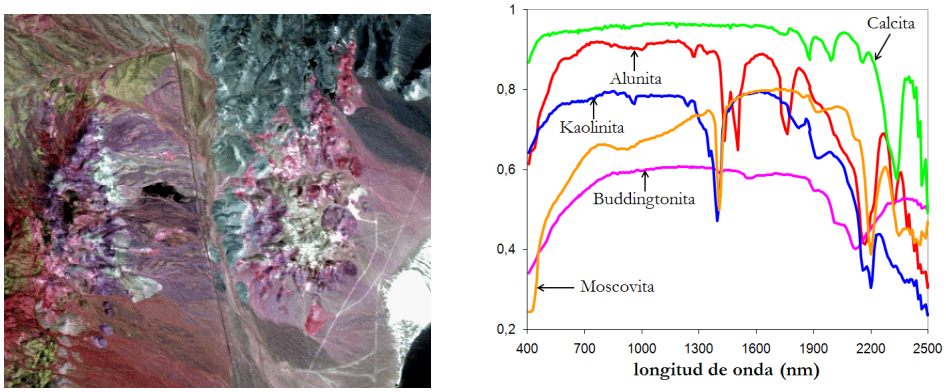
- El primer conjunto de datos corresponde a la conocida escena AVIRIS Cuprite (ver Figura 4.11(a)), recogida en el verano de 1997 y disponible online en unidades de reflectancia después de ser corregida atmosféricamente[4]. La porción utilizada en los experimentos corresponde a un subconjunto de 350 × 350 píxeles del sector, etiquetados como f970619t01p02_r02_sc03.a.rfl en los datos online, que cuenta con 224 bandas espectrales en el rango de 400 a 2500 nanómetros y un tamaño total de alrededor de 50 megabytes. Las bandas 1-3, 105-115 y 150-170 han sido eliminadas antes del análisis debido a la absorción por agua y la baja relación señal-ruido o signal-to-noise ratio (SNR) de estas bandas. La zona es bien conocida mineralógicamente, y tiene varios minerales expuestos de interés, incluyendo *alunita*, *buddingtonita*, *calcita*, *caolinita* y *moscovita*. Las firmas de referencia de suelo de los minerales mencionados (ver Figura 4.11(b)), disponibles en la biblioteca U.S. Geological Survey library (USGS)[5], se utilizarán para

[4] http://aviris.jpl.nasa.gov
[5] http://speclab.cr.usgs.gov/spectral-lib.html

evaluar la pureza de la firma de los endmembers en este trabajo.

- El segundo conjunto de datos corresponde al sensor EO-1 Hyperion disponible en unidades de radiancia (es decir, no está corregida atmosféricamente). Los datos fueron recopilados a lo largo del mismo distrito minero de Cuprite, al igual que la citada escena AVIRIS, en el verano de 2001. En este caso, se utilizó toda la resolución de la línea de vuelo del sensor EO-1 Hyperion de dimensiones mucho más grandes, es decir, 6479×256 píxeles y 242 bandas espectrales de las cuales las que presentan absorción por agua y las bandas con baja SNR, también fueron eliminadas antes del análisis. Puesto que la escena no está corregida atmosféricamente, no es posible establecer comparaciones entre los endmembers de la imagen y las firmas espectrales de la librería USGS, sin embargo, se utilizará esta escena para ilustrar el rendimiento del procesamiento con un conjunto de datos mucho más grande (con un tamaño total de alrededor de 800 megabytes).

- Los datos del tercer y cuarto conjunto corresponden a dos grupos de datos AVIRIS de la reserva biológica de Jasper Ridge en California. Los datos están disponibles en unidades de radiancia (sin corregir) y reflectancia (corregida atmosféricamente). Cada uno de los conjuntos de datos, adquiridos en abril de 1998, constan de 512×614 píxeles y 224 bandas espectrales (para un tamaño total de alrededor de 140 megabytes cada una). Las bandas de absorción por agua y con baja SNR se eliminaron antes del análisis. En un estudio previo de los materiales de superficie en este área, se derivaron los endmembers de esta imagen basándose en un extenso conocimiento del terreno [GU01]. La Figura 4.12 muestra las firmas espectrales en unidades de radiancia y reflectancia asociadas a los principales materiales constituyentes de la escena Jasper Ridge. Estas firmas, que corresponden a materiales, tales como *tierra*, *bosque siempre verde*, *hierba seca*, *vegetación chaparral* y *lago*, se obtuvieron de la escena de la imagen utilizando un método híbrido que combina inspección visual e información previa sobre la escena. La ubicación de estos materiales también se identifica en la Figura. 4.12. El conocimiento del terreno se utilizó para identificar vegetación homogénea, sombra y las zonas del suelo de la escena. Dentro de esas



(a) (b)

Figura 4.11: (a) Composición de falso color de la escena hiperespectral AVIRIS sobre la región minera de Cuprite en Nevada. (b) Firmas espectrales de los minerales en la librería U.S. Geological Survey utilizadas para la validación.

áreas, los píxeles representativos fueron seleccionados como espectros verdaderos del terreno, comparándolos con una biblioteca espectral con datos de campo, que se utiliza para representar los componentes del paisaje en la escena Jasper Ridge. En este proceso, nos aseguramos de que los espectros de la librería se corresponden con la fenología en el momento que se capturó la imagen, y que había un pequeño error de calibración entre los espectros de campo y los espectros de la imagen.

4.3.3. Evaluación de los endmembers

En este apartado se evalúa la exactitud de los endmembers extraídos por la implementación PPI propuesta utilizando diferentes escenas hiperespectrales. Una cuestión importante en la extracción de endmembers es si los datos hiperespectrales han sido corregidos atmosféricamente o no. Para ello, se han utilizado en los experimentos realizados tanto datos corregidos atmosféricamente como sin corregir, con el fin de evaluar el impacto de este proceso en los resultados finales de extracción de endmembers. Cabe señalar que, si el proceso de extracción de endmembers se va a realizar a bordo del instrumento de imagen, sería necesario decidir si es necesario incluir un módulo de

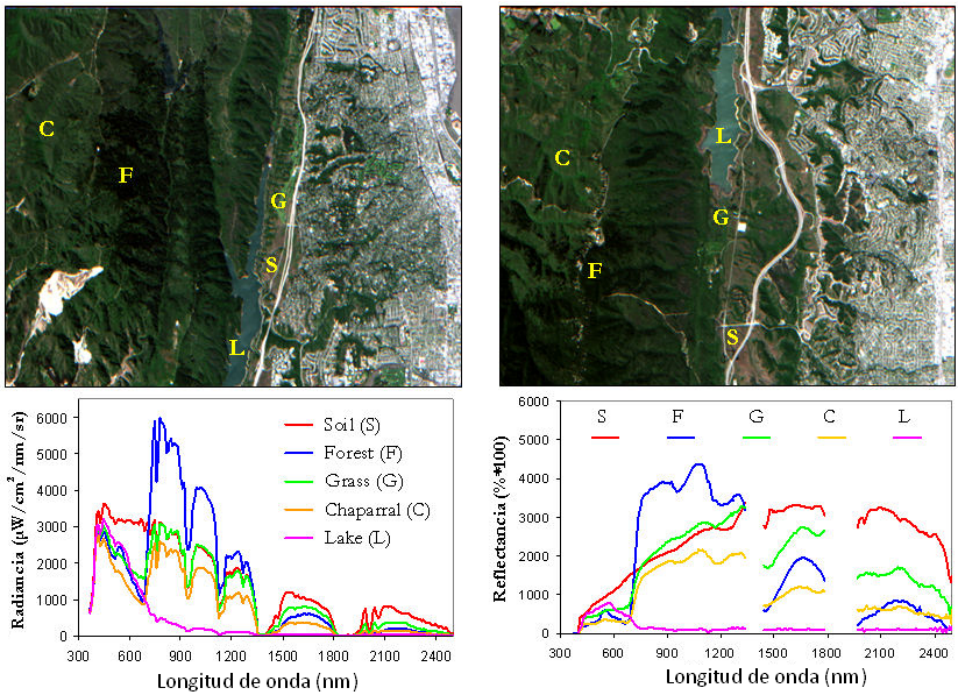


Figura 4.12: Imágenes hiperespectrales AVIRIS recogidas sobre la reserva biológica de Jasper Ridge en unidades de radiancia (izquierda) y reflectancia (derecha), junto con las firmas espectrales y la localización espacial de los endmembers representativos en los dos escenarios considerados.

corrección atmosférica antes de extraer los endmembers. Esto puede tener un impacto importante en la cadena de procesamiento completo. Nuestros resultados experimentales revelan que el algoritmo PPI puede extraer endmembers espectrales de alta calidad en ambos conjuntos de datos, corregidos atmosféricamente y sin corregir.

Antes de describir los resultados obtenidos, se describe primero la métrica utilizada para la comparación cuantitativa en los experimentos realizados. Con el fin de reducir el impacto de las fuentes de interferencia atmosférica en la evaluación realizada, se utiliza el ángulo espectral (AE) entre el endmember más similar detectado por la implementación propuesta y la firma espectral de referencia disponible en cada escena. El AE entre un píxel $\mathbf{X}(i,j)$ seleccionado por el algoritmo PPI y una firma espectral de referencia $\mathbf{S}_i$ dis-

ponible *a priori*, puede calcularse simplemente como:

$$\mathrm{AE}\left[\mathbf{X}(i,j),\mathbf{S}_k\right] = \cos^{-1}\frac{\mathbf{X}(i,j)\cdot\mathbf{S}_k}{\|\mathbf{X}(i,j)\|\cdot\|\mathbf{S}_k\|}, \tag{4.3}$$

es decir, el AE mide el ángulo formado por dos vectores n-dimensionales. Como resultado, valores bajos de AE significan una alta similitud espectral entre los vectores comparados. Esta medida de similitud espectral es independiente de la multiplicación de $\mathbf{X}(i,j)$ y $\mathbf{S}_i$ por constantes y, en consecuencia, es independiente ante escalas multiplicativas desconocidas que puedan surgir debido a las diferencias en la iluminación y al ángulo de incidencia [FP01]. Esto nos puede ayudar a compensar las diferentes condiciones de adquisición para un píxel en la imagen original y para una firma espectral recogida sobre el terreno (como es el caso de las firmas de referencia de la librería USGS utilizadas para la imagen AVIRIS Cuprite).

Debido a la falta de firmas espectrales de referencia de los datos de EO–1 Hyperion, se han llevado a cabo los experimentos sobre la precisión de los endmembers extraídos con las escenas AVIRIS Cuprite y AVIRIS Jasper Ridge con el fin de evaluar la similitud espectral entre los endmembers obtenidos por el algoritmo PPI y las firmas de referencia disponibles para estas escenas, que comprenden un conjunto de firmas de la biblioteca espectral USGS en el caso de AVIRIS Cuprite (disponibles en unidades de reflectancia) y un conjunto de píxeles de la imagen etiquetados como espectralmente puros en el caso de AVIRIS Jasper Ridge (disponible tanto en unidades de radiancia como reflectancia).

Las Tablas 4.1, 4.2 y 4.3 muestran los valores de AE de los endmembers detectados por la implementación original de ENVI (utilizando la herramienta de visualización supervisada N-dimensional para obtener el conjunto final de endmembers), la interpretación del algoritmo descrita en la sección 4.1 (implementada en lenguaje C) y la implementación basada en FPGA, frente a las firmas espectrales de referencia disponibles para las escenas AVIRIS Cuprite y AVIRIS Jasper Ridge tanto en unidades de radiancia como de reflectancia, respectivamente. En todos los casos, se utilizaron $K = 10^4$ skewers. Cabe señalar que las tablas sólo muestran la menor puntuación de AE de todos los endmembers extraídos con respecto a su firma de referencia en

Tabla 4.1: Ángulo espectral entre los endmembers extraídos en la escena AVIRIS Cuprite por las diferentes implementaciones de PPI y las firmas de referencia seleccionadas de la librería USGS.

Mineral en USGS	software ENVI	implementación PPI en C	implementación PPI en FPGA
Alunita	0.084	0.084	0.084
Buddingtonita	0.071	0.068	0.068
Calcita	0.089	0.089	0.089
Kaolinita	0.136	0.132	0.132
Moscovita	0.092	0.081	0.081

Tabla 4.2: Ángulo espectral entre los endmembers extraídos en la escena AVIRIS Jasper Ridge (en unidades de radiancia) por las diferentes implementaciones de PPI y las firmas espectrales puras disponibles.

Mineral en USGS	software ENVI	implementación PPI en C	implementación PPI en FPGA
Suelo	0.065	0.062	0.062
Bosque	0.60	0.058	0.058
Hierba	0.043	0.040	0.040
Chaparral	0.042	0.040	0.040
Lago	0.031	0.028	0.028

Tabla 4.3: Ángulo espectral entre los endmembers extraídos en la escena AVIRIS Jasper Ridge (en unidades de reflectancia) por las diferentes implementaciones de PPI y las firmas espectrales puras disponibles.

Mineral en USGS	software ENVI	implementación PPI en C	implementación PPI en FPGA
Suelo	0.030	0.027	0.027
Bosque	0.026	0.022	0.022
Hierba	0.024	0.021	0.021
Chaparral	0.031	0.019	0.019
Lago	0.019	0.017	0.017

cada caso. Los resultados se muestran en radianes (el rango de valores para el AE es de $[0, \pi/2]$ radianes).

Como se muestra en las Tablas 4.1, 4.2 y 4.3, las dos implementaciones consideradas no producen exactamente los mismos resultados que los obtenidos por el algoritmo original PPI implementado en Research Systems ENVI 4.0. Esto se debe a que la implementación PPI de ENVI tiene un procedimiento de supervisión manual para seleccionar el conjunto final de endmembers, por lo tanto, es dependiente del usuario. En los experimentos realizados con la herramienta de visualización N-dimensional disponible en ENVI, se han efectuado un gran número de rotaciones interactivas con el fin de seleccionar los mejores endmembers posibles. En todos los casos, tanto la interpretación propuesta en 4.1 como la implementación en FPGA de la sección 4.2 producen resultados muy similares a los encontrados por el algoritmo PPI de ENVI, pero de una manera completamente automática.

Tienen particular relevancia los resultados mostrados en las Tablas 4.2 y 4.3, que indican que el algoritmo PPI puede obtener con éxito los endmembers espectrales de datos en unidades de radiancia y reflectancia. Esto abre la pregunta de si el proceso de extracción de endmembers debe llevarse a cabo antes o después de la corrección atmosférica. Dado que el proceso de corrección atmosférica también se puede aplicar una vez que se han identificado las posiciones de los píxeles endmembers, nuestra tesis es que el proceso de extracción de endmembers se puede realizar al mismo tiempo que los datos se recogen en el sensor, sin la necesidad de tener un módulo de corrección atmosférica anterior, que también debería ser implementado en hardware. Este tema será objeto de investigación en nuestro trabajo futuro.

4.3.4. Evaluación del rendimiento

Las Tablas 4.4 y 4.5 muestran los recursos empleados en la implementación hardware propuesta del algoritmo PPI para diferente número de skewers para la FPGA Virtex-II Pro XC2VP30 de la placa XUPV2P (desde $K = 20$ a $K = 100$) y para la FPGA Virtex-4 XC4VFX60 de la placa ML410 (desde $K = 120$ a $K = 200$). Estas FPGAs tienen un total de 13696 y 25280 slices, respectivamente, y disponen de algunos recursos heterogéneos, como

dos PowerPCs y Block RAMs. En la implementación propuesta nos valemos de estos recursos para optimizar el diseño. Un PowerPC monitoriza las comunicaciones y las Block RAMs se emplean para implementar las FIFOs, por lo que la gran mayoría de los slices se utilizan para la implementación del algoritmo PPI. Como muestran la Tablas 4.4 y 4.5, nuestro diseño puede escalarse hasta 100 y 186 skewers en cada caso (por lo tanto, se necesitan $P = 100$ y $P = 54$ iteraciones para procesar $K = 10^4$ skewers). Un comportamiento interesante del array sistólico propuesto es que se puede escalar sin que aumente el retardo del camino crítico (la frecuencia de reloj permanece constante). En comparación con la implementación del algoritmo FPPI en FPGA presentada en [VPVRP05], nuestro array sistólico utiliza la mitad de slices para el mismo número de dot–products y su frecuencia de reloj es 10 veces mayor. Cabe señalar que, en la implementación actual, la imagen hiperespectral completa se almacena en una memoria externa DDR2 SDRAM DIMM. Sin embargo, con un controlador adecuado, pueden ser compatibles otras opciones, como el uso de memoria flash para almacenar los datos hiperespectrales.

Frecuentemente la E/S es el cuello de botella en los sistemas paralelos. Por lo tanto, se ha puesto un especial cuidado en este aspecto. En diseños previos [CP06, VPVRP05], el módulo de generación aleatoria se ubicaba en un procesador externo, por lo que se requerían muchas comunicaciones. Una de las mejoras del sistema propuesto es el desarrollo de un módulo hardware de generación aleatoria basado en el diseño propuesto en [Gor06]. Esta aproximación reduce significativamente las comunicaciones de E/S. Por otra parte, se ha incluido un DMA y se ha aplicado una técnica de prebúsqueda con el fin de ocultar la latencia de las comunicaciones. Básicamente, mientras que el array sistólico está procesando un conjunto de datos, el DMA está buscando los datos siguientes, y almacenándolos en la FIFO de escritura. Teniendo en cuenta la optimización propuesta relativa a la utilización de los recursos disponibles, es importante encontrar un equilibrio entre el número de operaciones de DMA y la capacidad de la FIFO de destino. En otras palabras, es necesario tener la suficiente información en la FIFO de escritura para que nunca deba pararse el array sistólico. Además, cuanto mayor sea la capacidad de la FIFO de escritura, se requerirá un menor número de operaciones de DMA. Se han evaluado diferentes tamaños para la FIFO de

Tabla 4.4: Resumen de los recursos utilizados para la implementación del algoritmo PPI en la FPGA Virtex-II Pro XC2VP30.

Componente	Número de skewers	Número de slice flip flops	Número de 4 input LUTs	Número de slices	Porcentaje total	Frecuencia máxima (MHz)
Array Sistólico	20	2240	3865	2085	15.22	187
	40	4480	7728	4170	30.44	187
	60	6720	11591	6254	45.66	187
	80	8960	15454	8339	60.88	187
	100	11200	19317	10423	76.1	187
Módulo de Generación Aleatoria	20	40	120	58	0.21	664
	40	80	240	115	0.42	664
	60	120	360	173	0.84	664
	80	160	480	230	1.68	664
	100	200	600	288	2.1	664
Transmisor RS232	-	69	128	71	0.52	208
Controlador DMA	-	170	531	367	2.68	102

Tabla 4.5: Resumen de los recursos utilizados para la implementación del algoritmo PPI en la FPGA Virtex-4 XC4VFX60.

Componente	Número de skewers	Número de slice flip flops	Número de 4 input LUTs	Número de slices	Porcentaje total	Frecuencia máxima (MHz)
Array Sistólico	120	13440	23660	12711	53.16	217
	140	15680	27603	14830	58.66	217
	160	17920	31546	16948	67.04	217
	180	20160	35489	19067	75.42	217
	200	22400	39432	21148	83.65	217
Módulo de	120	240	720	346	1.36	684
Generación	140	280	840	403	1.59	684
Aleatoria	160	320	960	461	1.82	684
	180	360	1080	521	2.06	684
	200	400	1200	576	2.27	684
Transmisor RS232	-	69	128	71	0.28	238
Controlador DMA	-	170	531	367	1.45	102

Tabla 4.6: Comparación de las distintas implementaciones del algoritmo PPI para la escena AVIRIS Cuprite.

Algoritmo	**Tiempo de cálculo** (segundos)
Implementación PPI en C	3068
Implementación en FPGA en [VPVRP05]	62
Implementación en FPGA propuesta (Virtex-II Pro XC2VP30)	31.23
Implementación en FPGA propuesta (Virtex-4 XC4VFX60)	16.86

escritura y se ha identificado que para 1024 posiciones o más, no se producen penalizaciones debidas a la entrada de datos. Para demostrar las ventajas de utilizar un DMA, se ha desarrollado otra versión en la que los datos de la imagen son leídos de memoria y escritos en la FIFO de escritura por el PowerPC en lugar del DMA. En esta versión, el tiempo de procesamiento se incrementó en más de un orden de magnitud por lo que podemos concluir que los recursos utilizados para el DMA están bien empleados.

Se ha realizado una comparación del diseño propuesto en FPGA con la implementación presentada en [VPVRP05] para la misma porción de la escena AVIRIS Cuprite. Como se comentó anteriormente, la implementación propuesta puede manejar hasta 100 y 186 skewers en paralelo en cada una de las FPGA mencionadas. La Tabla 4.6 muestra el tiempo de cálculo para tres implementaciones diferentes: nuestra interpretación del algoritmo PPI de la sección 4.1 implementado en el lenguaje C, la implementación en FPGA presentada en [VPVRP05], y la implementación basada en FPGA propuesta en la sección 4.2 de este trabajo. La implementación PPI en software fue ejecutada en un procesador AMD Athlon 2,6 GHz con 512 MB de RAM. La implementación en FPGA de [VPVRP05] se llevó a cabo en una Xilinx Virtex-II XC2V6000-6 con 33792 slices disponibles. Por último, la implementación propuesta en FPGA se llevó a cabo en una Xilinx Virtex-II Pro XC2VP30 y en una Xilinx Virtex-4 XC4VFX60 con 13696 y 25280 slices dis-

Tabla 4.7: Tiempo de ejecución del algoritmo PPI propuesto para las escenas EO-1 Hyperion Cuprite y AVIRIS Jasper Ridge.

	EO-1 Hyperion Cuprite	**AVIRIS Jasper Ridge**
Algoritmo	**Tiempo de cálculo** (segundos)	**Tiempo de cálculo** (segundos)
Implementación PPI en C	41540	7873
Implementación en FPGA (Virtex-II Pro XC2VP30)	422.84	80.14
Implementación en FPGA (Virtex-4 XC4VFX60)	228.28	43.26

ponibles respectivamente. Los resultados demuestran que este algoritmo es muy adecuado para su implementación en FPGA dado que consigue *speedups* de entre 50 y 175. También es interesante resaltar que la implementación que proponemos obtiene un *speedup* de 3.5 con respecto a [VPVRP05] a pesar de utilizar una FPGA más pequeña.

Por último, la Tabla 4.7 da información de los tiempos de procesamiento de la implementación en FPGA considerada y la versión software equivalente desarrollada en lenguaje C y ejecutada en un PC con un procesador AMD Athlon 2.6 GHz y 512 Mb de RAM, para las escenas Hyperion EO-1 Cuprite y AVIRS Jasper Ridge. Ya que los tiempos de procesamiento de los datos AVIRIS Jasper Ridge en unidades de radiancia y reflectancia son exactamente iguales, solo se muestra uno de ellos en la tabla.

Para concluir esta sección, la Figura 4.13 muestra el tiempo de ejecución a medida que aumentamos el número de unidades dp en paralelo para calcular un número fijo de proyecciones (10^4) en la escena AVIRIS Cuprite. Debemos tener en cuenta que este comportamiento depende del número de veces que tenemos que procesar la imagen completa y por lo tanto no siempre se calculan 10^4 proyecciones. Por ejemplo, si tenemos 90 unidades dp en paralelo, necesitamos al menos 112 iteraciones del algoritmo para calcular 10^4 proyecciones o más, por lo que en realidad se calcularían $112 \times 90 = 10080$ proyecciones.

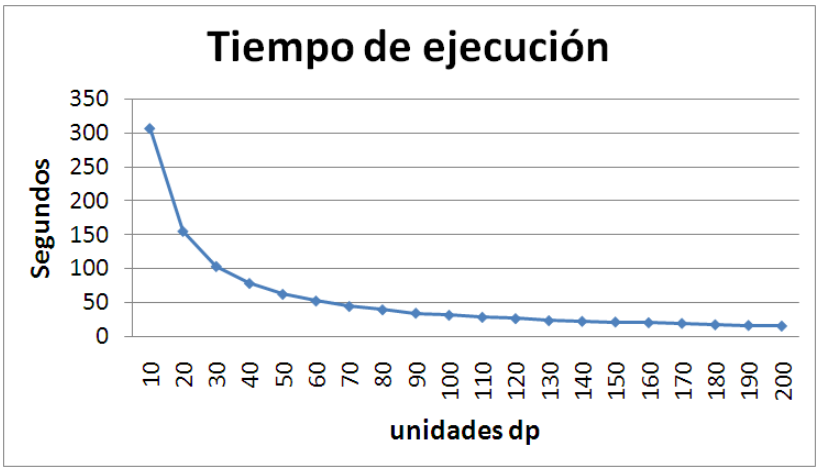


Figura 4.13: Tiempo de ejecución en segundos según el número de unidades dp en paralelo para calcular 10^4 proyecciones en la escena AVIRIS Cuprite.

4.4. Conclusiones

El procesado a bordo de imágenes hiperespectrales de la superficie terrestre ha sido un objetivo muy perseguido en teledetección. El número de aplicaciones que requieren una respuesta en tiempo real ha crecido exponencialmente en los últimos años. El diseño actual de sensores puede verse enormemente beneficiado con la incorporación de módulos especializados para el procesamiento, como las FPGAs, que pueden ser fácilmente empotradas en el sensor debido a su tamaño compacto. En este trabajo, se ha descrito una implementación en FPGA del algoritmo pixel purity index (PPI), una de las aproximaciones más conocidas para el análisis de datos hiperespectrales en teledetección. Los resultados experimentales, llevados a cabo en las FPGAs Xilinx Virtex-II Pro XC2V2P30 y Virtex-4 XC4VFX60 demuestran que la implementación hardware propuesta hace un uso apropiado de los recursos disponibles en la arquitectura considerada. Además, la versión hardware propuesta del algoritmo PPI supera de manera significativa (en términos de tiempo de cálculo) la versión original semisupervisada, disponible en software comercial, una aproximación completamente automática del algoritmo, y una implementación en FPGA recientemente desarrollada en una Xilinx Virtex-II XC2V6000-6. Otra característica interesante de la implementación propuesta es que es fácilmente escalable para FPGAs mayores.

Por ejemplo, si dispusiéramos de una FPGA Virtex-4 XQR4VLX200 (89088 slices) certificada para el espacio, podríamos tener 3.5 veces más unidades dot–product en paralelo simplemente sintetizando un array sistólico y un generador aleatorio para el nuevo número de unidades en paralelo. De esta manera, por ejemplo, el tiempo de cómputo estaría un poco por debajo de los 5 segundos para la escena AVIRIS Cuprite.

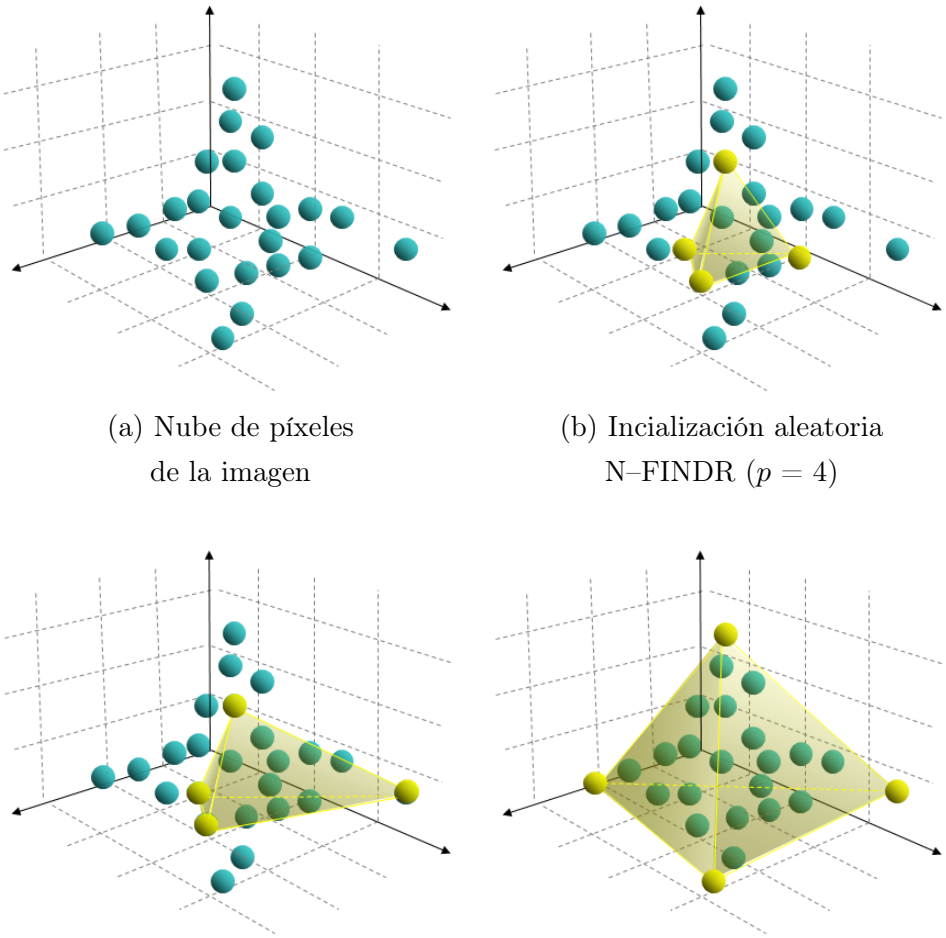
Capítulo 5

Algoritmo N-finder (N–FINDR)

Una búsqueda comienza siempre con la suerte del principiante y termina con la prueba del conquistador.

Paulo Coelho

En la última década, se han desarrollado varios algoritmos para la extracción automática o semiautomática de endmembers espectrales [PMPP04]. Los algoritmos más conocidos son el algoritmo PPI (que se acaba de describir en el capítulo anterior) y el algoritmo N–FINDR (que se desarrollará a lo largo del presente capítulo). El algoritmo N–FINDR desarrollado por Winter [Win03], es una de las técnicas más exitosas aplicadas para determinar de forma automática endmembers en datos de imágenes hiperespectrales. Este algoritmo utiliza una técnica basada en identificar los endmembers como los vértices del simplex de mayor volumen que puede formarse en la nube de puntos dada por todos los píxeles de la imagen hiperespectral (ver Figura 5.1(a)). Para ello, busca un conjunto de píxeles que formen el mayor volumen posible dentro de los datos mediante el "inflado" de un simplex. El procedimiento comienza con una selección aleatoria inicial de píxeles (ver Figura 5.1(b)). Posteriormente, se evalúan cada uno de los píxeles de la imagen con el fin de perfeccionar la estimación de endmembers, buscando el conjunto de



(a) Nube de píxeles de la imagen

(b) Incialización aleatoria N–FINDR ($p = 4$)

(c) Reemplazo de un endmember en N-FINDR

(d) Volumen final estimado por N–FINDR

Figura 5.1: Interpretación gráfica del algoritmo N–FINDR en un espacio tridimensional.

píxeles que maximice el volumen del simplex definido por los endmembers seleccionados. Se calcula el volumen correspondiente para cada píxel en cada una de las posiciones de los endmembers reemplazando dicho endmember y buscando el volumen resultante. Si el reemplazo resulta en un incremento del volumen, el píxel reemplaza al endmember (ver Figura 5.1(c)). Este procedimiento se repite hasta que no haya más reemplazos de endmembers (ver Figura 5.1(d)).

La definición matemática del volumen de un simplex, formado por un conjunto de endmembers candidatos, es proporcional al determinante del conjunto mencionado con una fila de unos. El determinante está definido solamente en el caso donde el número de características es $p - 1$, siendo p el número deseado de endmembers [Cha07]. Dado que en las imágenes hiperespectrales es muy común que el número de bandas espectrales sea mucho mayor que el número de endmembers, es decir $n >> p$, es necesaria una transformación que reduzca la dimensionalidad de los datos de entrada. Mientras que la fase de determinación de endmembers de N–FINDR en la versión comercial distribuida por la compañía Pacific Spectral Technology[1] ha sido optimizada para el procesamiento de alta velocidad, el rendimiento computacional del algoritmo depende de la exactitud de la primera selección aleatoria de endmembers y, sobre todo, de las dimensiones de la escena hiperespectral y del número de endmembers a buscar.

En este capítulo, se presenta un diseño para la implementación en FPGA del algoritmo N–FINDR. Se ha escogido la FPGA Virtex–4 XC4VFX60 de Xilinx como arquitectura representativa para demostrar el rendimiento de la implementación propuesta. Dicha implementación ha sido validada utilizando imágenes hiperespectrales reales captadas por el Airborne Visible Infra–Red Imaging Spectrometer (AVIRIS) de la NASA sobre la región minera de Cuprite en Nevada y sobre la reserva biológica de Jasper Ridge en California, y la captada por el satélite EO–1 Hyperion a lo largo del mismo distrito minero de Cuprite, al igual que la citada escena AVIRIS. Los resultados experimentales revelan que el sistema hardware propuesto es fácilmente escalable y es capaz de proporcionar resultados precisos con un tamaño compacto alcanzando un gran rendimiento, lo que hace que este sistema sea atractivo para el procesamiento a bordo de imágenes hiperespectrales.

5.1. Descripción del algoritmo

A continuación, se ofrece una descripción detallada paso a paso del algoritmo original N–FINDR desarrollado por Winter [Win03]. Cabe destacar que el algoritmo descrito a continuación representa nuestro propio esfuerzo

[1] http://www.pacificspectral.com

para determinar los pasos empleados por N–FINDR utilizando las referencias disponibles en la literatura [Win03, Win04]. Sin embargo, también es digno de mención que el algoritmo N–FINDR nunca ha sido revelado plenamente. Como resultado, esta descripción se ha desarrollado en base a los limitados resultados publicados disponibles y nuestra propia interpretación. Sin embargo, el algoritmo descrito a continuación ha sido verificado mediante el software comercial N–FINDR, proporcionado por los autores, donde hemos comprobado experimentalmente que el software produce esencialmente los mismos resultados que nuestro código, siempre y cuando los endmembers iniciales se generen aleatoriamente. El algoritmo original N–FINDR puede resumirse en los siguientes pasos:

1. *Reducción.* Aplicar una transformación de reducción dimensional, como la Fracción de Mínimo Ruido (FMR) [GBSC88] o el Análisis de Componentes Principales (ACP) [Sch97b], para reducir la dimensionalidad de los datos de n a $p-1$, donde p es un parámetro de entrada al algoritmo (número de endmembers a extraer).

2. *Inicialización.* Sea $\{\mathbf{E}_1^{(0)}, \mathbf{E}_2^{(0)}, \cdots, \mathbf{E}_p^{(0)}\}$ un conjunto de endmembers extraídos aleatoriamente de la entrada de datos.

3. *Cálculo del volumen.* En la iteración $k \geq 0$, calcular el volumen definido por el conjunto actual de endmembers de la siguiente manera:

$$V(\mathbf{E}_1^{(k)}, \mathbf{E}_2^{(k)}, \cdots, \mathbf{E}_p^{(k)}) = \frac{\left| det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{E}_1^{(k)} & \mathbf{E}_2^{(k)} & \cdots & \mathbf{E}_p^{(k)} \end{bmatrix} \right|}{(p-1)!} \tag{5.1}$$

4. *Reemplazo.* Para cada píxel $\mathbf{X}(i,j)$ en la entrada de datos, se recalcula el volumen evaluando el píxel en todas las p posiciones de los endmembers, es decir, primero se calcula $V(\mathbf{X}(i,j), \mathbf{E}_2^{(k)}, \cdots, \mathbf{E}_p^{(k)})$, después $V(\mathbf{E}_1^{(k)}, \mathbf{X}(i,j), \cdots, \mathbf{E}_p^{(k)})$, y así hasta $V(\mathbf{E}_1^{(k)}, \mathbf{E}_2^{(k)}, \cdots, \mathbf{X}(i,j))$. Si ninguno de los p volúmenes recalculados es mayor que $V(\mathbf{E}_1^{(k)}, \mathbf{E}_2^{(k)}, \cdots, \mathbf{E}_p^{(k)})$, entonces no se reemplaza ningún endmember. En caso contrario, se mantiene la combinación con el máximo volumen. Supongamos que el endmember ausente en la combinación resultante con el

máximo volumen se denota por $\mathbf{E}_j^{(k+1)}$. En este caso, se produce un nuevo conjunto de endmembers dejando $\mathbf{E}_j^{(k+1)} = \mathbf{X}(i, j)$ y $\mathbf{E}_i^{(k+1)} = \mathbf{E}_i^{(k)}$ para todo i $\neq$ j. El paso de *reemplazo* se repite de forma iterativa, utilizando tantas iteraciones como sean necesarias hasta que no haya más sustituciones de endmembers.

Hasta donde alcanza nuestro conocimiento, a pesar de la importancia del algoritmo N–FINDR en la comunidad de desmezclado hiperespectral, no hay implementaciones disponibles en hardware reconfigurable de este algoritmo en la literatura. En una sección próxima se proporciona una implementación hardware de este algoritmo que muestra la utilidad de este tipo de tecnología.

5.2. Reducción dimensional

El hecho de utilizar técnicas de preprocesado de imágenes hiperespectrales orientadas a la reducción de la dimensionalidad de los datos de entrada viene propiciado, entre otros motivos, por el conocido como fenómeno de Hughes [Hug68], que se describe a continuación. En un problema de clasificación típico, el objetivo es asignar una etiqueta de clase a los datos de entrada. El error mínimo esperado que se puede alcanzar al realizar la clasificación es lo que se conoce como el error de Bayes [Fuk90]. El error de Bayes es una función que decrece con la dimensionalidad de los datos. Una nueva característica añade información sobre el ejemplo y entonces, se esperaría que la clasificación fuese tan buena como cuando esta información no se había introducido. Sin embargo, en la práctica esto no es así, cuando se añade una nueva característica a los datos el error de Bayes disminuye, pero al mismo tiempo las desviaciones del error de la clasificación aumentan. Este incremento se debe al hecho de que se necesitan calcular más parámetros partiendo del mismo número de ejemplos. Si el incremento de las desviaciones en la clasificación del error es mayor que el decremento del error de Bayes, entonces el uso de la característica adicional degrada la regla de decisión. Y este fenómeno es lo que se conoce como el efecto Hughes [Hug68]. Además, cuando la dimensionalidad de los datos y la complejidad de la regla de decisión aumentan, el efecto Hughes puede llegar a ser más grave [Lan02].

En resumen, el rendimiento de un clasificador supervisado decrece con la dimensionalidad de los datos a menos que el número de muestras sea infinito [Hug68]. La reducción dimensional que se plantea es una etapa previa utilizada con objeto de reducir la carga computacional de pasos sucesivos mediante la eliminación de ruido e información redundante en la imagen. Estos métodos realizan una disminución del número de bandas, cuyo objetivo es obtener una representación mínima de la imagen que contenga la información indispensable para realizar el análisis sobre un subconjunto reducido de la imagen original [KZKP00]. Por otra parte, las técnicas de reducción dimensional suelen traer como consecuencia una mejora de la relación SNR en los datos a través de la eliminación de ruido [CDSA99], lo cual hace atractiva su utilización de forma previa al paso de clasificación. El inconveniente que presenta esta alternativa es la dificultad para interpretar los datos espectrales tras la etapa de reducción.

Es importante distinguir las técnicas de reducción dimensional de las técnicas de compresión de imágenes hiperespectrales [QHWM00]. Contrariamente al objetivo de los métodos de compresión, el proceso de simplificación dimensional no permite, por regla general, reconstruir la imagen original. Al contrario, el objetivo de la reducción dimensional es obtener una representación mínima de la imagen que contenga la información indispensable para realizar el análisis sobre un subconjunto reducido de la imagen original. De este modo, los algoritmos de reducción dimensional suelen estar diseñados de forma que minimizan los errores cometidos al trabajar con dicho subconjunto, despreocupándose de la posibilidad de recuperar la imagen original [PC07].

5.2.1. Análisis de componentes principales

El Análisis de Componentes Principales (ACP) utiliza una transformación lineal para maximizar la varianza de los datos. Los componentes principales se utilizan para producir nuevas bandas no correlativas, para eliminar los componentes de ruido y para reducir la dimensionalidad de los datos [Ste86].

El ACP es una transformación del espacio vectorial, normalmente utili-

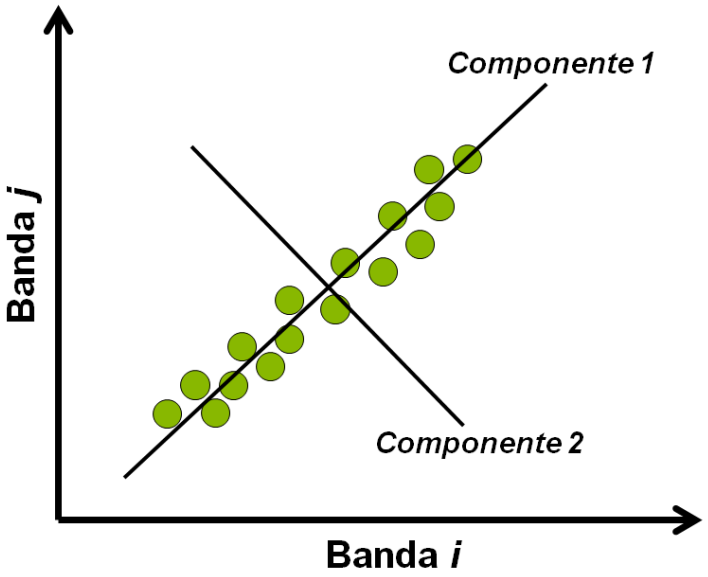


Figura 5.2: Ilustración gráfica de la transformación ACP.

zado como paso previo para reducir la alta dimensionalidad de los datos a una menor dimensionalidad para su posterior análisis.

El ACP construye una transformación lineal que busca un nuevo sistema de coordenadas para el conjunto original de datos en el cual la mayor varianza del conjunto de datos sea capturada en el primer eje (llamado el Primer Componente Principal), la segunda mayor varianza sea el segundo eje, y así sucesivamente, (ver Figura 5.2). Para obtener esta transformación lineal debe construirse primero la matriz de covarianzas o matriz de coeficientes de correlación. Debido a la simetría de esta matriz existe una base completa de vectores propios de la misma. La transformación que lleva de las antiguas coordenadas a las coordenadas de la nueva base es precisamente la transformación lineal necesaria para reducir la dimensionalidad de los datos. Además las coordenadas en la nueva base dan la composición en factores subyacentes de los datos iniciales.

El ACP es el análisis de valores multivaluados basados en autovectores más simple. Se usa para reducir la dimensionalidad de los datos reteniendo aquellas características del conjunto de datos que contribuyen más a su varianza, y manteniendo los componentes principales de orden más bajo e

ignorando los más altos. Ya que los componentes de orden bajo contienen los datos "más importantes". Sin embargo, dependiendo de la aplicación este podría no ser siempre el caso.

Supongamos que existe una muestra con n individuos para cada uno de los cuales se han medido m variables. El ACP permite encontrar un número de factores subyacentes $p < m$ que explican aproximadamente el valor de las m variables para cada individuo. El hecho de que existan estos p factores subyacentes puede interpretarse como una reducción de la dimensionalidad de los datos: donde antes necesitábamos m valores para caracterizar a cada individuo ahora nos bastan p valores. Cada uno de los p encontrados se llama Componente Principal, de ahí el nombre del método.

En este trabajo, el paso previo de reducción dimensional se ha llevado a cabo mediante el ACP del software ENVI 4.0 generando así, las imágenes reducidas dimensionalmente que serán la entrada del algoritmo N-FINDR.

5.3. Implementación en FPGA

Esta sección se organiza de la siguiente manera. En el apartado 5.3.1 se ofrece un resumen de las propiedades de los determinantes que se utilizarán para el diseño de la versión hardware propuesta del algoritmo N–FINDR para FPGA. El apartado 5.3.2 describe dicha implementación, explicando el funcionamiento de cada módulo hardware diseñado. Por último, el apartado 5.3.3 describe uno de estos módulos (el módulo hardware N–FINDR) en términos del lenguaje de descripción hardware.

5.3.1. Uso de las propiedades de los determinantes

La parte que más tiempo consume del algoritmo N–FINDR es el paso de *cálculo del volumen*. Los limitados recursos disponibles en una FPGA pequeña o mediana para calcular determinantes de gran orden (multiplicadores empotrados, look-up tables y slices), complican el desarrollo de una implementación hardware eficiente del algoritmo. El cálculo de un determinante es una sucesión de sumas y restas de los productos de los elementos de la diagonal principal para todas las permutaciones de las columnas de la ma-

triz. Para hallar un determinante de orden p mayor que 3 no es aconsejable recurrir al desarrollo de este como suma de $n!$ términos, cada uno de los cuales es un producto de n elementos. Esta vía es, salvo en casos triviales, excesivamente larga y requiere un gran número de operaciones. Los desarrollos de un determinante (desarrollo por elementos de una fila o columna), tampoco suelen proporcionar métodos numéricos que resuelvan el problema de modo eficaz. Lo aconsejable para calcular determinantes es hacer uso de las propiedades fundamentales de los determinantes, y aplicarlas con método y acertadamente para ir transformando el determinante en otros que sean cada vez más fáciles de calcular, hasta llegar a uno que se halle trivialmente (por ejemplo, el determinante de una matriz triangular). Nótese que, según se acaba de indicar, el método que aquí se propone, para hallar el determinante de $\mathbf{A}$, recurre a realizar adecuadas operaciones elementales en las filas y en las columnas de la matriz $\mathbf{A}$. En este trabajo, se utiliza el método de triangulación de matrices que nos permite calcular determinantes de orden suficiente para extraer los endmembers necesarios en la mayor parte de las imágenes hiperespectrales utilizando una FPGA pequeña.

A continuación, se ofrece una descripción del método de triangulación de matrices adoptado en este trabajo (ver Algoritmo 3 para más detalles). Para hallar el determinante de una matriz cuadrada $\mathbf{A}$, de tamaño $p \times p$, hágase lo que sigue:

1. Si el primer elemento de la matriz $a_{11} \neq 0$, no hay aquí, en (1), nada que hacer y se pasará directamente al paso (2). Si $a_{11} = 0$ y $a_{l1} \neq 0$ para algún l, permútense las filas 1^a y l-ésima, con lo que el determinante cambia de signo y tiene el elemento de lugar $(1, 1)$ distinto de cero. Si todas las a_{l1} son nulas, el problema ya está resuelto pues el determinante de $\mathbf{A}$ es cero, y no hay nada más que hacer.

2. Se puede ya suponer que queremos hallar el determinante de orden p (denotado por Δ_p) de la matriz $\mathbf{A}$, cuyo elemento a_{11} es no nulo. Para simplificar la notación, a los elementos de la primera columna de $\mathbf{A}$ los llamaremos $[\alpha_1, \alpha_2, \cdots, \alpha_p]^T$, siendo $\alpha_1 \neq 0$. Restémosle a la fila l-ésima lo que resulta de multiplicar la primera fila por α_l/α_1 y hagamos esto para $l = 2, 3, \cdots, p$; con ello, no se ha alterado el valor

Algoritmo 3 Método de triangulación de matrices aplicado a la matriz cuadrada $\mathbf{A}$ de tamaño p. En esta descripción algorítmica, por simplicidad y siguiendo la notación estándar del texto, se asume que $\mathbf{A}[i][j] \equiv a_{ij}$.

```
for (i = 0; i < p; i++) { //p denota el tamaño de la matriz cuadrada A
  //Paso 1
  if (A[i][i] == 0) {
    for (j = i + 1; j < p; j++) {
      if (A[j][j] != 0) {
        aux_row = A[i]; //A[i] denota la fila i-ésima completa
        A[i] = A[j];
        A[j] = aux_row;
        break;
      } end if
    } end for
  } end if
  if (A[i][i] == 0) return 0;
  //Paso 2
  for (j = i + 1; j < p; j++) {
    A[j] = A[j] - A[i]*(A[j][i]/A[i][i]);
  } end for
} end for
```

del determinante, que sigue valiendo Δ_p, y los elementos de la primera columna pasan a ser $[\alpha_1, 0, \cdots, 0]^T$. El determinante Δ_p se puede, pues, poner en la forma $\Delta_p = \alpha_1 \Delta_{p-1}$ donde Δ_{p-1} es el determinante, de orden $p-1$, que resulta de suprimir la primera fila y la primera columna de Δ_p.

3. Con el determinante Δ_{p-1} se repite el proceso anterior y, así, se le reduce a un determinante Δ_{p-2}, de orden $p-2$. Reiterando este proceso, se llega a un determinante de 3^{er} o de 2^o orden, que se calcula trivialmente.

En el diseño propuesto se itera hasta obtener una matriz triangular, y posteriormente se multiplican los elementos de la diagonal principal. Por otro lado, se utiliza la propiedad de que el determinante de $\mathbf{A}$ es igual al

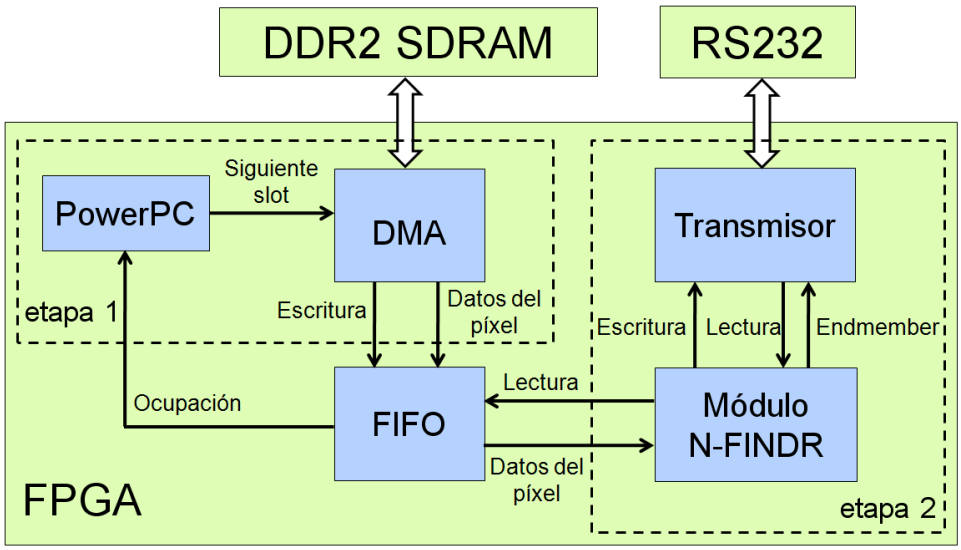


Figura 5.3: Arquitectura hardware para implementar el sistema N–FINDR completo.

determinante de $\mathbf{A}^T$ y se tiene en cuenta que el cambio de signo del determinante cuando intercambiamos dos filas no afecta a nuestro cálculo porque se está buscando el valor absoluto. Con estas elecciones de diseño en mente, en el apartado siguiente se verá que el diseño FPGA propuesto es capaz de extraer hasta $p = 21$ endmembers utilizando una FPGA pequeña como la Virtex-4 XC4VFX60 de Xilinx. Si el número necesario de endmembers es menor que esta cifra, se pueden ejecutar en paralelo en la FPGA el cálculo de los determinantes, reduciendo así el tiempo de ejecución.

5.3.2. Implementación hardware

La Figura 5.3 muestra la arquitectura hardware utilizada para implementar el algoritmo N–FINDR, junto con las comunicaciones de E/S. Nuevamente, para la entrada de datos se utiliza una memoria DDR2 SDRAM y un DMA (controlado por el PowerPC) con una FIFO para almacenar los datos de los píxeles. El módulo N–FINDR se emplea para implementar la versión propuesta del algoritmo N–FINDR. Finalmente, se utiliza un transmisor para enviar los endmembers a través del puerto RS232. La estructura general del hardware empleado para implementar el algoritmo N–FINDR

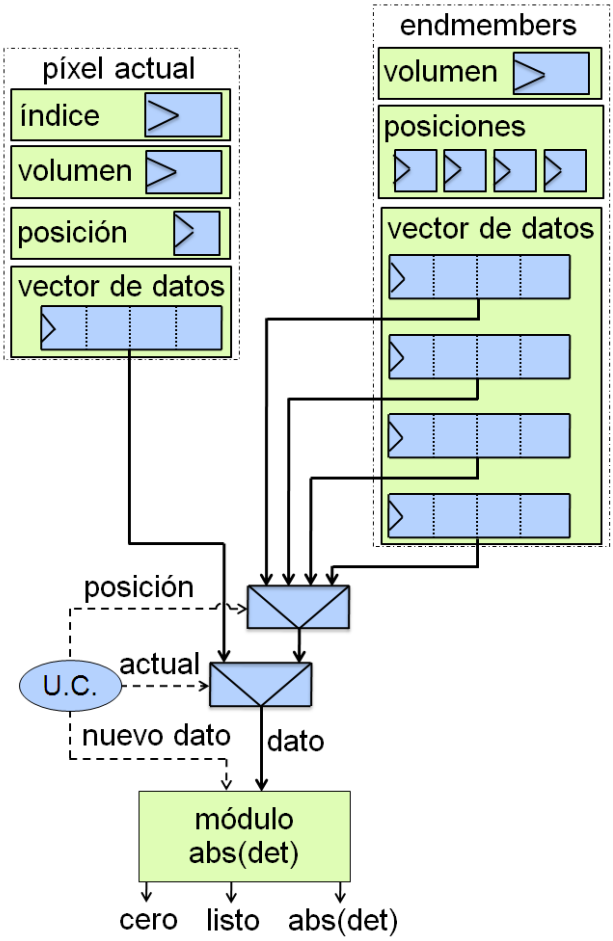


Figura 5.4: Arquitectura hardware para implementar el algoritmo N–FINDR.

puede ser visto como una arquitectura segmentada. Se pueden distinguir dos etapas que se comunican utilizando una estructura first-in first-out (FIFO): La primera etapa trae de memoria los píxeles de la imagen a la FIFO mientras la segunda etapa lleva a cabo el proceso de extracción de endmembers con los píxeles almacenados anteriormente y finalmente envía los endmembers a través del puerto RS232. Por lo tanto, las dos etapas están trabajando en paralelo.

Por otro lado, la Figura 5.4 muestra la arquitectura hardware utilizada para implementar el algoritmo N–FINDR. Se utilizan registros para almacenar los datos de los píxeles seleccionados como endmembers hasta el momento, sus posiciones en la imagen y su volumen, los valores del píxel actual, su

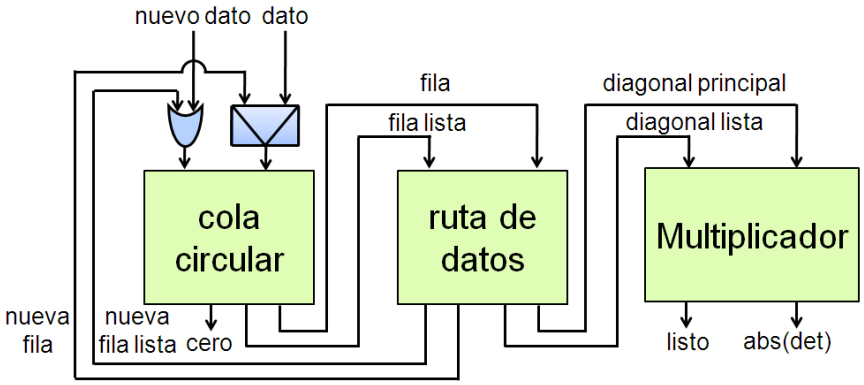


Figura 5.5: Arquitectura hardware para implementar el valor absoluto de un determinante(*módulo abs(det)* de la Figura 5.4).

posición, el mayor volumen y la posición del endmember donde se obtiene, y finalmente el módulo para calcular el valor absoluto del determinante. Para ser capaz de proporcionar un nuevo píxel (una fila de la matriz) en cada ciclo de reloj, se emplean registros para almacenar el conjunto actual de endmembers y un circuito combinacional consistente en multiplexores, donde todos estos módulos están manejados por la unidad de control. Por simplicidad, el hardware necesario para realizar el paso de reemplazo no se muestra en la Figura 5.4. Para implementar este paso se utiliza un comparador entre los registros de volúmenes y, si con el píxel actual el volumen es mayor, el reemplazo es llevado a cabo por la unidad de control.

La Figura 5.5 describe la arquitectura del módulo utilizado para calcular el valor absoluto del determinante utilizando el método de triangulación de matrices. Se pueden distinguir claramente tres unidades: La *cola circular*, con una pequeña unidad de control, es la responsable de proporcionar las diferentes filas a la *ruta de datos* durante el proceso de triangulación de la matriz e intercambiar las nuevas filas provenientes de la *ruta de datos* con el fin de tener el elemento $a_{ii} \neq 0$ (paso 1 del algoritmo). La *ruta de datos* recibe las últimas m filas de la matriz $(p \times p)$, almacena la primera y lleva a cabo la reducción a cero de los elementos de lugar $m-p+1$ en el resto de filas mediante la sustracción de la fila almacenada multiplicada por el resultado de la división entre los elementos $m-p+1$ de la fila actual y de la fila almacenada (paso 2 del algoritmo). Finalmente, el *multiplicador* calcula la multiplicación

de los elementos de la diagonal principal de la matriz triangular y obtiene el valor absoluto. A continuación, se describe el comportamiento de estos módulos en mayor detalle.

Primero, para $j = 2, \cdots, p$ hacemos un múltiplo a_{j1}/a_{11} de la primera fila y se la restamos a la fila j-esima, para hacer $a_{j1} = 0$. De esta manera, se han eliminado todos los elementos de $\mathbf{A}$ por debajo del "pivote" a_{11} en la primera columna. Ahora, para $j = 3, \cdots, p$ hacemos un múltiplo a_{j2}/a_{22} de la segunda fila y se la restamos a la fila j-ésima. Cuando se ha terminado esta operación, todos los elementos por debajo de la diagonal principal en la segunda columna son cero, y estamos listos para procesar la tercera columna. Aplicando este procedimiento a las columnas $i = 1, \cdots, p-1$ (no hay elementos que eliminar por debajo de la diagonal principal en la columna p-ésima) se completa el proceso de triangulación de la matriz y $\mathbf{A}$ se ha reducido a la forma de una matriz triangular superior. Es importante señalar, que al reducir a cero los elementos por debajo de la diagonal principal columna a columna a partir de la primera columna, estos elementos permanecerán valiendo cero ya que, mientras estemos restando un múltiplo a_{ji}/a_{ii} de la fila i a la fila j $(j > i)$, para reducir a_{ji}, sólo estamos restando múltiplos de cero a cero en las columnas 1 a $i - 1$.

La Figura 5.6 muestra la arquitectura de la *ruta de datos* utilizada en el diseño propuesto para implementar el paso 2 del proceso de triangulación de matrices. Básicamente, la *ruta de datos* almacena la primera fila de cada iteración en un registro y sucesivamente (para cada una de las filas restantes) calcula a_{ji}/a_{ii}, multiplica el resultado por la fila almacenada y finalmente se la resta a la fila correspondiente. Además, almacena los elementos de la diagonal principal de la matriz triangular superior y avisa cuando se han obtenido todos los elementos.

Para que el diseño anterior funcione se debe garantizar que los valores de los pivotes sean distintos de cero. Evidentemente, si uno de los pivotes a_{ii} de la diagonal es cero, no se puede utilizar a_{ii} para reducir a cero los elementos por debajo suya; no se puede cambiar a_{ji} mediante la resta de ningún múltiplo de $a_{ii} = 0$ a éste. Se debe intercambiar la fila i con otra fila l por debajo suya (la permutación con una fila por encima destruiría alguno de los ceros introducidos anteriormente), que contenga un elemento

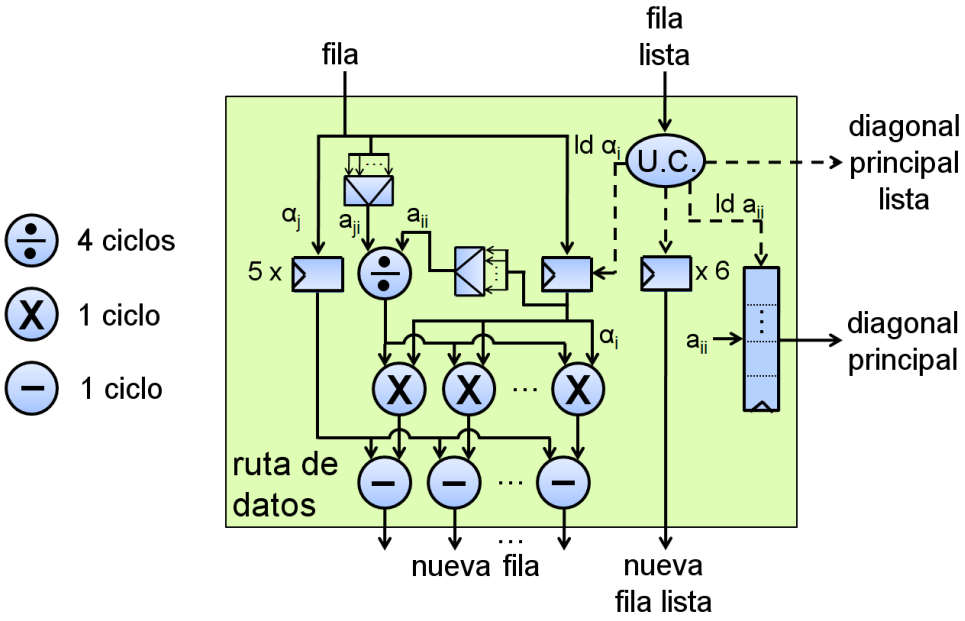
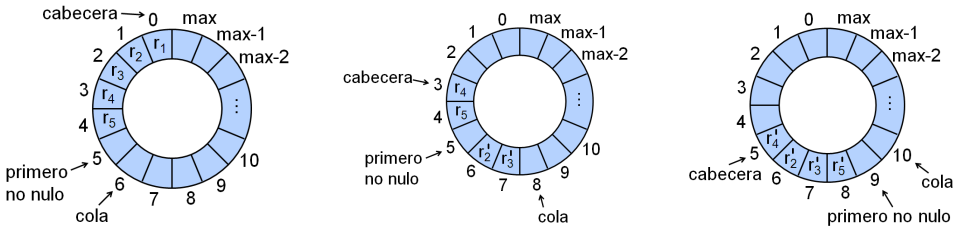


Figura 5.6: Arquitectura hardware para implementar el paso (2) del método de triangulación de matrices.

a_{li} no nulo en la columna i-ésima. Ahora el nuevo pivote a_{ii} no es nulo y puede continuar el proceso de triangulación de matrices. Si $a_{li} = 0$ para $l = i, \cdots, p$, entonces no será satisfactorio intercambiar la fila i con ninguna de las filas por debajo de ésta ya que todos los potenciales pivotes son cero y por lo tanto el determinante de $\mathbf{A}$ también será cero.

Para llevar a cabo el intercambio de filas se emplea una *cola circular* modificada. Además de utilizar un puntero para indicar el lugar del primer elemento (*cabecera*), se utiliza un segundo puntero para indicar la ubicación de la primera fila (con pivote distinto de cero) para la siguiente iteración (*primero no nulo*) y otro para indicar dónde se coloca la última fila por detrás del indicador anterior (*cola*). Una vez que las filas de la matriz son almacenadas, se envían sucesivamente a la *ruta de datos*. De esta manera, recibimos en orden las nuevas filas que se almacenarán en la posición indicada por el puntero *primero no nulo* o por el puntero *cola*, dependiendo de si previamente se ha detectado una fila con un pivote distinto de cero para la siguiente iteración. La correcta actualización de los punteros es llevaba a cabo por una pequeña unidad de control. En el caso de que el potencial



$$r_1 \begin{pmatrix} 1 & 2 & 5 & 1 & 2 \\ 1 & 2 & 6 & 2 & 3 \\ 1 & 2 & 7 & 4 & 3 \\ 1 & 3 & 9 & 2 & 4 \\ 1 & 4 & 5 & 6 & 6 \end{pmatrix} \quad r_1 \begin{pmatrix} 1 & 2 & 5 & 1 & 2 \\ & & & & \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 3 & 1 \\ & & & & \end{pmatrix} \quad r_1 \begin{pmatrix} 1 & 2 & 5 & 1 & 2 \\ 0 & 1 & 4 & 1 & 2 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 3 & 1 \\ 0 & 2 & 0 & 5 & 4 \end{pmatrix}$$

(a) Matriz inicial (b) Resultado parcial (c) Resultado final

Figura 5.7: Arquitectura hardware para implementar el paso (1) del proceso de triangulación de matrices.

pivote para la siguiente iteración sea cero en todas las filas, la señal *cero* se activa indicando que el valor del determinante es cero.

A continuación, se ofrece un ejemplo de cómo se realiza la operación mencionada. Después de almacenar las cinco filas de la matriz (ver Figura 5.7(a)), las filas r_1, r_2 y r_3 son enviadas y se reciben las nuevas filas $r_2^{'}$ y $r_3^{'}$ pero, en ambos casos, el elemento pivote para la siguiente iteración es cero, por lo que se han almacenado en la posición indicada por el puntero *cola* (ver Figura 5.7(b)). Después de enviar las otras filas, se obtiene la nueva fila $r_4^{'}$ cuyo pivote es distinto de cero y por ello ha sido almacenada en la posición indicada por el puntero *primero no nulo*. Por último, obtenemos la nueva fila $r_5^{'}$ que, con independencia del valor del pivote, se ha almacenado en la posición indicada por el puntero *cola* (ver Figura 5.7(c)). De esta manera, las filas están listas para la siguiente iteración.

Cuando la matriz se reduce a una forma triangular superior, el *multiplicador* calcula el producto de los elementos de la diagonal principal. Hoy en día, la gran mayoría de las FPGAs contienen multiplicadores hardware empotrados. Esto permite que la multiplicación pueda hacerse dentro de la FPGA sin el uso de un gran número de look-up tables y también con un

consumo bajo de potencia. Nuestro *multiplicador* es capaz de multiplicar n datos de entrada en $\log_2(n)$ ciclos de reloj utilizando n multiplicadores.

Para concluir esta sección, se ofrece una descripción paso a paso de cómo la arquitectura propuesta realiza la extracción de un conjunto de p endmembers a partir de una imagen hiperespectral:

- En primer lugar, el PowerPC selecciona aleatoriamente un conjunto inicial de p endmembers y envía una orden al DMA para escribirlos en la FIFO.

- Posteriormente, la unidad de control lee estos endmembers y los envía a los registros donde se almacenan. Entonces, se calcula el volumen del conjunto inicial de endmembers y el resultado se almacena en un registro.

- Después de que el PowerPC ha escrito el conjunto inicial de endmembers, envía una orden al DMA para comenzar a copiar un trozo de la imagen de la memoria DDR2 SDRAM a la FIFO. Como se mencionó anteriormente, el principal cuello de botella en este tipo de sistema es con frecuencia la entrada de datos que se aborda en nuestra aplicación con la incorporación de un DMA que elimina la mayor parte de las penalizaciones de E/S. Por otra parte, el PowerPC monitorea la FIFO de entrada y envía una nueva orden al DMA cada vez que se detecta que la FIFO de escritura está medio vacía. Esta vez, el DMA traerá un trozo de la imagen que ocupa la mitad de la capacidad total de la FIFO.

- Cuando los datos del primer píxel se han escrito en la FIFO, el módulo N–FINDR comienza a trabajar. Los datos del píxel son almacenados y en cada ciclo de reloj una nueva fila es enviada por la unidad de control al módulo encargado de calcular el valor absoluto del determinante. Cuando se calcula el volumen, se compara con el volumen almacenado y, si el primero es mayor, los registros de volumen y la posición se actualizan. Luego, se recalcula el volumen testeando el píxel en la siguiente posición de los endmembers de la misma manera. Este volumen se calcula para todas las p posiciones de los endmembers. Por

último, si el nuevo cálculo del mayor volumen con el píxel actual es mayor que el volumen anterior, la unidad de control actualiza el conjunto de endmembers, el volumen y los registros de posición. Este paso se repite varias veces en función del número total de píxeles de la imagen hiperespectral.

- Por último, el transmisor extrae los endmembers de los registros y los envía a través de un puerto RS232.

5.3.3. Módulo hardware N–FINDR

En este apartado, se ofrecen algunos detalles acerca del módulo hardware N–FINDR propuesto. Este módulo es el componente más importante del sistema y se tuvo especial cuidado en su definición utilizando VHDL como lenguaje de descripción hardware. Se siguió un estilo de descripción estructural que describe el módulo superior como un conjunto de instancias de diferentes componentes hardware (registros, ALU, multiplexores, ...) que coincide exactamente con la arquitectura representada en la Figura 5.4. Algunos de estos módulos son en realidad muy complejos en sí mismos y se vuelve a usar una descripción estructural para su definición. La única excepción es la unidad de control que se ha descrito como una máquina de estados.

Dado que los requisitos de cada imagen pueden ser muy diferentes (por ejemplo, el número óptimo de endmembers varía de una imagen a otra) es necesario un diseño flexible que se adapte a estos requisitos. Por esta razón se ha diseñado un módulo completamente personalizable utilizando varios parámetros "genéricos" que nos permiten crear una instancia del diseño de acuerdo a las características de la imagen hiperespectral considerada. Esta creación de instancias se hace sin la necesidad de hacer ningún cambio en el código. Con el fin de personalizar el diseño propuesto sólo tenemos que establecer el valor apropiado para cada parámetro "genérico". Estos parámetros son el número de endmembers a extraer, el número de píxeles en los datos hiperespectrales, el número de bits que representan la reflectancia en una banda y el número de bits de la posición que ocupa el píxel dentro de la imagen. Por ejemplo, para realizar la adaptación del diseño para un nú-

mero diferente de endmembers, es tan simple como ajustar el valor genérico correspondiente y sintetizar el módulo. De ahí que el diseño propuesto se pueda utilizar para cualquier número de endmembers siempre y cuando se disponga de suficientes recursos.

La Figura 5.8 describe el módulo hardware N–FINDR en VHDL siguiendo estas ideas. Para simplificar, sólo se han incluido las instancias de los componentes principales, y hemos eliminado la definición de los componentes y las señales, y algunos detalles de interconexión.

Como se puede observar en la Figura 5.8, todos los módulos utilizan uno o varios parámetros genéricos. Por lo tanto se pueden personalizar según las necesidades de las imágenes consideradas. Por otra parte, algunas de las instancias se encuentran dentro de una estructura "for generate" (justo después del primer *begin*) que se utiliza para crear instancias de un número variable de elementos de un determinado componente. En nuestro caso se utiliza para adaptar la ruta de datos al número de endmembers seleccionados, incluyendo registros suficientes para almacenar la información de cada endmember.

5.4. Resultados experimentales

5.4.1. Plataforma reconfigurable

La arquitectura hardware descrita en la sección 5.3.2 ha sido implementada utilizando VHDL para la especificación del módulo N–FINDR. Además, se han empleado los entornos Xilinx ISE y Embedded Developed Kit (EDK) para especificar el sistema completo. Este sistema ha sido implementado en la placa ML410 (ver Figura 4.10), una placa reconfigurable de bajo coste con una sola FPGA Virtex-4 XC4VFX60, una ranura DDR2 SDRAM que admite hasta 2 GBytes, un puerto RS232 y algunos componentes adicionales que no se han utilizado en la implementación propuesta.

5.4.2. Conjunto de imágenes hiperespectrales

Nuevamente, se vuelve a utilizar el mismo conjunto de imágenes hiperespectrales empleados en la evaluación del algoritmo PPI y descrito en detalle

```
entity NFINDRmodule is
   generic(endmembers : integer := 16;
           numBitsData : integer := 16;
           numPixels : integer := 350*350;
           numBitsPositions : integer := 17);
   port(clk, rst, newVectorData: in std_logic;
        din : in std_logic_vector(endmembers*numBitsData-1 downto 0);
        position : in std_logic_vector(numBitsPositions-1 downto 0 );
        ready : out std_logic;
        doutPositions : out std_logic_vector(endmembers*numBitsPositions-1 downto 0));
end NFINDRmodule;

architecture NFINDRmoduleArch of NFINDRmodule is

  -- Definición de componentes

  -- Definición de señales

begin

  endmembersPositionsAndData : for I in 0 to endmembers-1 generate

    data: register generic map (endmembers*numBitsData) port map (clk, rst,
    ldEndmembers(I), dinEndmemberData, doutData(I*numBitsData+numBitsData-1 downto
    I*numBitsData));

    positions: register generic map (endmembers*numBitsPositions) port map (clk, rst,
    ldEndmembers(I), position, doutPositions(I*numBitsPositions+numBitsPositions-1
    downto I*numBitsPositions));

  end generate;

  endmembersVolume : register generic map (numBitsData)
    port map (clk, rst, ldEndmembersVolume, absDet, doutEndmembersVolume);

  pixelIndex : register generic map (numBitsPositions)
    port map (clk, rst, ldPixelIndex, dinPixelIndex, doutPixelIndex);

  pixelVolume : register generic map (numBitsData)
    port map (clk, rst, ldPixelVolume, absDet, doutPixelVolume);

  pixelPosition : register generic map (numBitsPositions)
    port map (clk, rst, ldPixelPosition, position, doutPixelPosition);

  pixelData : register generic map (endmembers*numBitsData)
    port map (clk, rst, ldPixelData, din, doutPixelData);

  mux01 : mux generic map (endmembers*numBitsData, endmembers)
    port map (doutData, position, doutMux01);

  mux02 : mux generic map (endmembers*numBitsData, 2)
    port map (doutMux01&doutPixelData, current, data);

  absDetModule : generic map (endmembers, numBitsData, numBitsCounter)
    port map (newData, data, zero, ready, absDet);

  controlUnit : generic map (endmembers, numBitsData, numPixels, numBitsPositions)
    port map (newData, current, position, ready, ...); -- Resto de señales

end NFINDRmoduleArch;
```

Figura 5.8: Módulo N–FINDR descrito en términos del lenguaje de descripción hardware VHDL.

en la sección 4.3.2. Las imágenes hiperespectrales por lo tanto, corresponden con la conocida escena AVIRIS Cuprite (en unidades de reflectancia) sobre la región minera de Cuprite en Nevada, la escena AVIRIS Jasper Ridge (tanto en unidades de radiancia como de reflectancia) sobre la reserva biológica de Jasper Ridge en California y la escena EO–1 Hyperion (en unidades de radiancia) a lo largo del mismo distrito minero de Cuprite, al igual que la citada escena AVIRIS.

5.4.3. Evaluación de los endmembers

Antes de analizar el rendimiento de la implementación propuesta, se llevaron a cabo los experimentos para comprobar la precisión de los endmembers extraídos para apreciar la similitud espectral entre las firmas de referencia para cada una de las escenas y los correspondientes endmembers extraídos por la implementación del algoritmo N-FINDR. En primer lugar, hacemos hincapié en que la versión hardware propuesta obtiene exactamente los mismos resultados que una implementación software propia, que se ha validado con la aplicación comercial de N–FINDR con el fin de garantizar que las tres versiones ofrecen exactamente los mismos resultados en los escenarios considerados. Nuevamente, se han utilizado en los experimentos realizados tanto datos corregidos atmosféricamente como sin corregir, con el fin de evaluar el impacto de este proceso en los resultados finales de extracción de endmembers. Si nos dirigimos al análisis a bordo y en tiempo real de imágenes hiperespectrales es importante decidir si es necesario incluir un módulo de corrección atmosférica antes de extraer los endmembers ya que, puede tener un impacto importante en la cadena completa de procesamiento.

Para llevar a cabo la comparación cuantitativa en los experimentos realizados se vuelve a utilizar el ángulo espectral (AE) entre el endmember más similar detectado por la implementación propuesta y la firma espectral de referencia disponible en cada escena. Recordemos que el AE entre un píxel $\mathbf{X}(i,j)$ seleccionado por el algoritmo N–FINDR y una firma espectral de referencia $\mathbf{S}_i$ disponible *a priori*, puede calcularse simplemente como:

$$\mathrm{AE}\left[\mathbf{X}(i,j),\mathbf{S}_k\right] = \cos^{-1}\frac{\mathbf{X}(i,j)\cdot\mathbf{S}_k}{\|\mathbf{X}(i,j)\|\cdot\|\mathbf{S}_k\|}, \tag{5.2}$$

es decir, el AE mide el ángulo formado por dos vectores n-dimensionales. Como resultado, valores bajos de AE significan una alta similitud espectral entre los vectores comparados. Esta medida de similitud espectral permite reducir el impacto de las fuentes de interferencia atmosférica en la evaluación realizada. Esto nos puede ayudar a compensar las diferentes condiciones de adquisición para un píxel en la imagen original y para una firma espectral recogida sobre el terreno (como es el caso de las firmas de referencia del USGS utilizadas para la imagen AVIRIS Cuprite).

Nuevamente, debido a la falta de firmas espectrales de referencia de los datos de EO–1 Hyperion, se han llevado a cabo los experimentos sobre la precisión de los endmembers extraídos con las escenas AVIRIS Cuprite y AVIRIS Jasper Ridge con el fin de evaluar la similitud espectral entre los endmembers obtenidos por la implementación N–FINDR propuesta y las firmas de referencia disponibles para estas escenas, que comprenden un conjunto de firmas de la biblioteca espectral USGS en el caso de AVIRIS Cuprite (disponibles en unidades de reflectancia) y un conjunto de píxeles de la imagen etiquetados como espectralmente puros en el caso de AVIRIS Jasper Ridge (disponible tanto en unidades de radiancia como reflectancia). El único parámetro de entrada para el algoritmo N–FINDR es el número de endmembers a extraer. En los experimentos realizados, este número se establece en $p = 16$ para la escena AVIRIS Cuprite y $p = 19$ para las escenas AVIRIS Jasper Ridge, después de estimar la dimensionalidad de los datos mediante el concepto *dimensionalidad virtual* [DC04].

Las Tablas 5.1 y 5.2 muestran los valores de AE entre los endmembers detectados por la implementación propuesta basada en FPGA y las firmas espectrales de referencia disponibles para las escenas AVIRIS Cuprite y AVIRIS Jasper Ridge, respectivamente. Con el fin de mostrar los resultados de una manera más efectiva, las tablas sólo muestran la menor puntuación de AE de todos los endmembers extraídos con respecto a su firma de referencia en cada caso. Los resultados se muestran en forma de radianes y grados (el rango de valores para el AE es de $[0, 90]$ grados). Estos resultados son consistentes con los publicados anteriormente en la literatura (por ejemplo, en [PMPP04]). En [PMPP04] podemos encontrar una comparación de los endmembers extraídos por el algoritmo N–FINDR con respecto a los obtenidos

Tabla 5.1: Similaridad espectral angular entre los endmembers extraídos por N–FINDR en la escena AVIRIS Cuprite y las firmas disponibles en la librería espectral USGS.

	Radianes	Grados
Alunita	0.084	4.812
Buddingtonita	0.089	5.099
Calcita	0.105	6.016
Kaolinita	0.138	7.906
Moscovita	0.108	6.187

Tabla 5.2: Similaridad espectral angular entre los endmembers extraídos por N–FINDR en las escenas AVIRIS Jasper Ridge (en unidades de radiancia y reflectancia) y las firmas espectrales puras disponibles en ambas escenas.

	Datos en radiancia				
	Suelo	Bosque	Hierba	Chaparral	Lago
Radianes	0.077	0.065	0.044	0.050	0.032
Grados	4.411	3.724	2.521	2.864	1.833
	Datos en reflectancia				
	Suelo	Bosque	Hierba	Chaparral	Lago
Radianes	0.028	0.025	0.022	0.020	0.019
Grados	1.604	1.432	1.260	1.145	1.088

por otros algoritmos populares de extracción de endmembers. Tienen particular relevancia los resultados mostrados en la Tabla 5.1, que indican que el algoritmo N–FINDR puede obtener con éxito los endmembers espectrales de datos en unidades de radiancia y reflectancia. Esto apoya nuestra tesis de que el proceso de extracción endmember se puede realizar al mismo tiempo que los datos se recogen en el sensor, sin la necesidad de tener un módulo de corrección atmosférica anterior, que también debería ser implementado en hardware. Este tema será objeto de investigación en nuestro trabajo futuro.

5.4.4. Evaluación del rendimiento

En este apartado se lleva a cabo una evaluación experimental del rendimiento computacional de la implementación en FPGA propuesta. A título ilustrativo, la Tabla 5.3 muestra los recursos utilizados para la implementación hardware del diseño propuesto para el algoritmo N–FINDR para diferentes valores de p (número de endmembers a extraer), llevada a cabo en la FPGA Virtex-4 XC4VFX60 de la placa ML410. Esta FPGA tiene un total de 25280 slices, 50560 slice flip flops y 50560 LUTs de cuatro entradas disponibles. Además esta FPGA incluye algunos recursos heterogéneos, tales como dos PowerPCs, 128 DSP48Es y Block RAMs distribuidas. En la implementación propuesta se han aprovechado estos recursos para optimizar el diseño. Un PowerPC supervisa las comunicaciones y las Block RAMs se utilizan para implementar la FIFO, por lo que la gran mayoría de los slices se utilizan para la implementación del algoritmo N–FINDR junto con los multiplicadores DSP48Es.

Como muestra la Tabla 5.3, el porcentaje de utilización hardware aumenta cuando se incrementa el número de endmembers a identificar. Para el máximo valor utilizado en los experimentos ($p = 21$) el porcentaje total de utilización hardware es del 97,40 % alcanzando casi la total ocupación. Sin embargo, para otros valores de p probados en los experimentos, por ejemplo, $p = 19$, aún queda espacio en la FPGA para algoritmos adicionales. A su vez, los valores de p superiores a los utilizados en los experimentos, son bastante improbables como se demuestra en los experimentos realizados con diferentes escenas y a los resultados anteriores publicados en la literatura de extracción de endmembers (además, el número de endmembers es generalmente mucho menor que el número de bandas espectrales). Por lo tanto, creemos que la implementación hardware propuesta será capaz de hacer frente a muchos escenarios diferentes de análisis. En cualquier caso, este valor puede aumentarse considerablemente en las placas de FPGA más actuales, pero en los experimentos realizados se ha decidido informar sólo de los resultados con esta placa porque es muy similar a otra certificada para el espacio.

Otro aspecto importante de la implementación hardware propuesta son las comunicaciones, que a menudo son el principal cuello de botella de los

Tabla 5.3: Resumen de los recursos utilizados para la implementación en FPGA del algoritmo N–FINDR para distinto número de endmembers en la FPGA Virtex-4 XC4VFX60.

Componente	Número de endmembers (p)	Número de DSP48Es	Número de slice flip flops	Número de 4 input LUTs	Número de slices	Porcentaje total	Frecuencia máxima (MHz)
Módulo N–FINDR	9	92	6322	11031	6231	24.65	43.1
	16	128	12036	20779	11700	46.23	42.9
	18	128	15095	24247	14056	55.6	42.8
	19	128	16646	26763	17577	69.53	42.6
	21	128	20077	34155	24622	97.40	42.3
Transmisor RS232	-	0	69	128	71	0.28	208
Controlador DMA	-	0	170	531	367	1.45	102

sistemas paralelos. Por lo tanto, al igual que en el diseño del algoritmo PPI, se ha prestado especial atención a este problema. Para reducir las penalizaciones de E/S, se ha incluido un DMA y se ha aplicado una técnica de precarga con el fin de ocultar la latencia de las comunicaciones. Básicamente, mientras que el módulo N–FINDR está procesando un conjunto de datos, el DMA irá a buscar la siguiente serie, y la almacenará en la FIFO. Para demostrar las ventajas de utilizar un DMA, se ha desarrollado otra versión en la que los datos de la imagen se leen de memoria y son escritos en la FIFO por el PowerPC en lugar del DMA. En esta versión, el tiempo de procesamiento se incrementó en más de un orden de magnitud por lo que podemos concluir que los recursos utilizados para el DMA (véase la Tabla 5.3) están bien empleados.

Por último, la Tabla 5.4 da información de los tiempos de procesamiento medios de la implementación en FPGA considerada y una versión software equivalente desarrollada en lenguaje C y ejecutada en un PC con un procesador AMD Athlon 2.6 GHz y 512 Mb de RAM, en el conjunto hiperespectral de datos considerados. Ya que los tiempos de procesamiento de los datos AVIRIS Jasper Ridge en unidades de radiancia y reflectancia son exactamente iguales, solo se muestra uno de ellos en la tabla. En todos los casos, se reporta el valor de p (número de endmembers a extraer), que dependerá de la escena, del tamaño total de la imagen en megabytes, y el *speedup* de la implementación hardware con respecto a la versión software desarrollada.

Para concluir esta sección, destacamos que el informe de tiempos de procesamiento en FPGA está todavía lejos de alcanzar la velocidad de adquisición de imágenes de un sensor hiperespectral y poder así procesar los píxeles a medida que lleguen al sistema. Por ejemplo, el tiempo de exploración de la línea de escaneo en AVIRIS, un instrumento de barrido, es bastante rápido (8.3 ms para recoger 512 píxeles completos). Esto introduce la necesidad de procesar un conjunto de datos hiperespectrales con 614×512 píxeles (por ejemplo, las escenas AVIRIS Jasper Ridge) en 4.98 segundos para alcanzar el pleno rendimiento en tiempo real. En los experimentos realizados, los tiempos de procesamiento logrados en la placa FPGA considerada para el mismo volumen de datos son del orden de diez veces esa cifra. Aunque la inclusión de placas de FPGA más recientes podría mejorar significativamente el informe

Tabla 5.4: Tiempos de procesamiento medios para la implementación hardware y para la versión software equivalente de N–FINDR para las imágenes hiperespectrales consideradas.

	AVIRIS Cuprite	**EO-1 Hyperion**	**AVIRIS Jasper Ridge**
Número de endmembers (p)	16	21	19
Tamaño total (Megabytes)	50	800	140
Tiempo de la versión software (segundos)	502.06	9110.24	1851.34
Tiempo de la versión hardware (segundos)	13.46	239.11	49.35
Speedup	37.29	38.10	37.50

de tiempos de procesamiento, como trabajo futuro, también nos centraremos en la mejora de la implementación propuesta para lograr un mejor aprovechamiento de los recursos hardware y reducir los tiempos de procesamiento, que en cualquier caso, se consideran aceptables en muchas aplicaciones de observación remota de la Tierra.

5.5. Conclusiones

Uno de los retos más importantes que debe abordarse en el análisis de imágenes hiperespectrales es la complejidad computacional de los algoritmos derivada de la cada vez mayor resolución espacial, espectral y temporal de las nuevas misiones de observación remota hiperespectral de la Tierra, muchas de ellas basadas en plataformas espaciales. Con los recientes avances en hardware reconfigurable, especialmente en las FPGAs, los algoritmos de análisis de imágenes hiperespectrales se pueden ahora implementar en FPGAs de alto rendimiento para acelerar sus tiempos de respuesta. Una técnica importante para el análisis de datos hiperespectrales es el desmezclado espectral, en el cual la extracción endmembers es una tarea fundamental. En este tra-

bajo, se ha desarrollado la primera implementación en FPGA del algoritmo N–FINDR para la extracción de endmembers, que ilustra las ventajas y desventajas de la tecnología reconfigurable en el contexto de las misiones aéreas y espaciales de observación remota de la Tierra. Los resultados experimentales, realizados en una FPGA Virtex-4 XC4VFX60 demuestran que la implementación hardware propuesta puede superar de manera significativa (en términos de tiempo de cálculo) una versión de software equivalente y también es capaz de proporcionar resultados precisos con un tamaño compacto, que hacen del sistema reconfigurable propuesto atractivo para el procesamiento a bordo de datos hiperespectrales.

Capítulo 6

Image Space Reconstruction Algorithm (ISRA)

Dime con quién andas,
y te diré quién eres.

Refrán popular

Las técnicas de análisis hiperespectral desarrolladas en la literatura presuponen que la medición obtenida por el sensor en un determinado píxel viene dada por la contribución de diferentes materiales que residen a nivel subpíxel. El fenómeno de la mezcla puede venir ocasionado por una insuficiente resolución espacial del sensor, pero lo cierto es que este fenómeno ocurre de forma natural en el mundo real, incluso a niveles microscópicos, por lo que resulta imprescindible el diseño de técnicas capaces de modelarlo de manera adecuada. No obstante, las técnicas basadas en este modelo son altamente costosas desde el punto de vista computacional.

En este sentido, el modelo lineal de mezcla [HC00] expresa los píxeles mezcla como una combinación lineal de las firmas asociadas a los componentes espectralmente puros (llamados *endmembers*) en la imagen [PMPP04]. Este modelo ofrece resultados satisfactorios cuando los componentes que residen a nivel subpíxel aparecen espacialmente separados, situación en la que los fenómenos de absorción y reflexión de la radiación electromagnética inci-

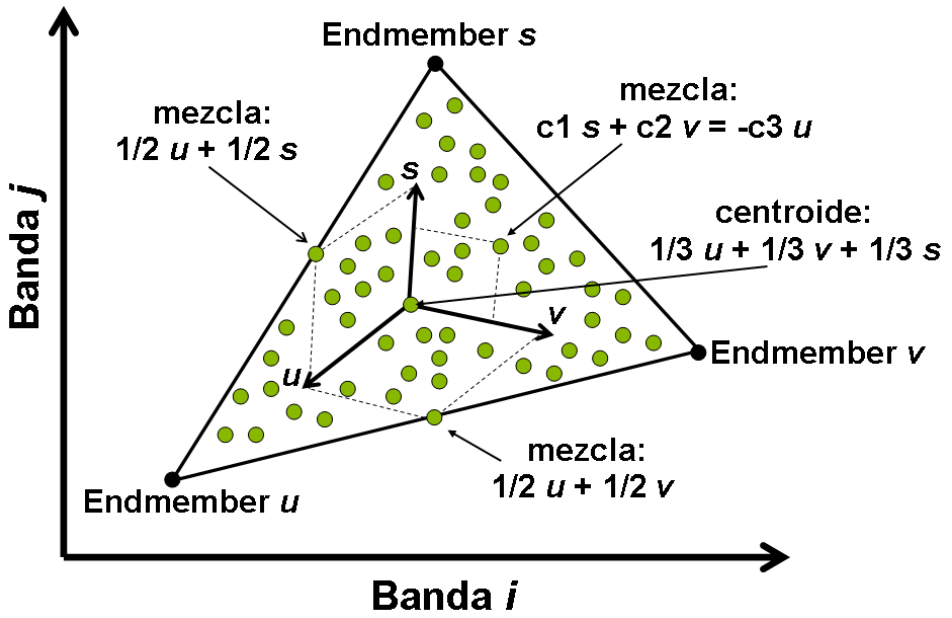


Figura 6.1: Interpretación gráfica del modelo lineal de mezcla.

dente pueden caracterizarse siguiendo un patrón estrictamente lineal. En la actualidad, el modelo lineal de mezcla es el más utilizado en análisis hiperespectral, debido a su sencillez y generalidad.

El modelo lineal de mezcla puede interpretarse de forma gráfica en un espacio bidimensional utilizando un diagrama de dispersión entre dos bandas poco correlacionadas de la imagen, tal y como se muestra en la Figura 6.1. En la misma, puede apreciarse que todos los puntos de la imagen quedan englobados dentro del triángulo formado por los tres puntos más extremos (elementos espectralmente más puros). Los vectores asociados a dichos puntos constituyen un nuevo sistema de coordenadas con origen en el centroide de la nube de puntos, de forma que cualquier punto de la imagen puede expresarse como combinación lineal de los puntos más extremos, siendo estos puntos los mejores candidatos para ser seleccionados como endmembers [BC96].

Es importante destacar que otra aproximación en la literatura al problema de la mezcla se basa en suponer que los endmembers interactúan según un modelo no lineal. Borel y Gerstl [BG94] demostraron que los efectos no

lineales que se producen en este caso se deben, fundamentalmente, a efectos de dispersión múltiple en la luz reflejada por los diferentes materiales [GHS02]. El modelo no lineal apareció ilustrado gráficamente mediante un ejemplo en la Figura 2.11(b). A pesar de que el modelo no lineal puede resultar más preciso en determinadas circunstancias, su correcta aplicación requiere información a priori acerca de la geometría y las propiedades físicas de los objetos observados, lo cual lo hace difícilmente abordable en situaciones en las que no existe dicha información. Además, el modelo no lineal es difícilmente generalizable a diferentes áreas de aplicación (por ejemplo, las propiedades físicas y la geometría de los minerales son completamente diferentes de las propiedades físicas y la geometría de las cubiertas vegetales, e incluso cada cubierta vegetal podría requerir un modelo específico). Por estos motivos, el modelo lineal es el más ampliamente utilizado en la literatura relacionada con el análisis de mezclas mediante endmembers.

En este capítulo, se proporciona una descripción detallada del Image Space Reconstruction Algorithm (ISRA), que sigue un modelo lineal de mezcla para la estimación de las fracciones de abundancia no negativas en píxeles mezcla, su implementación en FPGA y la validación de la arquitectura hardware paralela propuesta mediante los resultados experimentales obtenidos.

6.1. Descripción del algoritmo

Los sensores de imágenes hiperespectrales poseen una alta resolución espectral (del orden de cientos de bandas) pero una resolución espacial relativamente moderada. Los píxeles mezcla son una consecuencia de esta moderada resolución espacial de los sensores de imágenes hiperespectrales y por lo tanto, la firma espectral medida es una mezcla de las firmas de los objetos que caen dentro del campo de visión del sensor [VRJ00]. Además, los píxeles mezcla pueden ser el resultado de la combinación de diferentes materiales en una mezcla homogénea [KM02]. El desmezclado espectral es el procedimiento de descomponer el espectro medido de los píxeles mezcla en un conjunto de espectros puros, *endmembers*, y el conjunto correspondiente de fracciones de abundancia, *abundancias* [VRJ00, KM02]. Cuando no se tiene ningún conocimiento sobre los endmembers y las abundancias, el proceso de desmezclado

espectral se denomina *problema de desmezclado completo*. Cuando se conoce información a priori sobre los endmembers, el proceso se denomina *problema de estimación de abundancias*. En la literatura, se pueden encontrar diferentes aproximaciones para solucionar el problema de la mezcla espectral en los que la mayoría siguen un modelo lineal de mezcla [VRJ00, KM02, PMPP02], dado por:

$$\mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \cdot x_i + \mathbf{w} = \mathbf{A} \cdot x + \mathbf{w} \tag{6.1}$$

donde $\mathbf{A} \in \Re_+^{m \times n}$ es la matriz de endmembers y $\mathbf{a}_i$ es la firma espectral del i-ésimo endmember; $\mathbf{x} \in \Re_+^n$ es el vector de abundancias; $\mathbf{b} \in \Re^m$ es el espectro medido del píxel; $\mathbf{w}$ es un vector de ruido; n es el número de endmembers y m es el número de bandas espectrales del sensor [VRJ00, KM02, Boa94]. Tengamos en cuenta que los valores correspondientes a las variables $\mathbf{A}$ y $\mathbf{b}$ son obligatoriamente positivos para poder tener significado físico y, además, el vector de abundancias necesita satisfacer $\mathbf{x} \geq 0$ y $\sum_{i=1}^{n} x_i \leq 1$ o $\sum_{i=1}^{n} x_i = 1$. El modelo lineal de mezcla asume que la luz incidente interactúa en la superficie con un solo endmember (no hay dispersión múltiple entre endmembers), la superficie total es una combinación lineal de las abundancias de los endmembers como se mostró en la Figura 2.11(a) [VRJ00, KM02]. La Figura 6.2 muestra la mezcla resultante de los endmembers de la Figura 6.1 utilizando el modelo lineal de mezcla. En la literatura, la mayoría de los algoritmos desarrollados para el problema de desmezclado no estiman los endmembers y las abundancias simultáneamente. Primero estiman los endmembers siguiendo algunos de los métodos ya comentados [PMPP02] y posteriormente estiman las abundancias. El problema de estimación de abundancias puede ser visto como un problema de minimización de distancia dado por:

$$\begin{aligned} &\hat{\mathbf{x}} = \mathbf{arg\ min}_x\ \mathbf{D(Ax,b)} \\ &\mathbf{con}\ \mathbf{x} \geq 0\ y\ \sum_{i=1}^{n} x_i = 1 \end{aligned} \tag{6.2}$$

donde **D(Ax,b)** es la función de "distancia", $\mathbf{A}$ es la matriz de endmembers, $\mathbf{b}$ es el píxel en observación, $\mathbf{x}$ es el vector de abundancias y $\mathbf{arg\ min}_x\ \mathbf{f}$ devuelve la $\mathbf{x}$ que hace minimiza la función $\mathbf{f}$. Los diferentes algoritmos que podemos encontrar en la literatura no consideran ninguna o sólo algunas de

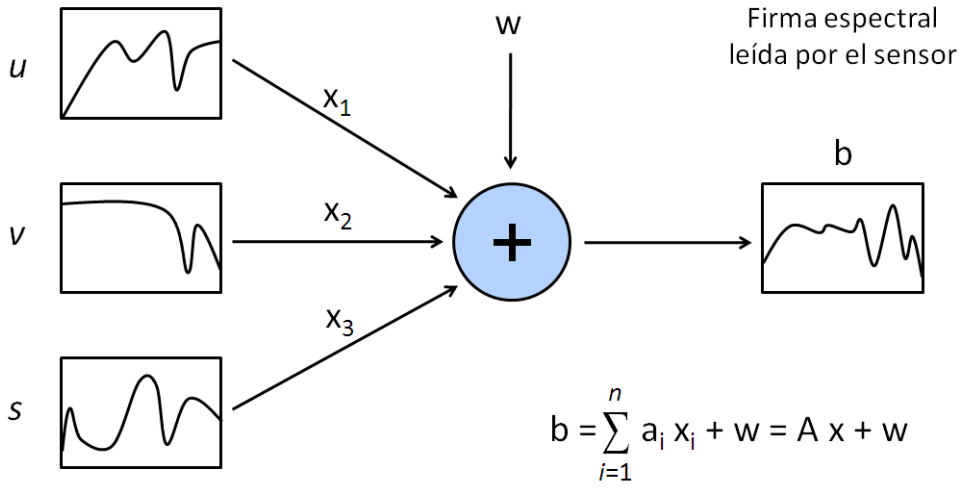


Figura 6.2: Composición de un píxel mezcla.

estas restricciones. Por otro lado, el uso de diferentes funciones de distancia da lugar a estimaciones diferentes. La distancia más común utilizada en la literatura es la de *mínimos cuadrados* (MC):

$$MC(Ax, b) = \|Ax - b\|^2 = \sqrt[2]{\sum_{i=1}^{n}((Ax)_i - b_i)^2} \tag{6.3}$$

Image Space Reconstruction Algorithm (ISRA) es uno de los métodos iterativos más conocidos utilizados para estimar la proporción de cada endmember (**A**) presente en el píxel (**b**) de una imagen hiperespectral. El problema de estimación de abundancias se aborda desde la perspectiva de un problema de minimización de distancia, utilizando la función de MC (ver Ecuación (6.3)), donde la diferencia entre el píxel o el espectro medido y estimado debe ser la más pequeña. ISRA es un ejemplo de algoritmo con restricciones solamente positivas. Esto significa que sólo se considera la restricción de fracciones de abundancias no negativas $\mathbf{x} \geq 0$. Este algoritmo garantiza valores positivos en el resultado de las abundancias y la convergencia asintótica del algoritmo [VRRPL03].

ISRA se utiliza en muchas aplicaciones, tales como la reconstrucción de la imagen en tomografía de emisión [DWM86] y el desmezclado espectral en

imágenes hiperespectrales [VRRPL03]. ISRA es un método de clasificación supervisada. Esto significa que la información sobre los endmembers es conocida de antemano. Una vez que se ha extraído el conjunto de endmembers $\mathbf{A}$ [PMPP04], las fracciones de abundancia $\hat{\mathbf{x}}$ correspondientes a un píxel determinado $\mathbf{b}$ pueden estimarse (utilizando mínimos cuadrados) mediante la siguiente expresión que no impone restricciones [Cha03]:

$$\hat{\mathbf{x}}_{MC} = (\mathbf{A}^T\ \mathbf{A})^{-1}\ \mathbf{A}^T\ \mathbf{b} \tag{6.4}$$

Las principales ventajas de esta aproximación son la sencillez de su implementación y su rápida ejecución. Su principal desventaja es que se producen valores negativos en las abundancias obtenidas, lo que no tiene un significado físico [KM02]. Ademas, los resultados estimados tampoco cumplen con la restricción de suma unitaria $\sum_{i=1}^{n} x_i = 1$. Imponiendo la restricción de abundancias no negativas resulta en el siguiente problema de optimización:

$$\begin{gathered} min\ _{\hat{\mathbf{x}}\,\in\,\Delta} \left\{(\mathbf{b} - \hat{\mathbf{x}} \cdot \mathbf{A})^T\ (\mathbf{b} - \hat{\mathbf{x}} \cdot \mathbf{A})\right\}, \\ sujeto\ a : \Delta = \{\hat{\mathbf{x}} \mid \hat{\mathbf{x}}_i \geq 0\ para\ todo\ 1 \leq i \leq n\} \end{gathered} \tag{6.5}$$

Como se indica en [CH00], se puede utilizar un algoritmo iterativo para resolver el problema de mínimos cuadrados con la restricción de no negatividad descrito en la Ecuación (6.5). La solución, es la Ecuación (6.6) que describe la base del algoritmo iterativo.

$$\hat{x}_j^{k+1} = \hat{x}_j^k\ \frac{\sum_{i=1}^{m} b_i\ a_{ij}}{\sum_{i=1}^{m} a_{ij}\ a_i^T\ \hat{X}} \tag{6.6}$$

El número de bandas y el número de endmembers se representan por m y n, respectivamente. La matriz $\mathbf{A} \in \Re_+^{m \times n}$ es la matriz de endmembers $(m \times n)$, donde a_{ij} es un elemento de $\mathbf{A}$ y a_i es el vector de la respuesta espectral de un endmember en todas las bandas i. El término $\mathbf{b} \in \Re_+^m$ denota el píxel que se está tratando y $\hat{\mathbf{x}}$ es el vector de abundancias.

Algoritmo 4 Pseudocódigo del ISRA

```
// Para cada uno de los píxeles
for (t = 0; t < P; t++) {
  // Calcular abundancias utilizando ISRA para un píxel
  for (k = 0; k <= NUM_ITER; k++) {
    // Para todos los endmembers
    for (j = 0; j < N; j++) {
      // Para todas las bandas
      for (i = 0; i < M; i++) {
        numerador = numerador + A[i][j]*B[i][t]; //Aij*Bi
        // Calcular el dot-product: traspuesta(Ai)*X
        for (s = 0; s < N; s++) {
          dot += A[i][s]*X[s][t];
        }end for
        denominador += dot * A[i][j]; // Aij*traspuesta(Ai)*X
        dot = 0;
      }end for
      // Calcular el nuevo X
      X[j][t] = X[j][t]*(numerador/denominador);
      numerador = 0;
      denominador = 0;
    }end for
  }end for
}end for
```

El Algoritmo 4 muestra el pseudocódigo para el ISRA. En este pseudocódigo las variables P y NUM_ITER representan el número de píxeles de la imagen y el número de iteraciones por píxel en el análisis, respectivamente. La ecuación ISRA se separa en el cálculo del numerador y el cálculo del denominador. Cuando estos se obtienen, se dividen y se multiplican por la abundancia anterior X.

El cálculo de las fracciones de abundancia para cada uno de los píxeles es independiente, por lo que se pueden calcular de manera simultánea. Con el fin de conseguir la mayor paralelización posible, el Algoritmo 4 es rees-

Algoritmo 5 Pseudocódigo reordenado del ISRA en paralelo

```
// Para cada uno de los píxeles
par (t = 0; t < P; t++) {
  // Calcular abundancias utilizando ISRA para un píxel
  for (k = 0; k <= NUM_ITER; k++) {
    // Para todos los endmembers
    for (j = 0; j < N; j++) {
      // Para todas las bandas
      for (i = 0; i < M; i++) {
        numerador = numerador + A[i][j]*B[i][t]; //Aij*Bi
      }end for
      numerador = numerador * X[j][t];
      // Para todas las bandas
      for (i = 0; i < M; i++) {
        // Calcular el dot-product: traspuesta(Ai)*X
        for (s = 0; s < N; s++) {
          dot += A[i][s]*X[s][t];
        }end for
        denominador += dot * A[i][j]; // Aij*traspuesta(Ai)*X
        dot = 0;
      }end for
      // Calcular el nuevo X
      X[j][t] = numerador/denominador;
      numerador = 0;
      denominador = 0;
    }end for
  }end for
}end par
```

crito pensando, en esta ocasión, en una implementación hardware en la que se produzca una mayor reutilización hardware y por lo tanto, cada unidad de procesamiento consuma los menos recursos posibles. El resultado es el Algoritmo 5. En esta ocasión, para obtener el numerador, se calcula el dot-product entre el píxel y el endmember, y el resultado se multiplica por la fracción de abundancia anterior. Posteriormente, el denominador utiliza la

estructura anterior para ir calculando las sumas parciales que se irán acumulando hasta obtener el resultado. Finalmente, se divide el numerador entre el denominador.

6.2. Implementación en FPGA

La Figura 6.3 muestra la arquitectura hardware utilizada para implementar el ISRA, junto con las comunicaciones de E/S. Al igual que en los anteriores diseños, para la entrada de datos se utiliza una memoria DDR2 SDRAM y un DMA (controlado por el PowerPC) con una FIFO de escritura para almacenar los datos de los píxeles. El módulo ISRA se emplea para implementar la versión propuesta del ISRA. Finalmente, se utiliza una FIFO de lectura junto con un transmisor para enviar las fracciones de abundancia a través del puerto RS232. La estructura general del hardware empleado para implementar el ISRA puede ser visto como una arquitectura segmentada. Se pueden distinguir tres etapas que se comunican utilizando una estructura first-in first-out (FIFO): La primera etapa proporciona los datos necesarios al sistema (endmembers y datos de la imagen), la segunda etapa lleva a cabo el proceso de cálculo de fracciones de abundancia para cada uno de los píxeles y finalmente en la tercera etapa se envían dichas fracciones de abundancia a través del puerto RS232. Por lo tanto, todas las etapas están trabajando en paralelo.

Por otro lado, la Figura 6.4 muestra la arquitectura hardware utilizada para implementar el ISRA. Se utilizan tres memorias diferentes para almacenar, respectivamente, los endmembers, el píxel actual y las fracciones de abundancia para el píxel actual. Finalmente, la unidad básica ISRA representa la ruta de datos utilizada para realizar los cálculos. La unidad de control es la encargada de llevar a cabo la correcta ejecución del ISRA: realiza las lecturas de las posiciones de memoria adecuadas en cada una de las memorias y realiza las actualizaciones de las fracciones de abundancia escribiendo en la memoria correspondiente. Además, se utiliza un circuito combinacional consistente en multiplexores para seleccionar la entrada de datos adecuada. Finalmente, realiza la escritura de las fracciones de abundancia estimadas en la FIFO de lectura.

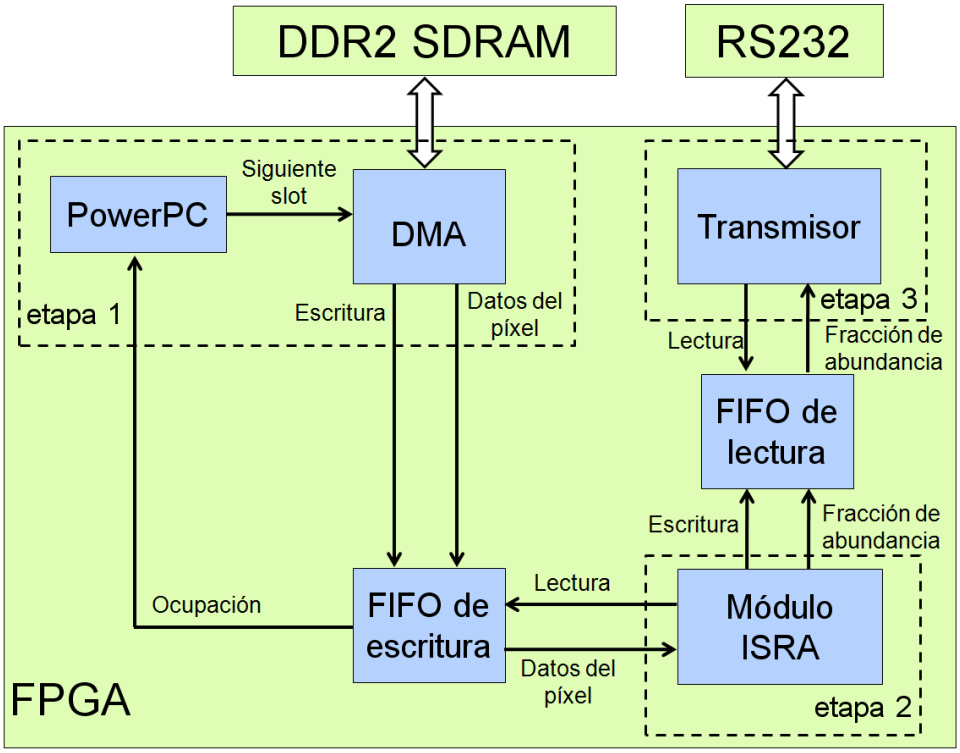


Figura 6.3: Arquitectura hardware para implementar el sistema ISRA completo.

La Figura 6.5 describe la arquitectura de la ruta de datos utilizada para implementar la unidad básica ISRA siguiendo los criterios de diseño discutidos en el apartado anterior. La unidad dot-product se utiliza tanto para el cálculo del numerador como del denominador, permitiendo así una reutilización de recursos hardware de forma óptima. Para llevar a cabo la actualización de una fracción de abundancia, se procede de la siguiente manera: durante m ciclos (siendo m el número de bandas), se computa el cálculo del dot-product entre el píxel actual y el endmember correspondiente a la fracción de abundancia que se está actualizando, gracias a la lectura de estos datos por parte de la unidad de control. En el ciclo de reloj siguiente, el resultado del dot-product se multiplica por la fracción de abundancia anterior y el resultado se almacena en el registro **num**, concluyendo así, el cálculo del numerador. Para calcular el denominador, el proceso anterior se repite n veces (siendo n el número de endmembers) con la entrada de datos adecuada,

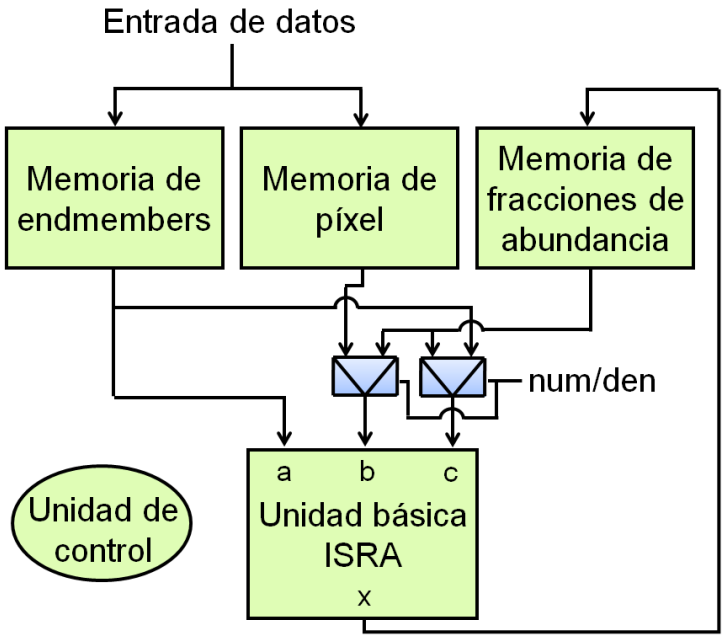


Figura 6.4: Arquitectura hardware para implementar el ISRA.

mientras los resultados parciales se van acumulando utilizando un sumador y el registro **den**. El cálculo del denominador requiere por lo tanto $n \times (m+1)$ ciclos de reloj. El proceso finaliza con la división entre el numerador y el denominador en 5 ciclos de reloj.

Dado que las fracciones de abundancia se pueden calcular de manera simultánea para distintos píxeles, la etapa segunda (la que lleva a cabo el ISRA) se puede paralelizar de la forma ilustrada en la Figura 6.6. Como se puede observar, el módulo ISRA se replica p veces y se añaden dos árbitros, uno para las lecturas de la FIFO de escritura y el envío de datos a la unidad o unidades correspondientes y otro para las escrituras de las fracciones de abundancia en la FIFO de lectura. El número de veces que podemos replicar la unidad ISRA vendrá determinado por la cantidad de recursos hardware disponibles y establecerá el *speedup* de la aplicación con respecto a la implementación sin paralelizar. Se podría pensar que introducir un árbitro, para un gran número de módulos, condicionaría el camino crítico del sistema. Sin embargo, dado que sobre todos los píxeles se realizan el mismo número de iteraciones, el comportamiento de los árbitros es muy predecible ya que, tan-

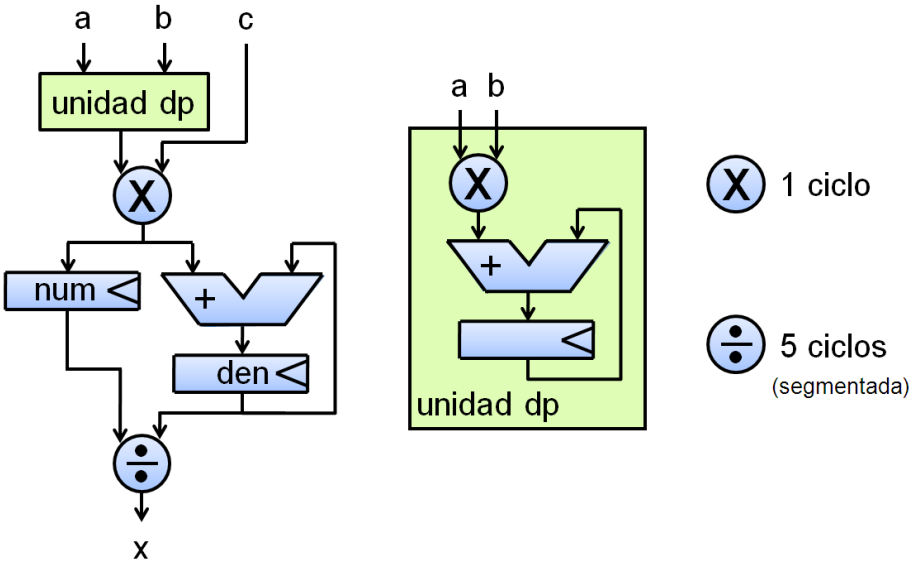


Figura 6.5: Arquitectura hardware para implementar la unidad básica ISRA.

to el envío de datos desde la FIFO de escritura como la escritura en la FIFO de lectura, se hará en orden. Por lo tanto, para su implementación basta con un contador que vaya atendiendo las peticiones en orden.

Con el fin de reducir la cantidad de recursos hardware necesarios para implementar el módulo ISRA, se realizó un estudio sobre el consumo de recursos hardware de la unidad básica ISRA. La Tabla 6.1 muestra el resultado de dicho estudio. Como se puede observar, el divisor por sí mismo consume un tercio de los recursos de la unidad básica y sin embargo, es el componente que se utiliza la menor parte del tiempo. Por lo tanto, es muy buen candidato para la mejora de consumo de recursos. La solución propuesta es tener un único divisor compartido por todos los módulos ISRA e ir atendiendo las peticiones de cada una de las unidades ISRA con ayuda de un árbitro. Dado que el número de ciclos de reloj que transcurre entre dos divisiones para un mismo módulo ISRA, $c = 5 + (m + 1) + n \times (m + 1)$, es elevado comparado con los ciclos de reloj de lectura de los píxeles m (es el tiempo que marca la separación entre las divisiones para dos módulos consecutivos), si el número de módulos ISRA en paralelo p cumple: $p < m.c.m.(c, m)/m$, estamos en condiciones de afirmar que el árbitro no es necesario puesto que no se producirán colisiones. Si alcanza o supera dicha cantidad, el uso de

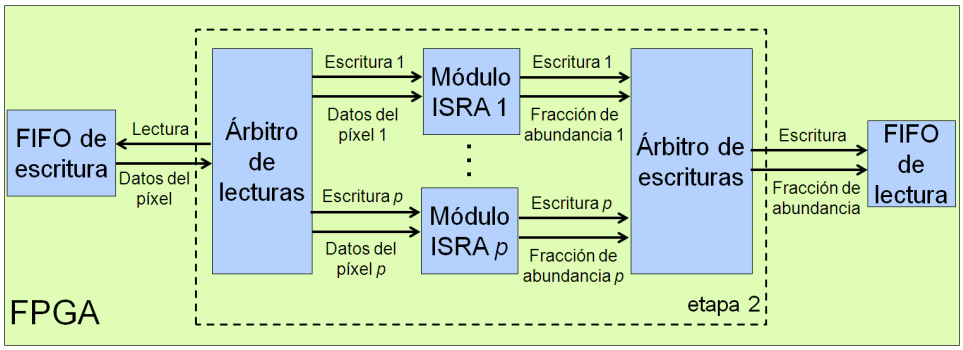


Figura 6.6: Arquitectura hardware para la paralelización del ISRA.

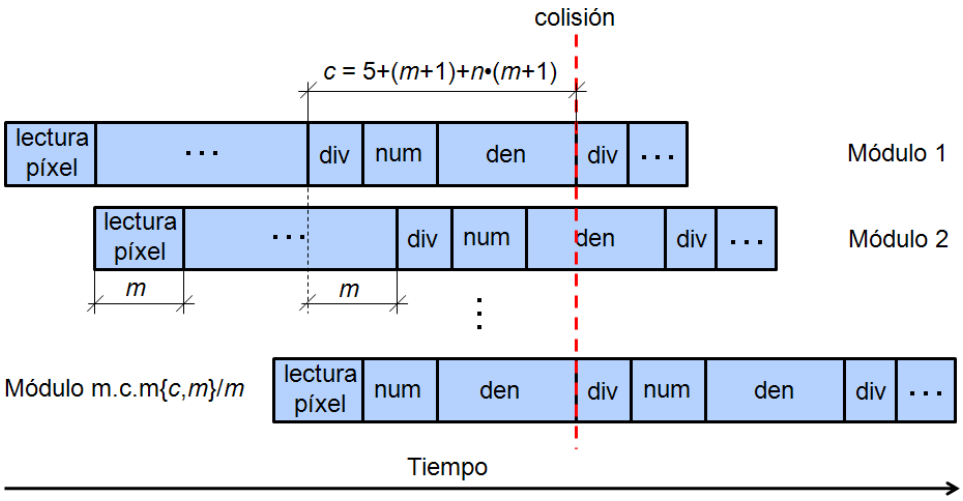


Figura 6.7: Diagrama de tiempos con los módulos ISRA en paralelo.

un árbitro es imprescindible. La Figura 6.7 muestra un diagrama de tiempos con los módulos ISRA en paralelo que ilustra esta colisión.

Para concluir esta sección, se ofrece una descripción paso a paso de cómo la arquitectura propuesta realiza la estimación de las fracciones de abundancia a partir de n endmembers para cada uno de los píxeles de una imagen hiperespectral:

- En primer lugar, el PowerPC envía una orden al DMA para escribir los n endmembers en la FIFO de escritura.
- Posteriormente, el árbitro de lectura lee estos endmembers y los envía

Tabla 6.1: Resumen de los recursos utilizados para la implementación en FPGA de la unidad básica ISRA en la FPGA Virtex-4 XC4VFX60.

Componente	Número de slice flip flops	Número de 4 input LUTs	Número de slices
Unidad dp	132	891	492
Registro num o den	32	0	0
Multiplicador	33	167	88
Sumador	0	538	301
Divisor	219	775	441
Unidad básica ISRA (con divisor)	458	2427	1367
Unidad básica ISRA (sin divisor)	239	1652	929

a todos los módulos ISRA donde se almacenan en cada memoria de endmembers.

- Después de que el DMA ha escrito el conjunto de endmembers, el PowerPC envía una orden al DMA para comenzar a copiar un trozo de la imagen de la memoria DDR2 SDRAM a la FIFO de escritura. Como se mencionó anteriormente, el principal cuello de botella en este tipo de sistema es con frecuencia la entrada de datos que se aborda en nuestra aplicación con la incorporación de un DMA que elimina la mayor parte de las penalizaciones de E/S. Por otra parte, el PowerPC monitorea la FIFO de escritura y envía una nueva orden al DMA cada vez que se detecta que la FIFO de escritura está medio vacía. Esta vez, el DMA traerá un trozo de la imagen que ocupa la mitad de la capacidad total de la FIFO.

- Cuando los datos del primer píxel se han escrito en la FIFO de escritura, el árbitro de lectura los envía al primer módulo ISRA y comienza a trabajar. Mientras se van almacenando los datos del píxel, se inicializan los valores de las fracciones de abundancia al valor 1 / *número de endmembers*, con la idea de que es una buena estimación de partida suponer que todos los endmembers contribuyen al píxel mezcla de la

misma manera. Una vez que el píxel se ha almacenado, se calculan sus fracciones de abundancia utilizando ISRA según se ha explicado anteriormente y finalmente se escriben dichas abundancias en la FIFO de lectura a través del árbitro de escrituras.

- Por último, el transmisor extrae las fracciones de abundancia de la FIFO de lectura y las envía a través de un puerto RS232.

- A medida que se van escribiendo los datos del siguiente píxel en la FIFO de escritura, el árbitro de lectura los envía al módulo ISRA que corresponda cuando este termine la ejecución de un píxel anterior. De esta manera se consigue que los distintos módulos ISRA trabajen en paralelo. Cabe señalar que, salvo inicialmente, no se producirán conflictos en el envío de datos a los distintos módulos ISRA y que jamás se producirá un conflicto en la escritura de las fracciones de abundancia al ser el número de endmembers menor que el número de bandas.

6.3. Resultados experimentales

6.3.1. Plataforma reconfigurable

La arquitectura hardware descrita en la sección 6.2 ha sido implementada utilizando VHDL para la especificación del módulo ISRA paralelo. Además, se han empleado los entornos Xilinx ISE y Embedded Developed Kit (EDK) para especificar el sistema completo. Este sistema ha sido implementado en la placa ML410 (ver Figura 4.10), una placa reconfigurable de bajo coste con una sola FPGA Virtex-4 XC4VFX60, una ranura DDR2 SDRAM que admite hasta 2 GBytes, un puerto RS232 y algunos componentes adicionales que no se han utilizado en la implementación propuesta.

6.3.2. Conjunto de imágenes hiperespectrales

Nuevamente, se vuelven a utilizar algunas de las imágenes hiperespectrales empleados en la evaluación de los algoritmos PPI y N-FINDR y que

fueron descritas en detalle en la sección 4.3.2. Las imágenes hiperespectrales corresponden con la conocida escena AVIRIS Cuprite (en unidades de reflectancia) sobre la región minera de Cuprite en Nevada y la escena AVIRIS Jasper Ridge (tanto en unidades de radiancia como de reflectancia) sobre la reserva biológica de Jasper Ridge en California.

6.3.3. Evaluación de las fracciones de abundancia

Antes de analizar las propiedades paralelas de la implementación en FPGA propuesta, se llevaron a cabo los experimentos para comprobar la precisión de las fracciones de abundancia estimadas y estudiar así la convergencia del algoritmo. Por lo tanto, en este apartado se evalúa la exactitud de los píxeles estimados por la implementación ISRA propuesta utilizando diferentes escenas hiperespectrales. En primer lugar, hacemos hincapié en que la versión hardware propuesta obtiene exactamente los mismos resultados que una implementación software propia. Ambas implementaciones se han validado tras testearlas con un amplio conjunto de datos sintéticos de resultados conocidos, con el fin de garantizar que las dos versiones ofrecen exactamente los mismos resultados en los escenarios considerados.

En este caso, vamos a considerar el *problema de estimación de abundancias* asumiendo que los endmembers son conocidos (en el siguiente capítulo se aborda el *problema de desmezclado completo*). Estos endmembers comprenden un conjunto de firmas de la biblioteca espectral USGS en el caso de la escena AVIRIS Cuprite (disponibles en unidades de reflectancia) y un conjunto de píxeles de la imagen etiquetados como espectralmente puros en el caso de la escena AVIRIS Jasper Ridge (disponible tanto en unidades de radiancia como reflectancia). A partir de dichos endmembers se van a estimar sus fracciones de abundancia para cada uno de los píxeles de la imagen hiperespectral.

Antes de mostrar los resultados obtenidos, se describe primero la métrica utilizada para la comparación cuantitativa en los experimentos realizados. Dado que ISRA utiliza la función de distancia de mínimos cuadrados en el problema de minimización de distancia, la métrica utilizada para evaluar la calidad de la reconstrucción es la raíz del error cuadrático medio (RECM) en-

tre la escena hiperespectral original y la reconstruida, que se pude definir de la siguiente manera. Asumamos que $\mathbf{I}^{(O)}$ es la escena hiperespectral original e $\mathbf{I}^{(R)}$ es una versión reconstruida de $\mathbf{I}^{(O)}$, obtenida utilizando la Ecuación (6.1) con un conjunto de endmembers y sus correspondientes fracciones de abundancia estimadas. Asumamos también que el píxel de coordenadas espaciales (i, j) en la escena hiperespectral original viene dado por $\mathbf{X}^{(O)}(i,j) = \left[x_1^{(O)}(i,j), x_2^{(O)}(i,j), \cdots, x_n^{(O)}(i,j)\right]$, mientras que el píxel correspondiente a las mismas coordenadas espaciales en la escena hiperespectral reconstruida viene dado por $\mathbf{X}^{(R)}(i,j) = \left[x_1^{(R)}(i,j), x_2^{(R)}(i,j), \cdots, x_n^{(R)}(i,j)\right]$. Con esta notación en mente, la RECM entre la escena hiperespectral original y reconstruida se calcula de la siguiente manera:

$$\mathbf{RECM}(\mathbf{I}^{(O)}, \mathbf{I}^{(R)}) = \frac{1}{s \times l} \sum_{i=1}^{s} \sum_{j=1}^{l} \left(\frac{1}{n} \sum_{k=1}^{n} \left[x_k^{(O)}(i,j) - x_k^{(R)}(i,j) \right] \right)^{\frac{1}{2}} \quad (6.7)$$

donde s denota el número de muestras, l el número de líneas y k el número de bandas. Como resultado, valores bajos de RECM significan una alta proximidad entre las imágenes comparadas.

Se han llevado a cabo los experimentos sobre la precisión de las fracciones de abundancia estimadas con las escenas AVIRIS Cuprite y AVIRIS Jasper Ridge con el fin de evaluar la similitud entre la escena hiperespectral original y la reconstruida por la implementación ISRA propuesta. El único parámetro de entrada para el ISRA (sin olvidarnos del conjunto de endmembers y la propia escena hiperespectral) es el número de iteraciones por píxel en el análisis. En los experimentos realizados, este número se hace variar entre 10 y 600 para ilustrar la convergencia del algoritmo (más allá de 600 iteraciones no se producen variaciones en los resultados).

Las Figuras 6.8, 6.9 y 6.10 muestran la RECM por píxel (y la media) entre la escena hiperespectral original y la reconstruida para distinto número de iteraciones en las escenas AVIRIS Cuprite y AVIRIS Jasper Ridge, esta última tanto en unidades de reflectancia como de radiancia. Colores más cálidos indican mayor error. A la vista de los resultados, podemos concluir que 100 iteraciones es un número suficiente y que más allá de 200 iteraciones la RECM decrece muy lentamente, por lo que la mejora en la similitud entre

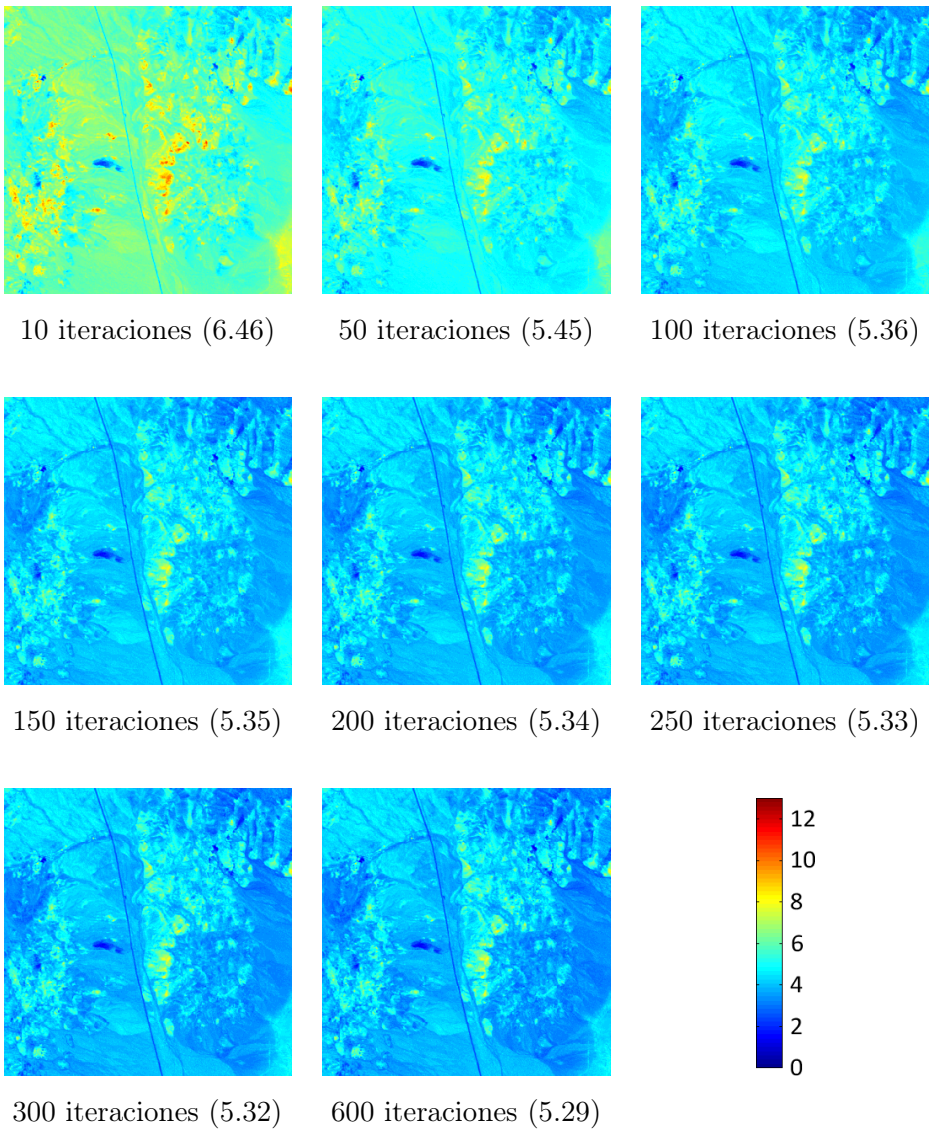


Figura 6.8: Mapa de errores RECM (en porcentaje) para distinto número de iteraciones después de la reconstrucción de la escena AVIRIS Cuprite utilizando las firmas de referencia de la librería USGS.

la imagen original y la reconstruida es muy pequeña. Cabe destacar, que el error cometido en la reconstrucción de la escena AVIRIS Cuprite es mayor que para las escenas AVIRIS Jasper Ridge. Esto se debe a que en el primer caso se utilizan como endmembers firmas extraídas de la librería USGS mientras que en el segundo caso, se emplean firmas espectrales puras de la

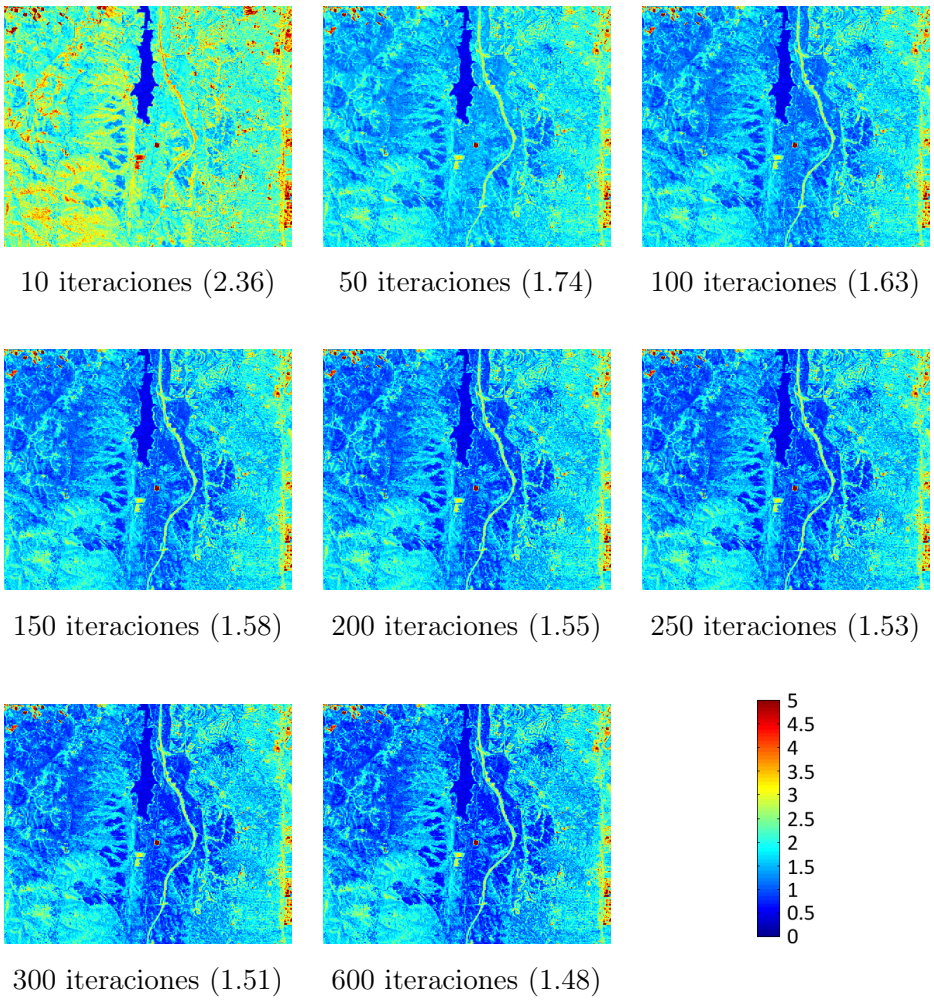


Figura 6.9: Mapa de errores RECM (en porcentaje) para distinto número de iteraciones después de la reconstrucción de la escena AVIRIS Jasper Ridge en unidades de reflectancia utilizando las firmas espectrales de terreno de referencia.

propia imagen hiperespectral. Finalmente, si el desmezclado espectral se va a utilizar para realizar una compresión con pérdidas a bordo, comparando las imágenes reconstruidas para la escena AVIRIS Jasper Ridge en unidades de reflectancia y radiancia, parece claro que es mejor realizar la compresión con los datos en unidades de radiancia, antes de realizar el proceso de corrección atmosférica. De esta forma, se obtendrá una tasa de compresión en torno al 77.6 % al pasar de tener 224 bandas con 16 bits de resolución a 5 fracciones

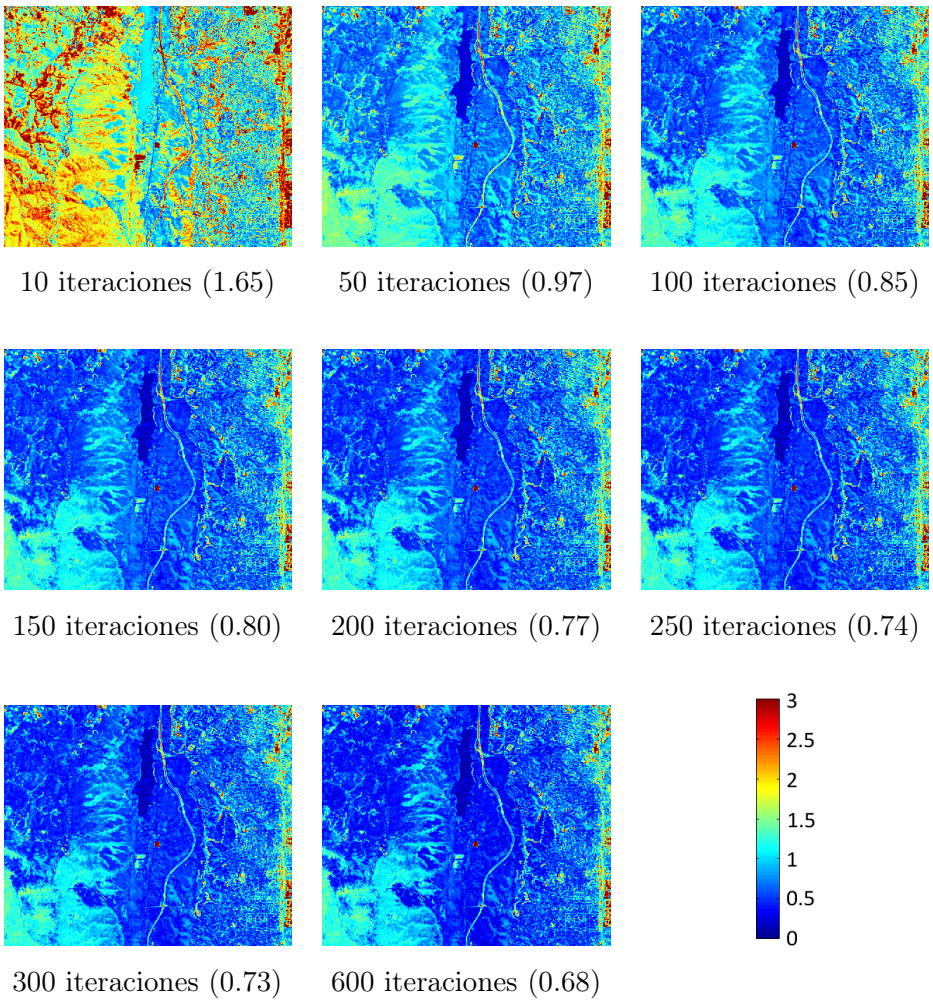


Figura 6.10: Mapa de errores RECM (en porcentaje) para distinto número de iteraciones después de la reconstrucción de la escena AVIRIS Jasper Ridge en unidades de radiancia utilizando las firmas espectrales de terreno de referencia.

de abundancia con 32 bits de resolución (224:10). Esta tasa se verá disminuida al añadir los metadatos propios de la técnica de compresión, en este caso, las 224 bandas de los 5 endmembers.

6.3.4. Evaluación del rendimiento

En este apartado se lleva a cabo una evaluación experimental del rendimiento computacional de la implementación en FPGA propuesta. A título ilustrativo, la Tabla 6.2 muestra los recursos utilizados para la implementación hardware del diseño propuesto para el ISRA para diferentes valores de p (número de módulos ISRA en paralelo), llevada a cabo en la FPGA Virtex-4 XC4VFX60 de la placa ML410. Esta FPGA tiene un total de 25280 slices, 50560 slice flip flops y 50560 LUTs de cuatro entradas disponibles. Además esta FPGA incluye algunos recursos heterogéneos, tales como dos PowerPCs, 128 DSP48Es y Block RAMs distribuidas. En la implementación propuesta se han aprovechado estos recursos para optimizar el diseño. Un PowerPC supervisa las comunicaciones y las Block RAMs se utilizan para implementar las FIFOs, por lo que la gran mayoría de los slices se utilizan para la implementación del ISRA junto con los multiplicadores DSP48Es.

Como muestra la Tabla 6.2, nuestro diseño puede escalarse hasta 16 módulos ISRA en paralelo. Un comportamiento interesante del módulo ISRA paralelo propuesto es que se puede escalar sin que apenas aumente el retardo del camino crítico (la frecuencia de reloj permanece prácticamente constante). Para el máximo valor utilizado en los experimentos ($p = 16$) el porcentaje total de utilización hardware es del 90,08 % alcanzando casi la total ocupación. Sin embargo, para otros valores de p, por ejemplo, $p = 10$, tan sólo se utilizan la mitad de los recursos disponibles, quedando aún espacio en la FPGA para algoritmos adicionales. En cualquier caso, este valor puede aumentarse considerablemente en las últimas placas de FPGA, pero en los experimentos realizados se ha decidido informar sólo de los resultados con esta placa porque es muy similar a otra certificada para el espacio.

Otro aspecto importante de la implementación hardware propuesta son las comunicaciones, que a menudo son el principal cuello de botella de los sistemas paralelos. Por lo tanto, al igual que en el diseño de los algoritmos PPI y N-FINDR, se ha prestado especial atención a este problema. Para reducir las penalizaciones de E/S, se ha incluido un DMA y se ha aplicado una técnica de precarga con el fin de ocultar la latencia de las comunicaciones. Básicamente, mientras que los módulos ISRA están procesando un conjunto

Tabla 6.2: Resumen de los recursos utilizados para la implementación en FPGA del ISRA para distinto número de módulos en paralelo en la FPGA Virtex-4 XC4VFX60.

Componente	Número de módulos (p)	Número de DSP48Es	Número de slice flip flops	Número de 4 input LUTs	Número de slices	Porcentaje total	Frecuencia máxima (MHz)
Módulo ISRA paralelo	10	100	3368	23357	12719	50.31	44.7
	11	110	3683	25654	13956	55.2	44.5
	12	120	3990	27920	15186	60.07	44.4
	13	128	4383	30394	16547	65.45	44.3
	14	128	4758	34300	18660	73.81	44.1
	15	128	5147	38112	20717	81.95	43.9
	16	128	5532	42047	22773	90.08	43.8
Transmisor RS232	-	0	69	128	71	0.28	208
Controlador DMA	-	0	170	531	367	1.45	102

Tabla 6.3: Tiempos de procesamiento en minutos para la implementación hardware y para la versión software equivalente de ISRA para las imágenes hiperespectrales consideradas según el número de iteraciones.

Número de	AVIRIS Cuprite		AVIRIS Jasper Ridge	
iteraciones	HW	SW	HW	SW
10	0.15	1.61	0.41	5.043
50	0.79	8.05	2.05	25.34
100	1.59	16.11	4.10	50.45
150	2.39	24.17	6.15	75.89
200	3.19	32.22	8.20	101.05
250	3.99	40.28	10.25	126.66
300	4.79	48.34	12.30	151.59
600	9.59	96.68	24.61	302.80

de datos, el DMA irá a buscar la siguiente serie, y la almacenará en la FIFO de escritura. Para demostrar las ventajas de utilizar un DMA, hemos desarrollado otra versión en la que los datos de la imagen se leen de memoria y son escritos en la FIFO de escritura por el PowerPC en lugar del DMA. En esta versión, el tiempo de procesamiento se incrementó en más de un orden de magnitud por lo que podemos concluir que los recursos utilizados para el DMA están bien empleados.

Por último, la Tabla 6.3 da información de los tiempos de procesamiento de la implementación en FPGA considerada y una versión software equivalente desarrollada en lenguaje C y ejecutada en un PC con un procesador AMD Athlon 2.6 GHz y 512 Mb de RAM, en el conjunto hiperespectral de datos considerados. Ya que los tiempos de procesamiento de los datos AVIRIS Jasper Ridge en unidades de radiancia y reflectancia son exactamente iguales, sólo se muestra uno de ellos en la tabla. En todos los casos, el número de módulos ISRA utilizados en paralelo para la versión en FPGA es de $p = 16$.

En los experimentos realizados, los tiempos de procesamiento logrados en la placa FPGA considerada muestran un *speedup* por encima de 10 al procesar la escena AVIRIS Cuprite y un *speedup* por encima de 12 cuando se trata

de la escena AVIRIS Jasper Ridge. Aunque la inclusión de placas de FPGA más recientes podría mejorar de forma considerable el informe de tiempos de procesamiento, como trabajo futuro también nos centraremos en la mejora de la implementación propuesta para lograr un mejor aprovechamiento de los recursos hardware y reducir los tiempos de procesamiento.

6.4. Conclusiones

Uno de los grandes retos en el análisis de imágenes hiperespectrales en observación remota de la tierra es la complejidad computacional como resultado de la necesidad de procesar enormes volúmenes de datos. Con los recientes avances en hardware reconfigurable, muchos de los algoritmos de procesamiento de imágenes hiperespectrales se pueden acelerar mediante FPGAs de alto rendimiento. Una técnica importante para el análisis de datos hiperespectrales es el desmezclado espectral, en el cual la estimación de abundancias es una tarea fundamental. En este capítulo, se ha descrito una implementación en FPGA del Image Space Reconstruction Algorithm (ISRA), una de las aproximaciones más conocidas para la estimación de fracciones de abundancia en el análisis de datos hiperespectrales en observación remota de la tierra. Este trabajo demuestra una vez más la viabilidad de la utilización de FPGAs en el análisis de imágenes hiperespectrales especialmente con algoritmos de cálculo iterativo o intensivo. Los resultados experimentales, realizados en una FPGA Virtex-4 XC4VFX60 demuestran que la implementación hardware propuesta puede superar de manera significativa (en términos de tiempo de cálculo) una versión software equivalente y también es capaz de proporcionar resultados precisos con un tamaño compacto, que hacen del sistema reconfigurable propuesto atractivo para el procesamiento a bordo de datos hiperespectrales. Otra característica interesante de la implementación propuesta es que es fácilmente escalable para FPGAs mayores. Por ejemplo, si dispusiéramos de una FPGA Virtex-4 XQR4VLX200 (89088 slices) certificada para el espacio, podríamos tener 3.5 veces más unidades ISRA en paralelo simplemente sintetizando la etapa 2 de la arquitectura hardware para el nuevo número de unidades en paralelo. De esta manera, se podría alcanzar un *speedup* de 3.5 sin realizar modificaciones en el diseño.

Capítulo 7

Cadena completa de desmezclado espectral

Una cadena es tan fuerte
como el más débil de sus eslabones.

Anónimo

Los instrumentos hiperespectrales captan la energía electromagnética que se encuentra dentro de su campo de visión terrestre en cientos de canales espectrales con una alta resolución. Muy a menudo, sin embargo, debido a la moderada resolución espacial del sensor o a la presencia de mezclas íntimas (mezcla de los materiales a una escala muy pequeña) en la escena, los vectores espectrales (recopilación de las señales adquiridas en diferentes bandas espectrales para un píxel determinado) adquiridos por los sensores hiperespectrales son en realidad mezclas de las firmas espectrales de los materiales presentes en la escena.

En este capítulo, se presenta un diseño para la implementación en FPGA de la cadena completa de desmezclado espectral. Tras una sección en la que se realiza una comparativa entre los algoritmos PPI y N-FINDR propuestos para determinar cual de los dos es más conveniente utilizar, se muestra la implementación en FPGA de la cadena de desmezclado espectral, utilizando la reconfigurabilidad de estos dispositivos, y los resultados experimentales

llevados a cabo en la FPGA Virtex–4 XCVFX60 de Xilinx empleando las imágenes hiperespectrales reales captadas por el Airborne Visible Infra–Red Spectrometer (AVIRIS) de la NASA sobre la región minera de Cuprite en Nevada y sobre la reserva biólogica de Jasper Ridge en California.

7.1. Comparativa entre los algoritmos PPI y N-FINDR

En esta sección se va a realizar una comparación entre las implementaciones propuestas en los Capítulos 4 y 5 de los algoritmos PPI y N-FINDR, respectivamente, para la extracción de endmembers. El proceso de reducción dimensional resulta imprescindible para la posterior aplicación del algoritmo N-FINDR, sin embargo, como se ha demostrado en el Capítulo 4, este paso previo no es necesario para la ejecución del algoritmo PPI. La reducción dimensional nos permite reducir la complejidad computacional del problema y atenuar el efecto del ruido, pero se debe tener en cuenta que la información descartada puede contener las pequeñas diferencias que nos hagan distinguir unos materiales de otros.

Con el fin de realizar una comparativa lo más justa posible, vamos a partir de las imágenes reducidas dimensionalmente, mediante el Análisis de Componentes Principales (ACP) del software ENVI 4.0, del Capítulo 5 para realizar la extracción de endmembers sobre las mismas con el algoritmo PPI propuesto en el Capítulo 4. Las Tablas 7.1 y 7.2 muestran el ángulo espectral (AE) para los endmembers más similares extraídos por el algoritmo PPI y por el algoritmo N-FINDR en la escena AVIRIS Cuprite reducida a 15 superbandas y en la escena AVIRIS Jasper Ridge reducida a 18 superbandas tanto en unidades de reflectancia como de radiancia. A su vez, la Tabla 7.3 muestra el tiempo de cómputo del algoritmo PPI y del algoritmo N-FINDR para las distintas escenas hiperespectrales consideradas una vez han sido reducidas dimensionalmente y tras la evaluación de 10^4 skewers. Ya que los tiempos de procesamiento de los datos AVIRIS Jasper Ridge en unidades de radiancia y reflectancia son exactamente iguales, solo se muestra uno de ellos en la tabla.

Tabla 7.1: Ángulo espectral entre los endmembers extraídos por PPI y N-FINDR en la escena AVIRIS Cuprite reducida a 15 superbandas y las firmas disponibles en la librería espectral USGS.

	PPI		**N-FINDR**	
	Radianes	Grados	Radianes	Grados
Alunita	0.084	4.812	0.084	4.812
Buddingtonita	0.073	4.182	0.089	5.099
Calcita	0.092	5.271	0.105	6.016
Kaolinita	0.136	7.792	0.138	7.906
Moscovita	0.092	5.271	0.108	6.187

Primeramente se va a realizar la comparación cuantitativa entre los endmembers extraídos por las implementaciones propuestas de los algoritmos PPI y N-FINDR. Observando las Tablas 7.1 y 7.2 nos damos cuenta de que, a pesar que los valores de AE en ambos algoritmos demuestran una alta correspondencia espectral con las firmas espectrales de referencia disponibles en cada escena, en el caso del algoritmo PPI se obtienen valores más pequeños que con el algoritmo N-FINDR, lo que significa una mayor similitud espectral.

Con respecto a los tiempos de respuesta de ambos algoritmos cuando se procesan las imágenes reducidas dimensionalmente, observando la Tabla 7.3, queda claro que el algoritmo PPI es más rápido que el algoritmo N-FINDR con un *speedup* ligeramente superior a 12.

A pesar de que ambos algoritmos obtienen un conjunto de endmembers similares a las firmas de referencia, se debe considerar que los endmembers extraídos por el algoritmo PPI tienen mayor similitud espectral con respecto a dichas firmas, pero sobre todo, que realiza la extracción de endmembers más rápidamente. Además, el algoritmo N-FINDR tiene una serie de limitaciones: Una de ellas es la determinación del número de endmembers necesarios a generar por el algoritmo N-FINDR. Otra es su complejidad computacional como resultado de una búsqueda exhaustiva. Una tercera es el requisito de reducción dimensional de la escena, un proceso que suele ser muy complejo y que por tanto, incrementa notablemente el tiempo de cómputo. Una cuarta

Tabla 7.2: Ángulo espectral entre los endmembers extraídos por PPI y N-FINDR en las escenas AVIRIS Jasper Ridge reducida a 18 superbandas (en unidades de radiancia y reflectancia) y las firmas espectrales puras disponibles en ambas escenas.

		Datos en radiancia				
		Suelo	Bosque	Hierba	Chaparral	Lago
PPI	Radianes	0.065	0.061	0.045	0.042	0.032
	Grados	3.724	3.495	2.578	2.406	1.833
N-FINDR	Radianes	0.077	0.065	0.044	0.050	0.032
	Grados	4.411	3.724	2.521	2.864	1.833
		Datos en reflectancia				
		Suelo	Bosque	Hierba	Chaparral	Lago
PPI	Radianes	0.030	0.026	0.024	0.031	0.019
	Grados	1.718	1.489	1.375	1.776	1.088
N-FINDR	Radianes	0.028	0.025	0.022	0.020	0.019
	Grados	1.604	1.432	1.260	1.145	1.088

y probablemente la cuestión más crítica es el uso de endmembers iniciales aleatorios que dan lugar a la selección de un conjunto de endmembers finales que pueden ser inconsistentes y que los resultados no son reproducibles. Por todas estas razones, nos decantamos por la utilización del algoritmo PPI a la hora de implementar una cadena de desmezclado en FPGA, ya que, el problema de la determinación del número de endmembers a extraer se reduce a la determinación de un valor umbral y no es necesaria la reducción dimensional de la escena (no se pierde nada de información). Además, para FPGAs mayores el algoritmo PPI resulta más fácilmente paralelizable ya que la unidad básica de procesamiento consume menos recursos hardware.

Como trabajo futuro, debemos estudiar la conveniencia o no de realizar el paso previo de reducción dimensional, además del de corrección atmosférica, a la hora de extraer endmembers mediante el algoritmo PPI. Tres son los posibles escenarios con los que podemos encontrarnos: realizar la extracción de endmembers una vez que la imagen ha sido corregida atmosféricamente y reducida dimensionalmente (tratamiento habitual), realizar la reducción

Tabla 7.3: Tiempos de procesamiento para el algoritmo PPI y N-FINDR con las escenas consideradas reducidas dimensionalmente.

	AVIRIS Cuprite	**EO-1 Hyperion**	**AVIRIS Jasper Ridge**
Número de endmembers (p)	16	21	19
Tamaño total (Megabytes)	50	800	140
Tiempo de la implementación PPI (segundos)	1.35	18.87	3.48
Tiempo de la implementación N-FINDR (segundos)	13.46	239.11	49.35
Speedup	12.48	12.10	12.44

dimensional sobre los datos sin corregir atmosféricamente y posteriormente extraer los endmembers, o realizar la extracción de endmembers directamente sobre los datos en bruto según son recogidos por el sensor. Basándonos en los experimentos realizados intuimos que la similitud espectral de los endmembers en todos los casos será muy parecida, por lo que deberemos centrarnos más en estudiar los distintos tiempos de cómputo. Tenemos que elegir el algoritmo de reducción dimensional más conveniente si resulta más adecuado que tratar los datos en bruto. Se ahonda en esta cuestión en el siguiente capítulo.

7.2. Implementación en FPGA

Por las razones anteriormente expuestas, se ha elegido utilizar la implementación propuesta del algoritmo PPI para realizar la etapa de extracción de endmembers en la cadena completa de desmezclado espectral. Por lo tanto, en este trabajo no se ha incluido el paso previo de reducción dimensional mostrado en la Figura 2.12, que se destina principalmente a reducir el tiempo de procesamiento, pero en el que en ocasiones se descarta información relevante en el dominio espectral. Por ello, la cadena de procesamiento a

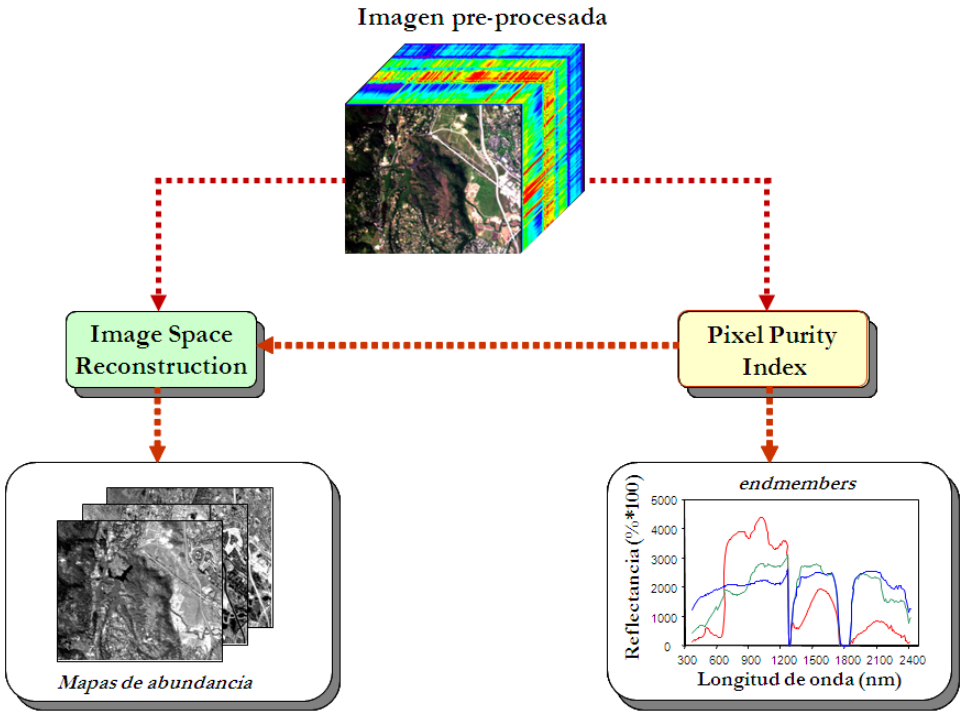


Figura 7.1: Cadena de desmezclado hiperespectral sin reducción dimensional.

implementar se ilustra en la Figura 7.1.

Nos vamos a valer de la reconfigurabilidad de las FPGAs para llevar a cabo el desmezclado espectral. La idea, ilustrada en la Figura 7.2, es paralelizar al máximo el algoritmo de extracción de endmembers ocupando toda la capacidad de la FPGA y aplicar reconfiguración dinámica para ocupar de nuevo toda la capacidad para el algoritmo de estimación de abundancias paralelizado. En nuestro caso particular, tanto el algoritmo PPI como el ISRA son fácilmente escalables y dado que sus unidades básicas de procesamiento consumen pocos recursos se aprovecha casi la totalidad de la FPGA. Uno de los principales inconvenientes ya comentados de las FPGAs es la penalización impuesta por el tiempo de reconfiguración, que hace que sea inviable en muchas aplicaciones. Esta penalización es del orden de milisegundos por lo que, en el contexto del análisis de imágenes hiperespectrales, supone una penalización muy poco significativa en comparación con el tiempo de ejecución de los algoritmos.

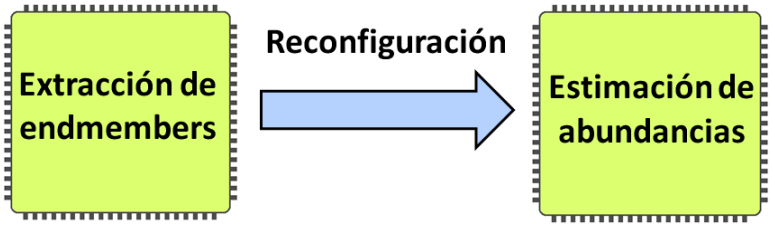


Figura 7.2: Cadena de desmezclado hiperespectral sin reducción dimensional.

Hasta el momento, el algoritmo PPI propuesto selecciona como endmembers aquellos píxeles de la imagen que aparecen como extremos por encima de un umbral predefinido. Sin embargo, algunos de estos endmembers son muy similares espectralmente por lo que se pueden considerar redundantes y ser eliminados. En el Capítulo 4, se comentó que este proceso se aborda generalmente como una etapa de posprocesamiento externo al algoritmo PPI, pero dado que se va a implementar una cadena completa de desmezclado espectral, es hora de abordar este proceso. Para la eliminación automática de endmembers con espectros redundantes se calcula el AE entre todos los pares y aquellos que tengan un valor de AE por debajo de un valor umbral predefinido serán considerados redundantes y uno de ellos será eliminado. Este proceso se ha implementado en el lenguaje C y ejecutado en el PowerPC empotrado en la FPGA, con unos tiempos de ejecución de 2.52 a 3.55 segundos para el tratamiento de 16 a 19 endmembers. Dado que la ejecución del ISRA supone la mayor fracción del tiempo de ejecución con aproximadamente 2 minutos, estos 3.55 segundos no suponen una fracción significativa del tiempo total de ejecución. Como trabajo futuro, se abordará la implementación del paso de eliminación de endmembers redundantes en FPGA aprovechándonos de nuevo de la reconfigurabilidad de las FPGAs, pero primeramente nuestros esfuerzos se centraran en la mejora del ISRA puesto que es el algoritmo que mayor fracción de tiempo está en ejecución.

Para que el proceso de eliminación de endmembers redundantes pueda llevarse a cabo en el PowerPC es necesario modificar la salida de datos del algoritmo PPI propuesto resultando en la arquitectura que se muestra en la Figura 7.3. En lugar de enviar los endmembers a través del puerto RS232, se van a llevar al PowerPC a través de la FIFO de lectura. Posteriormente

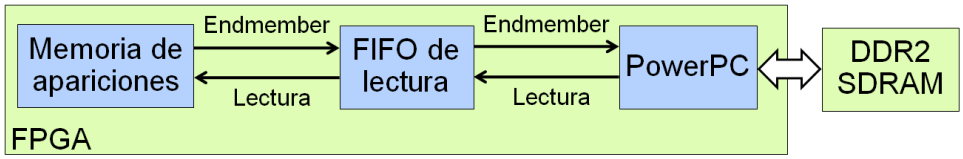


Figura 7.3: Modificación realizada en la implementación del algoritmo PPI para la eliminación de endmembers redundantes.

el PowerPC realiza el proceso de eliminación de endmembers redundantes y los endmembers finales son almacenados en la memoria DDR2 SDRAM para que posteriormente sean leídos por el ISRA.

Para concluir esta sección, se ofrece una descripción a grandes rasgos de cómo se realiza el proceso de desmezclado espectral propuesto:

- Primeramente, se lleva a cabo el proceso de extracción de endmembers mediante el algoritmo PPI tal y cómo se explicó en la sección 4.2.2 del Capítulo 4.
- Posteriormente, teniendo en cuenta la modificación de la salida de datos para el algoritmo PPI que se acaba de comentar, el PowerPC realiza el proceso de eliminación de endmembers redundantes y escribe los endmembers finales en la memoria DDR2 SDRAM.
- Después de que el PowerPC ha escrito el conjunto final de endmembers, envía una orden al System ACE (System Advanced Configuration Environment) para que cargue la configuración correspondiente del ISRA.
- Finalmente, el ISRA realiza la estimación de abundancias tal y cómo se explicó en la sección 6.2 del Capítulo 6.

7.3. Resultados experimentales

Este sistema ha sido implementado en la placa ML410 (ver Figura 4.10), una placa reconfigurable de bajo coste con una sola FPGA Xilinx Virtex–4 XC4VFX60, una ranura DDR2 SDRAM que admite hasta 2 GBytes, un

puerto RS232 y algunos componentes adicionales que no se han utilizado en la implementación propuesta. Nuevamente, se vuelve a utilizar el mismo conjunto de imágenes hiperespectrales empleados en la estimación de abundancias mediante ISRA del capítulo anterior. Las imágenes hiperespectrales por lo tanto, corresponden con la conocida escena AVIRIS Cuprite (en unidades de reflectancia) sobre la región minera de Cuprite en Nevada y la escena AVIRIS Jasper Ridge (tanto en unidades de radiancia como de reflectancia) sobre la reserva biológica de Jasper Ridge en California. En esta ocasión se van a utilizar los endmembers extraídos por el algoritmo PPI para la estimación de abundancias en lugar de las firmas de referencia disponibles para cada escena.

La Figura 7.4 muestra la RECM por píxel (y la media) entre la escena hiperespectral original y la reconstruida para distinto número de iteraciones en la escena AVIRIS Cuprite. Si la comparamos con la Figura 6.8 cabe destacar, que el error cometido en la reconstrucción de la escena AVIRIS Cuprite es menor en la Figura 7.4. Esto se debe a que en el primer caso se utilizan como endmembers firmas extraídas de la librería USGS mientras que en el segundo caso, se emplean las firmas espectrales de los endmembers de la propia imagen hiperespectral.

Las Figuras 7.5 y 7.6 muestra la RECM por píxel (y la media) en la escena AVIRIS Jasper Ridge en unidades de reflectancia y de radiancia, respectivamente. Si las comparamos con las Figuras 6.9 y 6.10 vemos que en esta ocasión el error cometido a la hora de reconstruir las imágenes originales es mayor en ambos casos. Esto es debido a la diferencia existente en los endmembers seleccionados por el algoritmo PPI y aquellos que se utilizaron como firmas espectrales puras del terreno.

El último experimento realizado trata de mostrar la influencia del proceso de corrección atmosférica a la hora de extraer los endmembers. Para ello, de la escena AVIRIS Jasper Ridge en unidades de radiancia se extrajeron los endmembers mediante el algoritmo PPI propuesto y posteriormente se utilizaron estos endmembers (sus posiciones) para reconstruir la escena AVIRIS Jasper Ridge en unidades de reflectancia. La Figura 7.7 muestra el resultado de la forma habitual. Comparando las Figuras 7.5 y 7.7 vemos que se producen resultados muy similares por lo que paralelizar el proceso

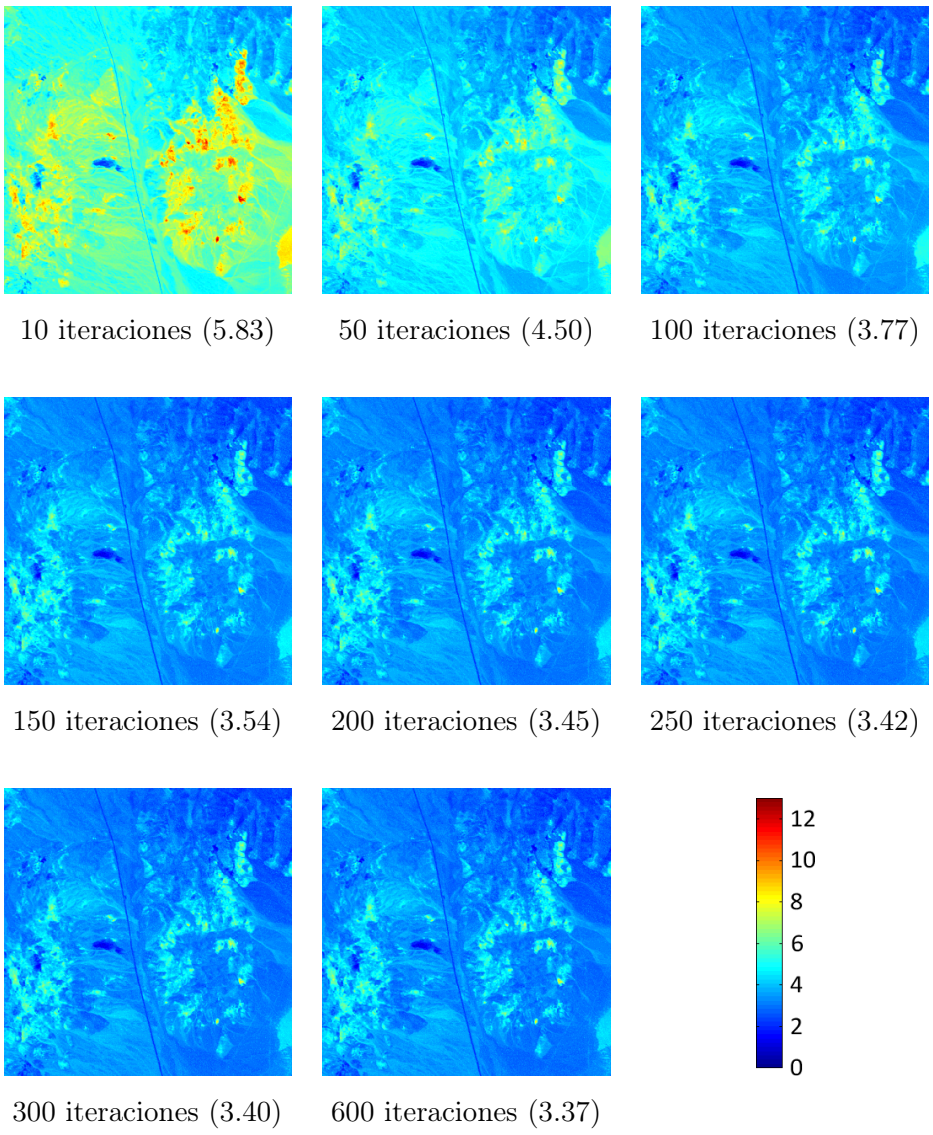


Figura 7.4: Mapa de errores RECM (en porcentaje) para distinto número de iteraciones después de la reconstrucción de la escena AVIRIS Cuprite con los endmembers extraídos por el algoritmo PPI.

de corrección atmosférica y el de extracción de endmembers puede aportar grandes beneficios en un sistema orientado al tratamiento en tiempo real.

Finalmente, la Tabla 7.4 da la información desglosada de los tiempos de procesamiento de la implementación en FPGA propuesta en el conjunto hiperespectral de datos considerados, tras la evaluación de 10^4 skewers y 100

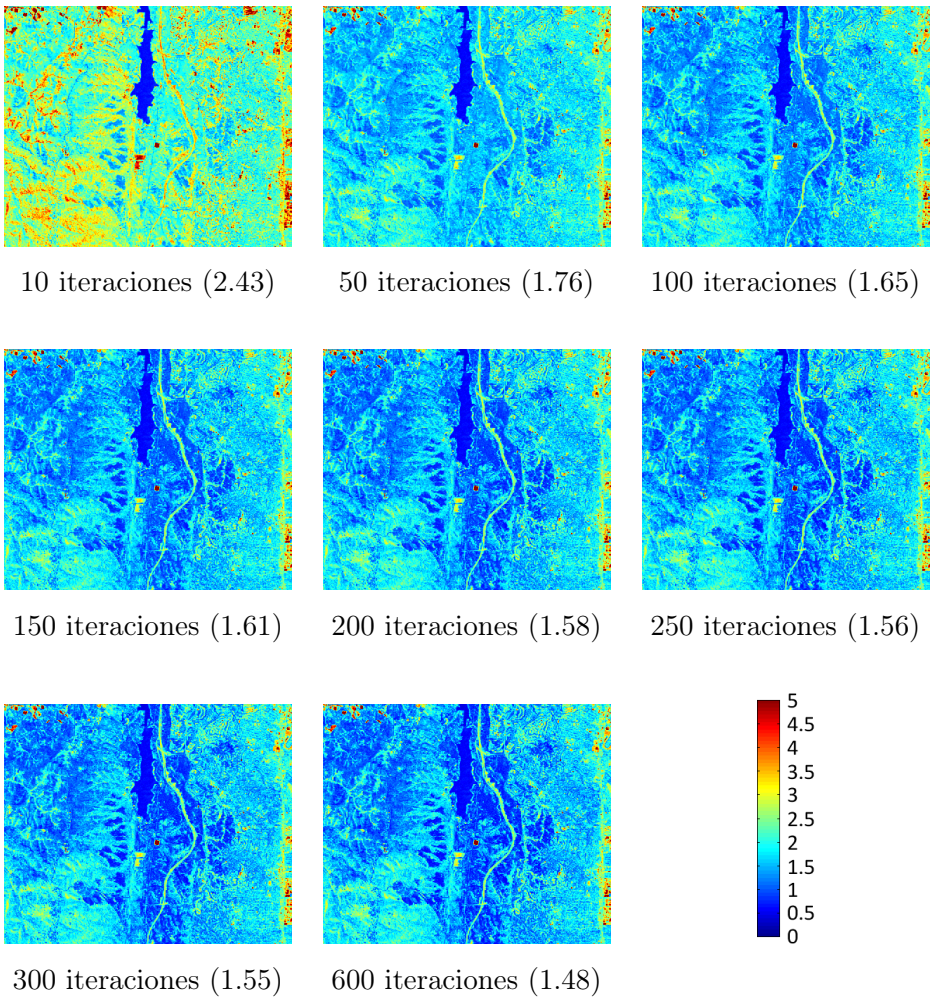


Figura 7.5: Mapa de errores RECM (en porcentaje) para distinto número de iteraciones después de la reconstrucción de la escena AVIRIS Jasper Ridge en unidades de reflectancia con los endmembers extraídos por el algoritmo PPI.

iteraciones.

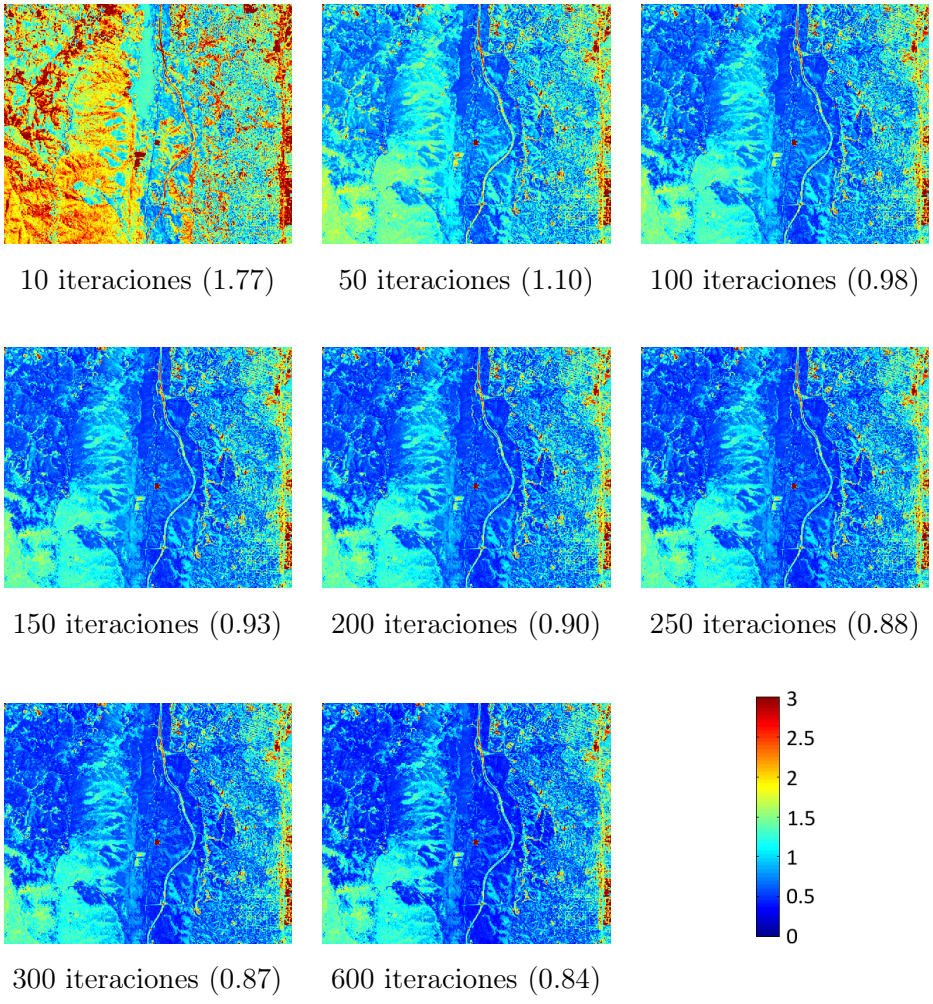


Figura 7.6: Mapa de errores RECM (en porcentaje) para distinto número de iteraciones después de la reconstrucción de la escena AVIRIS Jasper Ridge en unidades de radiancia con los endmembers extraídos por el algoritmo PPI.

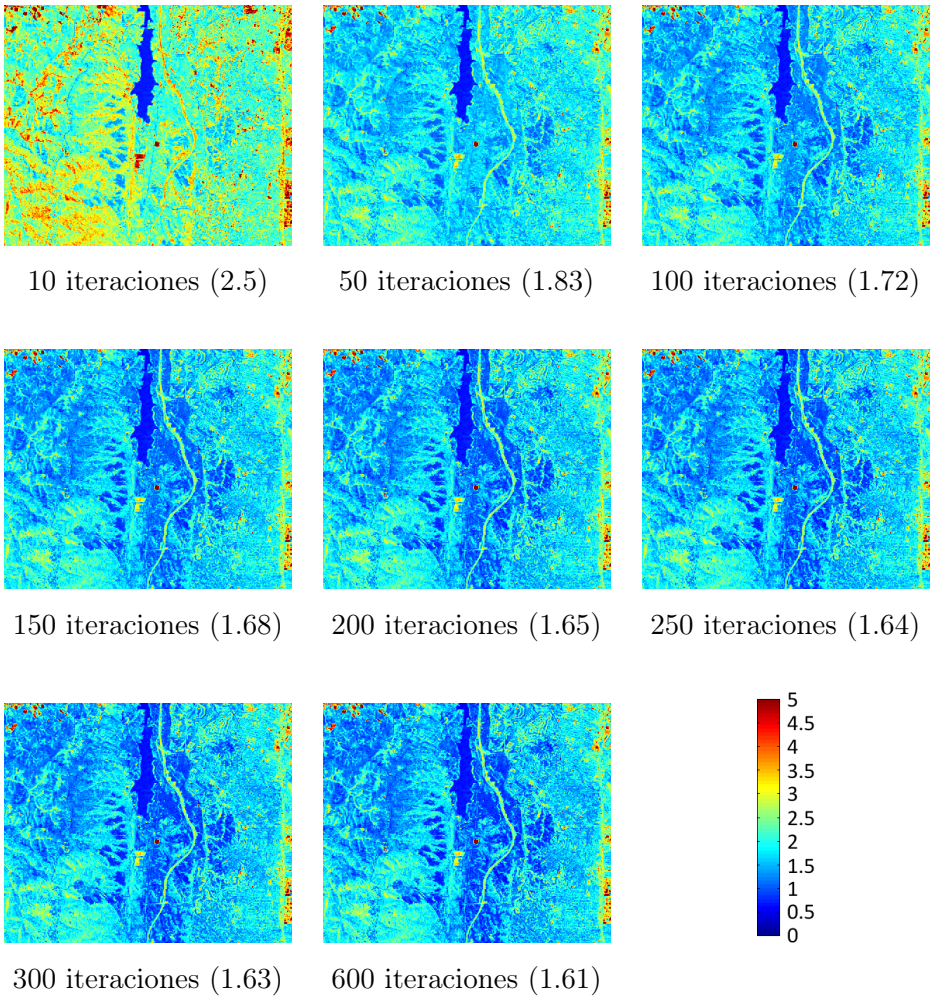


Figura 7.7: Mapa de errores RECM (en porcentaje) para distinto número de iteraciones después de la reconstrucción de la escena AVIRIS Jasper Ridge en unidades de reflectancia con los endmembers extraídos por el algoritmo PPI en la escena en unidades de radiancia.

Tabla 7.4: Tiempos de procesamiento para la implementación hardware de desmezclado espectral para las imágenes hiperespectrales consideradas en la FPGA Virtex-4 XC4VFX60.

	AVIRIS Cuprite	**AVIRIS Jasper Ridge**
Extracción de endmembers	16.86 seg	43.26 seg
Eliminación de redundantes	2.52 seg	3.55 seg
Reconfiguración total	694 ms	694 ms
Estimación de abundancias	1.59 min	4.10 min
Total	**1.92 min**	**4.89 min**

Capítulo 8

Conclusiones y trabajo futuro

Son los problemas sin resolver, no los resueltos, los que mantienen activa la mente

Erwin Guido Kolbenheyer

8.1. Conclusiones

La evolución de la observación remota de la Tierra ha sido testigo de la obtención de mediciones con una amplitud espectral y espacial cada vez mayor. Este incremento, ha estado motivado por un deseo de extraer cada vez más información sobre las propiedades de los materiales residentes en los píxeles de una escena tanto en aplicaciones civiles como militares. Si bien la detección multiespectral ha logrado en gran medida la clasificación de píxeles completos, el análisis de los materiales que forman un píxel se ve limitado por un número relativamente bajo de mediciones espectrales. El conocimiento de que los píxeles de interés suelen ser una combinación de numerosos materiales diferentes ha introducido la necesidad de descomponer cuantitativamente estas mezclas. Mediante la recopilación de datos en cientos de bandas espectrales, los sensores hiperespectrales propocionan la información necesaria para realizar el análisis de mezclas espectrales.

En las imágenes hiperespectrales, los píxeles mezcla son una combinación

de varios materiales diferentes, y existen por alguna de estas dos razones. En primer lugar, si la resolución espacial del sensor no es lo suficientemente alta, distintos materiales puedan ocupar conjuntamente un solo píxel y entonces el resultado de la medición espectral será una composición de los espectros de estos materiales individuales. Este caso se produce cuando las plataformas de observación remota de la Tierra vuelan a gran altitud o realizan la vigilancia de un área amplia, donde la resolución espacial normalmente es baja. En segundo lugar, los píxeles mezcla aparecen cuando se combinan distintos materiales en una mezcla homogénea. Esta circunstancia puede ocurrir independientemente de la resolución espacial del sensor.

El desmezclado espectral es el procedimiento mediante el cual se descompone el espectro medido de un píxel mezcla en una colección de espectros constituyentes, o *endmembers*, y un conjunto de fracciones correspondientes, o *abundancias*, que indican la proporción de cada endmember presente en el píxel. Los endmembers normalmente se corresponden con objetos macroscópicos conocidos en la escena, como agua, suelo, metal, vegetación, etc. En términos generales, el desmezclado es un caso especial del problema de inversión generalizado en el que los parámetros estimados describen un objeto mediante la observación de una señal que ha interactuado con el objeto antes de llegar al sensor. En el caso de imágenes hiperespectrales en régimen de reflexión, e ignorando los efectos atmosféricos, la señal incidente es la radiación electromagnética procedente del sol, que es medida por el sensor después de que la radiación se refleje hacia arriba por los materiales naturales y los fabricados por el hombre en la superficie de la Tierra.

Los modelos analíticos para la mezcla de diferentes materiales proporcionan la base del desarrollo de técnicas para recuperar las estimaciones de los espectros de los materiales constituyentes y sus proporciones en píxeles mezcla. La premisa básica del modelo de mezcla es que dentro de una escena determinada, la superficie está compuesta por un pequeño número de materiales diferentes que tienen propiedades espectrales relativamente constantes. Estas sustancias distintas (por ejemplo, el agua, el pasto o los tipos de minerales) se denominan endmembers, y las fracciones en las que aparecen en un píxel mezcla se llaman fracciones de abundancia. Si la mayor parte de la variabilidad espectral dentro de una escena es una consecuencia

de que los endmembers aparezcan en proporciones variables, lógicamente se deduce que una combinación de sus propiedades espectrales puede modelar la variabilidad espectral observada por el sistema de teledetección.

Habitualmente, la superficie reflectante es considerada como una mezcla en forma de franjas, y cada uno de los paquetes de radiación incidente interactúa sólo con un componente (es decir, no hay dispersión múltiple entre los componentes). Si se considera que la superficie total se divide proporcionalmente de acuerdo a la fracción de abundancia de cada endmember, entonces la radiación reflejada transmitirá las características de los materiales asociados en las mismas proporciones. En este sentido, existe una relación lineal entre las fracciones de abundancia de los materiales que comprenden el área que está siendo fotografiado y el espectro de la radiación reflejada.

Por lo tanto, este modelo se conoce con el nombre de modelo lineal de mezcla. Generalmente, en el modelo lineal de mezcla se imponen dos restricciones sobre las fracciones de abundancia. Para que tengan un significado físico, la condición de no negatividad impone que todas las fracciones de abundancia sean positivas. En segundo lugar, como forma de dar idea de toda la composición de un píxel mezcla, se impone la restricción de suma unitaria.

La alta dimensionalidad de los datos hiperespectrales provoca que la ejecución de este tipo de algoritmos sea lenta. Hasta la fecha, las técnicas tradicionales para abordar este problema han optado por soluciones basadas en el uso de clusters y sistemas multiprocesador para realizar computación de alta prestaciones. Sin embargo, ambas soluciones suponen un *payload* excesivo para el procesamiento a bordo en plataformas de observación remota de la Tierra. La reconfiguración, el tamaño compacto y la elevada potencia de cálculo de las Field Programmable Gate Arrays (FPGAs), las hacen particularmente atractivas para su explotación en aplicaciones de observación remota de la Tierra que requieran respuesta en tiempo real.

Uno de los retos más importantes que debe abordarse en el análisis de imágenes hiperespectrales es la complejidad computacional de los algoritmos derivada de la cada vez mayor resolución espacial, espectral y temporal de las nuevas misiones hiperespectrales de observación remota de la Tierra,

muchas de ellas basadas en plataformas espaciales. El procesado a bordo de imágenes hiperespectrales ha sido un objetivo muy perseguido en teledetección. Con el trabajo desarrollado en esta tesis se pretenden demostar las ventajas del procesamiento a bordo de imágenes hiperespectrales mediante hardware reconfigurable. Para demostrar esta utilidad se han estudiado, diseñado, implementado y validado dos algoritmos para la extracción de endmembers (pixel purity index –PPI– y N–FINDR) y uno para la estimación de abundancias (Image Space Reconstruction Algorithm –ISRA–), con el fin de proporcionar una cadena completa de desmezclado espectral en FPGA que haga frente a los dos principales retos de las imágenes hiperespectrales: la alta dimensionalidad de los datos y el problema de la mezcla espectral.

Hasta el momento, las implementaciones disponibles en la literatura del algoritmo PPI en FPGA habían demostrado la eficacia de una implementación hardware en una placa reconfigurable, pero estas soluciones resultaban muy dependientes de la placa utilizada y no eran escalables. La implementación en FPGA propuesta tiene por objetivo superar estos inconvenientes. En primer lugar, la arquitectura especificada se puede adaptar fácilmente a diferentes plataformas. En segundo lugar, la arquitectura propuesta es escalable en función de la cantidad de recursos disponibles, ya que los recursos necesarios crecen proporcionalmente con el número de skewers y el ciclo de reloj permanece constante. Además, se proporciona un estudio detallado sobre las distintas estrategias de paralelización posibles con el fin de establecer las bases del diseño más adecuado para el algoritmo PPI en FPGA. Los resultados experimentales, llevados a cabo en las FPGAs Xilinx Virtex–II Pro XC2V2P30 y Virtex–4 XC4VFX60 (plataformas con las mismas arquitecturas y áreas similares a FPGAs resistentes a la radiación que han sido certificadas por organismos internacionales de teledetección y que son utilizadas habitualmente en plataformas de observación remota de la Tierra) demuestran que la implementación hardware propuesta hace un uso apropiado de los recursos disponibles en la arquitectura considerada y es capaz de proporcionar endmembers precisos. Además, la versión hardware propuesta del algoritmo PPI supera de manera significativa (en términos de tiempo de cálculo) una versión software equivalente desarrollada en lenguaje C, la versión original semisupervisada (disponible en software comercial) y una implementación en FPGA recientemente desarrollada en una Xilinx Virtex–II

XC2V6000–6.

En este trabajo de investigación, se ha desarrollado la primera implementación en FPGA del algoritmo N–FINDR para la extracción de endmembers, que ilustra las ventajas y desventajas de la tecnología reconfigurable en el contexto de las misiones aéreas y espaciales de observación remota de la Tierra. Tras un minucioso estudio del algoritmo y de las propiedades de los determinantes, se establece el método de triangulación de matrices como adecuado para el cálculo del volumen del simplex formado por los endmembers seleccionados. Los resultados experimentales, realizados de nuevo en la FPGA Xilinx Virtex–4 XC4VFX60 demuestran que la implementación hardware propuesta obtiene un alto rendimiento con un *speedup* de 37 comparado con una versión software equivalente y también es capaz de proporcionar resultados precisos.

Una vez que se han identificados los endmembers de la escena, es el turno de determinar sus proporciones en cada uno de los píxeles. En este trabajo de tesis, se ha descrito una implementación en FPGA del Image Space Reconstruction Algorithm (ISRA), uno de los algoritmo iterativos más conocidos para la estimación de fracciones de abundancia en el análisis de datos hiperespectrales en observación remota de la Tierra. Este algoritmo sigue un modelo lineal de mezcla para la estimación de las fracciones de abundancia y puede ser visto como un problema de minimización de distancia de mínimos cuadrados con la restricción de abundancias no negativas. Este trabajo demuestra una vez más la viabilidad de la utilización de FPGAs en el análisis de imágenes hiperespectrales especialmente con algoritmos de cálculo iterativo o intensivo. En los experimentos realizados, se ha variado el número de iteraciones para estudiar la convergencia del algoritmo, estableciendo 100 como el número adecuado de iteraciones. Además de la precisión en los resultados y del rendimiento obtenido, otra característica interesante de la implementación propuesta es que es fácilmente escalable para FPGAs mayores.

Una de las técnicas más importante para el análisis de datos hiperespectrales es el desmezclado espectral. En este trabajo nos valemos de la reconfigurabilidad de las FPGAs para llevar a cabo la implementación de la cadena completa de desmezclado espectral en FPGA. Se ha realizado una

comparativa entre los algoritmos PPI y N–FINDR para mostrar las ventajas e inconvenientes de ambos y así, establecer cual de los dos es más adecuado para la realización de la etapa de extracción de endmembers. Los resultados experimentales, demuestran que la penalización impuesta por la reconfiguración (principal problema en los sistemas multitarea reconfigurables) supone una penalización muy poco significativa en comparación con el tiempo de ejecución de los algoritmos en el contexto del análisis de imágenes hiperespectrales. Además, ofrece un incremento del rendimiento significativo comparado con una versión software equivalente y también es capaz de proporcionar resultados precisos con un tamaño compacto, que hacen del sistema reconfigurable propuesto atractivo para el procesamiento a bordo de datos hiperespectrales.

8.2. Trabajo futuro

A pesar de que los tiempos de procesamiento todavía están lejos del rendimiento de un sensor hiperespectral, los nuevos avances, tanto en el diseño de algoritmos como en la disponibilidad de mejor hardware, nos llevan a creer que el objetivo del procesamiento a bordo de imágenes hiperespectrales en el espacio, a medida que se capturan, se llevará a cabo en un futuro próximo. Por ejemplo, si dispusiéramos de una FPGA Virtex–4 XQR4VLX200 (89088 slices), certificada para el espacio, podríamos disponer de 3.5 veces más recursos hardware que en la FPGA Virtex–4 XC4VFX60 utilizada en este trabajo, y gracias a la escalabilidad de las implementaciones de los algoritmos desarrollados, se podría alcanzar un *speedup* de 3.5 sin realizar modificaciones en el diseño. En el caso de una plataforma aerotransportada, se podría utilizar una FPGA Virtex–6 XQ6VLX550T (550000 logics cells) que dispone de casi 10 veces más celdas lógicas que la FPGA utilizada.

Con respecto a los avances en el diseño de algoritmos, a continuación se recogen varios aspectos en los que se pueden mejorar los algoritmos propuestos junto con las consideraciones sobre la cadena óptima de desmezclado.

Con respecto al algoritmo PPI, en el estudio de las distintas estrategias de paralelización se contemplan dos dimensiones: la paralelización por píxeles y la paralelización por skewers. Una vez demostradas las ventajas de utilizar

la paralelización por skewers, queremos estudiar una tercera dimensión: la paralelización por bandas. Esto significa que se realizará la multiplicación (en nuestra implementación la suma o la resta) de más de un componente del píxel con los correspondientes del skewer en un sólo ciclo de reloj. Deberemos estudiar si compensa el incremento en la complejidad (en cuanto a consumo de recursos) de la unidad básica, lo que provocará que tengamos un menor número de unidades en paralelo, y el descenso en la frecuencia de reloj, con respecto a calcular el dot–product en un menor número de ciclos de reloj. Otro aspecto muy importante a considerar es que la entrada de datos no se convierta en el cuello de botella del sistema degradando el rendimiento. Por otra parte, también nos centraremos en la mejora de la implementación propuesta para lograr un mejor aprovechamiento de los recursos hardware y reducir los tiempos de procesamiento, que en cualquier caso, se consideran aceptables en muchas aplicaciones de observación remota de la Tierra.

A pesar de que no suponga una fracción muy significativa del tiempo total de ejecución de la cadena de desmezclado espectral, se abordará la implementación del paso de eliminación de endmembers redundantes en FPGA aprovechándonos de su reconfigurabilidad.

Por otra parte, se debe investigar sobre la conveniencia o no de realizar el paso previo de reducción dimensional a la hora de ejecutar el algoritmo PPI y qué tipo de reducción es la más adecuada. Se pueden encontrar diversos trabajos en la literatura que abordan la cuestión de la reducción dimensional en FPGA. Algunos de estos trabajo se basan en el Análisis de Componentes Principales [ZCS08, DMJ$^+$08, DQ04, NMC06] entre los que destaca la Tesis Doctoral realizada por Ignacio Bravo [Bra07] en la Universidad de Alcalá. El trabajo de investigación citado destaca principalmente porque permite escalar el problema a un mayor número de variables (las bandas espectrales en nuestro caso) y es independiente, en lo que a arquitectura hardware se refiere, del número de individuos (píxeles en nuestro caso) de la muestra. El resto de implementaciones presentan limitaciones en cuanto al número de variables (se utilizan estructuras fijas para calcular un número determinado de autovalores y autovectores) y/o en cuanto al número de individuos (tamaño de la escena). Otra de las técnicas empleadas para la reducción dimensional de los datos hiperespectrales es el Análisis de Componentes Independientes (ACI),

del que existen diversas implementaciones en FPGA [NHS01, DQ04, DQ06]. ACI es una técnica que extrae las fuentes de señal independientes buscando una transformación lineal o no lineal que minimice la dependencia estadística entre componentes. ICA se ha utilizado en variedad de aplicaciones de procesado de señal incluida la reducción dimensional en el análisis de imágenes hiperespectrales.

Con respecto al algoritmo N-FINDR, uno de los primeros aspectos que debemos solventar es la necesidad de determinar el número de endmembers a generar. Como trabajo futuro, estamos investigando las implementaciones en FPGA de técnicas para estimar el número de endmembers en la escena, como el concepto de dimensionalidad virtual [DC04]. También estamos investigando otros aspectos, como la posibilidad de almacenar diferentes implementaciones del algoritmo N–FINDR optimizadas para diferente número de endmembers (con el fin de seleccionar adaptativamente el más adecuado según la escena), además de realizar un análisis sobre el impacto de los endmembers iniciales aleatorios sobre el algoritmo N–FINDR (en el futuro podrían aplicarse estrategias más inteligentes para la inicialización). Como trabajo futuro también se tiene en mente realizar una comparativa entre nuestra implementación hardware del método de triangulación de matrices con otras técnicas disponibles en la literatura que utilizan el mismo método en hardware mediante la descomposición en valores singulares (SVD) [BDL03, AAB03], con el fin de mejorar el rendimiento.

Con respecto al ISRA, nos plateamos modificar el algoritmo para que el número de iteraciones por píxel en el análisis no sea siempre un valor preestablecido e igual para todos los píxeles que componen la imagen hiperespectral. Para ello, debemos tener en cuenta tanto la convergencia del algoritmo como lo próximos que están el píxel medido y estimado. Un pseudocódigo general para ISRA, teniendo en cuenta los aspectos que se acaban de mencionar, podría ser el que se muestra a continuación:

1. Inicializar $\hat{\mathbf{x}}^0$, ϵ y ρ
2. while(error $\geq \epsilon$) and (stop $\geq \rho$) do
   a) estimar $\hat{\mathbf{x}}$ utilizando la Ecuación 6.6

$b)$ error $= \left\| \mathbf{A}\hat{\mathbf{x}}^k - \mathbf{b} \right\|^2$

$c)$ stop $= \left\| \hat{\mathbf{x}}^{k+1} - \hat{\mathbf{x}}^k \right\| / \left\| \hat{\mathbf{x}}^k \right\|$

3. return $\hat{\mathbf{x}}$

El pseudocódigo asume que $\hat{\mathbf{x}}$ es un vector con valores no negativos, al igual que la matriz de endmembers $\mathbf{A}$. El algoritmo finalizará su ejecución cuando el error cometido sea menor que ϵ o cuando la abundancia estimada actual sea muy similar a la abundancia estimada anterior, es decir, $\hat{\mathbf{x}}^{k+1}$ sea muy similar a $\hat{\mathbf{x}}^k$), determinado por ρ. La convergencia de este algoritmo ha sido comprobada en [VRRPL03] y en otros trabajos, además, si el sistema $\mathbf{Ax} = \mathbf{b}$ tiene solución (sistema consistente) el algoritmo converge a la solución pero en general cuando hay ruido presente en el sistema $\mathbf{Ax} \neq \mathbf{b}$ (sistema inconsistente) el algoritmo converge a la mínima distancia **MC(Ax,b)**.

También se pretende estudiar las formas en que se puede acelerar la convergencia del algoritmo. El ratio de convergencia de ISRA es lento como muestra las Figuras 6.8, 6.9 y 6.10. Se pueden utilizar diferentes métodos para acelerar el ratio de convergencia del algoritmo. En el trabajo de Meidunas [DWM86], se utiliza el Block Iterative Method para acelerar el ratio de convergencia del algoritmo Expectation Maximization Maximum Likelihood (EMML). Esa misma idea de relajación se podría utilizar para acelerar la convergencia del ISRA, dando lugar al algoritmo iterativo ISRA acelerado cuya base es:

$$\hat{x}_j^{k+1} = (1 - w)\, \hat{x}_j^k + w\, \hat{\mathbf{x}}_{ISRA} \tag{8.1}$$

Donde $\hat{\mathbf{x}}_{ISRA}$ es la estimación básica ISRA y w es el parámetro de relajación.

Finalmente, pretendemos aprovecharnos de la correspondencia espectral entre píxeles espacialmente próximos, partiendo como estimación inicial de las fracciones de abundancia estimadas anteriormente para un píxel vecino. Se debe tener en cuenta que para el ISRA si alguna de las fracciones de abundancia alcanza el valor cero, permanecerá con el valor cero.

Dado que el proceso de corrección atmosférica también se puede aplicar una vez que se han identificado las posiciones de los píxeles endmembers, nuestra tesis es que el proceso de extracción de endmembers se puede realizar según los datos se recogen en el sensor, sin la necesidad de tener un módulo de corrección atmosférica anterior, que también debería ser implementado en hardware con el fin de cumplir plenamente con los requisitos de tiempo real. A pesar de que los resultados experimentales realizados así lo demuestran, no consideramos que sean concluyentes y por lo tanto, este tema continuará siendo objeto de investigación en nuestro trabajo futuro.

Por último, también estamos evaluando otras plataformas hardware para la explotación a bordo de datos hiperespectrales, tales como unidades de procesamiento gráfico (GPU). Cabe señalar que las GPUs representan una alternativa interesante para la implementación en tiempo real, pero este tipo de hardware todavía no está certificado para operar en el espacio debido principalmente a su alto consumo de energía en comparación con las FPGAs.

Publicaciones generadas

Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.

Mahatma Gandhi

Se presentan a continuación las publicaciones mediante las que se ha divulgado el trabajo de investigación realizado. Están divididas en publicaciones en revistas con índice de impacto, artículos en congresos internacionales y artículos en congresos nacionales, ordenadas siguiendo un orden cronológico inverso.

Publicaciones en revistas con índice de impacto:

- Carlos González, Daniel Mozos, Javier Resano y Antonio Plaza. "FPGA Implementation of the N-FINDR Algorithm for Remotely Sensed Hyperspectral Image Analysis". *IEEE Transactions on Geoscience and Remote Sensing*. En proceso de revisión con cambios menores.

- Juan Antonio Clemente, Javier Resano, Carlos González y Daniel Mozos. "A Hardware Implementation of a Run-Time Scheduler for Reconfigurable Systems". *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*. Pendiente de publicación.

- Carlos González, Javier Resano, Daniel Mozos, Antonio Plaza y David Valencia. "FPGA Implementation of the Pixel Purity Index Algorithm for Remotely Sensed Hyperspectral Image Analysis". *EURASIP Journal on Advances in Signal Processing*. Volumen 2010, páginas 1-13. Mayo de 2010.

- Juan Antonio Clemente, Carlos González, Javier Resano y Daniel Mozos. "A Task-Graph Execution Manager for Reconfigurable Multi-tasking Systems". *Microprocessors and Microsystems*. Volumen 34, páginas 73-83. Marzo de 2010.

- Javier Resano, Juan Antonio Clemente, Carlos González, Daniel Mozos y Francky Catthoor. "Efficiently scheduling run-time reconfigurations". *ACM Transactions on Design Automation of Electronic Systems (TODAES)*. Volumen 13, páginas 58-69. Septiembre de 2008.

Artículos publicados en congresos internacionales:

- Carlos González, Javier Resano, Daniel Mozos y Antonio Plaza. "FPGA Design of the N-FINDR Algorithm for Endmember Extraction from Hyperspectral Images". *2nd International Conference on Space Technology*. Pendiente de publicación.

- Javier Olivito, Carlos González y Javier Resano. "FPGA Implementation of a Strong Reversi Player". *IEEE International Conference on Field Programmable Technology (FPT)*. Páginas 507-510. 8-10 de diciembre de 2010.

- Carlos González, Daniel Mozos, Javier Resano y Antonio Plaza. "FPGA for Computing the Pixel Purity Index Algorithm on Hyperspectral Image". *The International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*. Páginas 293-296. 12-15 de julio de 2010.

- Carlos González, Javier Olivito y Javier Resano. "An Initial Specific Processor for Sudoku Solving". *IEEE International Conference on Field Programmable Technology (FPT)*. Páginas 530-533. 9-11 de diciembre de 2009.

- Carlos González, Javier Resano y Daniel Mozos. "FPGA Support for Satellite Computations of Hyper Spectral Images". *International Conference on Field Programmable Logic (FPL)*. Páginas 715-716. 31 de agosto a 2 de septiembre de 2009.

- Carlos González, Javier Resano y Daniel Mozos. "FPGA pre-processing of hyper spectral images: A preliminary approach". *Fifth International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES)*. 12-18 de julio de 2009.

- Juan Antonio Clemente, Carlos González, Javier Resano y Daniel Mozos. "A Hardware Task-Graph Scheduler for Reconfigurable Multi-tasking Systems". *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. Páginas 79-84. 3-5 de diciembre de 2008.

- Juan Antonio Clemente, Carlos González, Javier Resano y Daniel Mozos. "Task-graph management for reconfigurable multi-tasking systems". *International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoc)*. Páginas 192-193. 9-11 de julio de 2008.

- Javier Resano, Juan Antonio Clemente, Carlos González, José Luis García y Daniel Mozos. "HW implementation of an execution manager for reconfigurable systems". *The International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*. Páginas 71-77. 25-28 de junio de 2007.

Artículos publicados en congresos nacionales:

- Carlos González, Daniel Mozos, Javier Resano y Antonio Plaza. "Implementación en FPGA del algoritmo Pixel Purity Index para el análisis de imágenes hiperespectrales en teledetección". *X Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA) dentro del III Congreso Español de la Informática (CEDI)*. 7-10 de septiembre de 2010.

- Javier Olivito, Carlos González y Javier Resano. "Versión preliminar de un procesador específico para la resolución de Sudokus". *X Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA) dentro del III Congreso Español de la Informática (CEDI)*. 7-10 de septiembre de 2010.

- Juan Antonio Clemente, Carlos González, Javier Resano y Daniel Mozos. "Implementaciones HW y SW de un gestor de ejecución de grafos de tareas en un sistema multitarea reconfigurable". *VIII Jornadas sobre*

Computación Reconfigurable y Aplicaciones (JCRA). Páginas 225-236. 18-19 de septiembre de 2008.

- Carlos González, Juan Antonio Clemente, José Luis García, Javier Resano y Daniel Mozos. "Un sistema para la gestión eficiente del HW reconfigurable". *VII Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA) dentro del II Congreso Español de la Informática (CEDI)*. Páginas 163-170. 11-14 de septiembre de 2007.

Bibliografía

Cuando bebas agua, recuerda la fuente.

Proverbio chino

[AAB03] A. Ahmedsaid, A. Amira, and A. Bouridane. Improved SVD systolic array and implementation on FPGA. *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT'03)*, 1:35–42, 2003.

[Act97] Actel. Accelerator Series FPGAs: ACT3 Family. *http://www.actel.com*, 1997.

[Act01] Actel. SX Family of High Performance FPGAs. *http://www.actel.com*, 2001.

[Act05] Actel. Flash FPGAs in the value-based market. *A White Paper*, 2005.

[AGG+90] M. Ahrens, A. El Gamal, D. Galbraith, J. Greene, and S. Kaptanoglu. An FPGA family optimized for high densities and reduced routing delay. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 31.5.1–31.5.4, 1990.

[Alt05] Altera. Available online: http://www.altera.co.jp/literature/sg/sg_ip.pdf. 2005.

[Alt10] Altera. Stratix V Field Programmable Gate Array. *http://www.altera.com/literature/br/br-stratix-v-hardcopy-v.pdf*, 2010.

[And99] T. Anderson. System-on-Chip Design with Virtual Components. *Circuit Cellar Online, http://www.circuitcellar.com*, 199.

[Arn05] J. M. Arnold. S5: the architecture and development flow of a software configurable processor. *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT '05)*, pages 121–128, 2005.

[ASI+98] A. Abnous, K. Seno, Y. Ichikawa, M. Wan, and J. Rabaey. Evaluation of a low-power reconfigurable DSP architecture. *Proceedings of the 5th Reconfigurable Architectures Workshop (RAW '98)*, 89:55–60, 1998.

[ASJ86] J. B. Adams, M. O. Smith, and P. E. Johnson. Spectral mixture modeling: A new analysis of rock and soil types at the Viking Lander 1 site. *Journal of Geophysical Research*, 91:8098–8112, 1986.

[BBY07] J. E. Ball, L. M. Bruce, and N. Younan. Hyperspectral pixel unmixing via spectral band selection and dc-insensitive singular value decomposition. *IEEE Geoscience and Remote Sensing Letters*, 4(3):382–386, 2007.

[BC96] C. A. Bateson and B. Curtiss. A Method for Manual Endmember Selection and Spectral Unmixing. *Remote Sensing of Environment*, 55(3):229–243, 1996.

[BDH+97] J. Burns, A. Donlin, J. Hogg, S. Singh, and M. De Wit. A Dynamic Reconfiguration Run-Time System. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'97)*, pages 66–76, 1997.

[BDIE99] E. Ben-Dor, J. R. Irons, and G .F. Epema. Soil Reflectance. *Manual of Remote Sensing, Remote Sensing for the Earth Sciences*, 3:111–188, 1999.

[BDL03] C. Bobda, K. Danne, and A. Linarth. Efficient implementation of the singular value decomposition on a reconfigurable

system. *Lecture Notes in Computer Science*, 2778:1123–1126, 2003.

[BG94] C. C. Borel and S. A. W. Gerslt. Nonlinear spectral mixing models for vegetative and soil surfaces. *Remote Sensing of Environment*, 47:403–416, 1994.

[BKG95] J. Boardman, F. Kruse, and R. Green. Mapping target signatures via partial unmixing of aviris data. *Proceedings of Summaries JPL Airborne Earth Sci. Workshop*, pages 23–26, 1995.

[Boa93] J. Boardman. Automating spectral unmixing of AVIRIS data using convex geometry concepts. *Summaries of Airborne Earth Science Workshop, JPL Publication*, JPL Publication 93–26:114–114, 1993.

[Boa94] J. W. Boardman. Geometric Mixture Analysis of Imaging Sectrometry Data. *Geoscience and Remote Sensing Symposium*, 4:2369–2371, 1994.

[BP02] K. Bondalapati and V.K. Prasanna. Reconfigurable Computing Systems. *Proceedings of the IEEE*, 90(7):1201–1217, 2002.

[Bra01] T. Branca. How to Add Features and Fix Bugs - Remotely. Here's What You Need to Consider When Designing a Xilinx Online Application. *Xilinx*, 2001.

[Bra07] Ignacio Bravo. *Arquitectura basada en FPGAs para la detección de objetos en movimiento, utilizando visión computacional y técnicas PCA.* Tesis doctoral, Universidad de Alcalá, 2007.

[Bre96] G. Brebner. A virtual hardware operating system for the Xilinx XC6200. *Proceedings of the 6th InternationalWorkshop on Field-Programmable Logic and Applications (FPL '96)*, pages 327–336, 1996.

[BTAS07] S. Bhunia, M. Tabib-Azar, and D. Saab. Ultralow–Power Reconfigurable Computing with Complementary Nano–Electromechanical Carbon Nanotube Switches. *Proceedings of ASP-DAC*, pages 86–91, 2007.

[CDSA99] C.-I Chang, Q. Du, T. Sun, and L. G. Althouse. A Joint Band Prioritization and Band-Decorrelation Approach to Band Selection for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 37:2631–2641, 1999.

[CGRM08] J. A. Clemente, C. González, J. Resano, and D. Mozos. A hardware task-graph scheduler for reconfigurable multi-tasking systems. *International Conference on Reconfigurable Computing and FPGAs*, 0:79–84, 2008.

[CGRM10] J. A. Clemente, C. González, J. Resano, and D. Mozos. A task graph execution manager for reconfigurable multi-tasking systems. *Microprocessors and Microsystems*, 34(2-4):73–83, 2010.

[CH00] C.-I Chang and D. Heinz. Constrained subpixel target detection for remotely sensed imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 38:1144–1159, 2000.

[CH02] K. Compton and S. Hauck. An Iterative Image Space Reconstruction Algorithm Suitable for Volume ECT. *ACM Computing Surveys*, 34(2):171–210, 2002.

[Cha03] C.-I Chang. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification.* Kluwer Academic/Plenum Publishers: New York, 2003.

[Cha07] C.-I Chang. *Hyperspectral Data Exploitation: Theory and Applications.* John Wiley & Sons: New York, 2007.

[Che99] J. M. Chen. Spatial Scaling of a Remotely Sensed Surface Parameter by Contexture. *Remote Sensing of Environment*, 69:30–42, 1999.

[Cla99] R. M. Clark. Spectroscopy of Rocks and Minerals, and Principles of Spectroscopy. *Manual of Remote Sensing, Remote Sensing for the Earth Sciences*, 3:3–58, 1999.

[CLC+02] K. Compton, Z. Li, J. Cooey, S. Knol, and S. Hauck. Configuration Relocation and Defragmentation for Run-Time Reconfigurable Computing. *IEEE Transactions on VLSI Systems*, 10(3):209–220, 2002.

[CP06] C.-I Chang and A. Plaza. A fast iterative algorithm for implementation of pixel purity index. *IEEE Geoscience and Remote Sensing Letters*, 3(1):63–67, 2006.

[CPSS00] Y. Chou, P. Pillai, H. Schmit, and J.P. Shen. PipeRench Implementation of the Instruction Path Coprocessor. *Proceedigns of the 33th Annual International Symposium on Microarchitecture (MICRO)*, 2778:147–158, 2000.

[CRGM10] J. A. Clemente, J. Resano, C. González, and D. Mozos. A Hardware Implementation of a Run-Time Scheduler for Reconfigurable Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, (99):1–14, 2010.

[DC04] Q. Du and C.-I Chang. Estimation of number of spectrally distinct signal sources in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 42(3):608–619, 2004.

[DeH96] A. DeHon. Reconfigurable Architectures for General Purpose Computing. *A.I. Technical report No. 1586. Artificial Intelligence Laboratory. Massachusetts Institute of Technology*, 1996.

[DMJ+08] Abhishek Das, Sanchit Misra, Sumeet Joshi, Joseph Zambreno, Gokhan Memik, and Alok Choudhary. An efficient fpga implementation of principle component analysis based network intrusion detection system. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '08, pages 1160–1165, New York, NY, USA, 2008. ACM.

[DQ04] Hongtao Du and Hairong Qi. An fpga implementation of parallel ica for dimensionality reduction in hyperspectral images. In *Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 IEEE International*, volume 5, pages 3257 – 3260 vol.5, 2004.

[DQ06] Hongtao Du and Hairong Qi. A reconfigurable FPGA system for parallel independent component analysis. *EURASIP Journal on Embedded Systems*, pages 1–13, 2006.

[DWM86] M.E. Daube-Witherspoon and G. Muehllehner. An Iterative Image Space Reconstruction Algorithm Suitable for Volume ECT. *IEEE Transactions on Medical Imaging*, 1986.

[DWM+05] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon. Demystifying 3D ICs: the Pros and Cons of Going Vertical. *IEEE Design and Test of Computers*, 22(6):498–510, 2005.

[FBC+03] M. Fischman, A. Berkun, F. Cheng, W. Chun, E. Im, and R. Andraka. Design and demostration of an advanced onboard processor for the second-generation precipitation radar. *Proceedings of the IEEE Aerospace Conference*, 2:1067–1075, 2003.

[FP01] M. Faraklioti and M. Petrou. Illumination invariant unmixing of sets of mixed pixels. *IEEE Transactions on Geoscience and Remote Sensing*, 39:2227–2234, 2001.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[GAC01] K. J. Guilfoyle, M. L. Althouse, and C.-I Chang. A quantitative and comparative analysis of linear and nonlinear spectral mixture models using radial basis function neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 39:2314–2318, 2001.

[GB97] A. F. Goetz and J. W. Boardman. Atmospheric Corrections: On Deriving Surface Reflectance from Hyperspectral Imagers. *Imaging Spectrometry III: Proceedings of SPIE*, pages 14–22, 1997.

[GBSC88] A. A. Green, M. Berman, P. Switzer, and M. D. Craig. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE Transactions on Geoscience and Remote Sensing*, 26:65–74, 1988.

[GCL02] J. Gause, P.Y.K. Cheung, and W. Luk. Reconfigurable Shape-Adaptive Template Matching Architectures. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'97)*, pages 98–110, 2002.

[GES+98] R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis, Ma. R. Olah, and O. Williams. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sensing of Environment*, 65(3):227–248, 1998.

[GHS02] F. J. García-Haro and S. Sommer. A fast canopy reflectance model to simulate realistic remote sensing scenarios. *Remote Sensing of Environment*, 81:205–227, 2002.

[GK99] A. F. Goetz and B. Kindel. Comparison of unmixing result derived from aviris, high and low resolution, and hydice images at cuprite, nv. *Proceedings of IX NASA/JPL Airborne Earth Science Workshop*, 1999.

[Gor06] M. Goretti. Digital circuits based on FPGAs for random number generation. *Tech. Rep., Department of Electricity and Electronics, University of Basque Country*, 2006.

[GOR09] C. González, J. Olivito, and J. Resano. An initial specific processor for sudoku solving. *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT '09)*, pages 530 –533, 2009.

[GP00] R. O. Green and B. Pavri. Aviris in-flight calibration experiment, sensitivity analysis, and intraflight stability. *Proceedings of IX NASA/JPL Airborne Earth Science Workshop*, 2000.

[GRM+10] C. González, J. Resano, D. Mozos, A. Plaza, and D. Valencia. FPGA implementation of the pixel purity index algorithm for remotely sensed hyperspectral image analysis. *EURASIP Journal on Advances in Signal Processing*, 969806:1–13, 2010.

[GU01] M. Garcia and S. L. Ustin. Detection of interannual vegetation responses to climatic variability using aviris data in a coastal savanna in california. *IEEE Transactions on Geoscience and Remote Sensing*, 39:1480–1490, 2001.

[GVPR04] B. Griese, E. Vonnahme, M. Porrmann, and U. Ruckert. Hardware support for dynamic reconfiguration in reconfigurable SoC architectures. *Proceedings of the 14th International Conference on Field-Programmable Logic and Applications (FPL '04)*, pages 842–846, 2004.

[GVSR85] A. F. Goetz, G. Vane, J. E. Solomon, and B. N. Rock. Imaging spectrometry for Earth remote sensing. *Science*, 228:1147–1153, 1985.

[Hau98] S. Hauck. Configuration Prefetch for Single Context Reconfigurable Coprocessors. *Proceedigns of the ACM International Symposium in Field Programmable Gate Arrays (FPGA'98)*, pages 65–74, 1998.

[HC00] D. Heinz and C.-I Chang. Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39:529–545, 2000.

[HCJ+90] H. Hsieh, W. Carter, J. Y. Ja, E. Cheung, S. Schreifels, C. Erickson, P. Freidin, and L. Tinkey. Third-generation Architecture Boosts Speed and Density of Field-programmable Gate arrays. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 31.2.1–31.2.7, 1990.

[HFHK04] S. Hauck, T.W. Fry, M.M. Hosler, and J.P. Kao. The Chimaera Reconfigurable Functional Unit. *IEEE Transactions on VLSI Systems*, 12(2):206–217, 2004.

[HLS98] S. Hauck, Z. Li, and E. Schewabe. Configuration Compression for the Xilinx XC6200 FPGA. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'98)*, pages 138–147, 1998.

[HSM00] W.J. Huang, N. Saxena, and E.J. McCluskey. A reliable LZ data compressor on reconfigurable coprocesors. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'00)*, pages 249–258, 2000.

[Hug68] F. G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14:55–63, 1968.

[HW97] J. Hauser and J. Wawrzynek. Garp: A MIPS Processor with a Reconfigurable Coprocessor. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'97)*, pages 12–21, 1997.

[JL98] L. Jimenez and D. A. Landgrebe. Supervised classification in highdimensional space: Geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transactions on System, Man, and Cybernetics, Part C: Applications and Reviews*, 28:39–54, 1998.

[JRMRD+05] L. O. Jimenez, J. L. Rivera-Medina, E. Rodriguez-Diaz, E. Arzuaga-Cruz, and 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 Transactions on Geoscience and Remote Sensing*, 43:844–851, 2005.

[KAT+03] M. Kifle, M. Andro, Q. K. Tran, G. Fujikawa, and P. P. Chu. Toward a dynamically reconfigurable computing and communication system for small spacecraft. *Proceedings of the 21st*

International Communication Satellite System Conference & Exhibit (ICSSC '03), 2003.

[KGV04] G. Kuzmanov, G. Gaydadjiev, and S. Vassiliadis. The MOLEN processor prototype. *Proceedings of 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '04)*, pages 296–299, 2004.

[KM02] N. Keshava and J. F. Mustard. Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57, 2002.

[KR06] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. *Proceedings of the ACM/SIGDA 14th International Symposium on Field-Programmable Gate Arrays (FPGA '06)*, pages 21–30, 2006.

[KZKP00] A. Kaarna, P. Zemcik, H. Kalviainen, and J. Parkkinen. Compression of multispectral remote sensing images using clustering and spectral reduction. *IEEE Transactions on Geoscience and Remote Sensing*, 38:1073–1082, 2000.

[LAB+04] J. Lázaro, A. Astarloa, U. Bidarte, J. Arias, and C. Cuadrado. High Throughput Serpent Encryption Implementation. *Lecture Notes in Computer Science*, 3203:996–1000, 2004.

[Lan02] D. Landgrebe. Hyperspectral image data analysis. *IEEE Signal Processing Magazine*, 19:17–28, 2002.

[Lan03] D. A. Landgrebe. *Signal Theory Methods in Multispectral Remote Sensing.* John Wiley & Sons: New York, 2003.

[Lap05] P. A. Laplante. Computing requirements for self-repairing space systems. *Journal of Aerospace Computing, Information and Communication*, 2(3):154–169, 2005.

[LCH00] Z. Li, K. Compton, and S. Hauck. Configuration Caching Management Techniques for Reconfigurable Computing. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'00)*, pages 22–36, 2000.

[LFDW99] D. Lavernier, E. Fabiani, S. Derrien, and C. Wagner. Systolic array for computing the pixel purity index algorithm on hyperspectral images. *Proceedings of SPIE*, 4480:130–138, 1999.

[LH00] Z. Li and S. Hauck. Configuration Compression for Virtex FPGAs. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01)*, pages 147–159, 2000.

[LH02] Z. Li and S. Hauck. Configuration prefetching techniques for partial reconfigurable coprocessor with relocation and defragmentation. *Proceedings of 10th ACM International Symposium on Field-Programmable Gate Arrays (FPGA '02)*, pages 287–195, 2002.

[LSL+99] G. Lu, H. Singh, M. Lee, N. Bagherzadeh, F. J. Kurdahi, and E. M. C. Filho. The morphoSys parallel reconfigurable system. *Proceedings of 5th International Euro-Par Conference on Parallel Processing (Euro-Par '99)*, pages 727–734, 1999.

[LTC03] A. Lodi, M. Toma, and F. Camp. A pipelined configurable gate array for embedded processors. *Proceedings of ACM/SIGDA 11th International Symposium on Field-Programmable Gate Arrays (FPGA '03)*, pages 21–29, 2003.

[LTS+00] D. Lavenier, J. Theiler, J. Szymansky, M. Gokhale, and J. Frigo. FPGA Implementation of the Pixel Purity Index Algorithm. *SPIE Photonics East, Workshop on Reconfigurable Architectures*, 2000.

[LTS+02] D. Lavernier, J. Theiler, J. Szymanski, M. Gokhale, and J. Frigo. FPGA implementation of the pixel purity index algorithm. *Proceedings of SPIE*, 4693:30–41, 2002.

[LWL00] K.H. Leung, K.W. Wong, and P.H.W. Leong. FPGA Implementation of a Microcoded Elliptic Curve Cryptographic Processor. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'00)*, pages 68–76, 2000.

[MKB+99] R. Maestre, F. J. Kurdahi, N. Bagherzadeh, H. Singh, R. Hermida, and M. Fernandez. Kernel scheduling in reconfigurable computing. *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pages 90–96, 1999.

[MM06] H. Masanori and K. Michitaka. A Multi-Context FPGA Using a Floating-Gate-MOS Functional Pass-Gate and its CAD Environment. *Proceedings of APCCAS*, pages 1803–1806, 2006.

[MMF98] O. Mencer, M. Morf, and M. J. Flynn. Hardware software tridesign of encryption for mobile communication units. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '98)*, 5:3045–3048, 1998.

[Moy01] B. Moyer. Low-power design for embedded processors. *Proceedings of the IEEE*, 89:1576–1587, 2001.

[ND99] B. Neil and A. Dawood. Reconfigurable computers in space: problems, solutions and future directions. *Proceedings of the Military and Aerospace Applications of Programable Logic Devices Conference*, 1999.

[NHS01] A. Nordin, C. Hsu, and H. Szu. Design of FPGA ICA for hyperspectral imaging processing. *Proceedings of SPIE*, 4391:444–454, 2001.

[NMC06] David T. Nguyen, Gokhan Memik, and Alok Choudhary. A reconfigurable architecture for network intrusion detection using principal component analysis. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, FPGA '06, pages 235–235, New York, NY, USA, 2006. ACM.

[OGR10] J. Olivito, C. González, and J. Resano. FPGA implementation of a strong reversi player. *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT '10)*, pages 507 –510, 2010.

[PB99] K. M. G. Purna and D. Bhatia. Temporal partitioning and scheduling data flow graphs for reconfigurable computers. *IEEE Transactions on Computers*, 48(6):579–590, 1999.

[PBB+09] A. Plaza, J. A. Benediktsson, J. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, J.A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni. Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, 113:110–122, 2009.

[PC06] A. Plaza and C.-I Chang. Impact of initialization on design of endmember extraction algorithms. *IEEE Transactions on Geoscience and Remote Sensing*, 44(11):3397–3407, 2006.

[PC07] A. Plaza and C.-I Chang. *High Performance Computing in Remote Sensing*. CRC Press: Boca Raton, FL, 2007.

[PKP+03] D. Pramanik, H. Kamberian, C. Progler, M. Sanie, and D. Pinto. Cost effective strategies for ASIC masks. *Proceedings of SPIE*, pages 142–152, 2003.

[PMPP02] A. Plaza, P. Martinez, R. Perez, and J. Plaza. Spatial/spectral endmember extraction by multidimensional morphological operations. *IEEE Transactions on Geoscience and Remote Sensing*, 40(9):2025–2041, 2002.

[PMPP04] A. Plaza, P. Martinez, R. Perez, and J. Plaza. A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 42(3):650–663, 2004.

[PPPM09] J. Plaza, A. Plaza, R. Perez, and P. Martinez. On the use of small training sets for neural network-based characterization of mixed pixels in remotely sensed hyperspectral images. *Pattern Recognition*, 42:3032–3045, 2009.

[PRMC07] E. Pérez, J. Resano, D. Mozos, and F. Catthoor. Memory Hierarchy for High-performance and Energy-aware Reconfigu-

rable Systems. *IET Computers and Digital Techniques*, pages 565–571, 2007.

[QHWM00] S.-E Qian, A. B. Hollinger, D. Williams, and D. Manak. Vector quantization using spectral index-based multiple subcodebooks for hyperspectral data compression. *IEEE Transactions on Geoscience and Remote Sensing*, 38:1183–1190, 2000.

[RCG+08] J. Resano, J. A. Clemente, C. González, D. Mozos, and F. Catthoor. Efficiently scheduling runtime reconfigurations. *ACM Transactions on Design Automation of Electronic Systems*, 13(4):58–70, 2008.

[RCGM07] J. Resano, J. A. Clemente, C. González, and D. Mozos. HW implementation of an execution manager for reconfigurable systems. *Proceedings of the The International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'07)*, pages 71–78, 2007.

[RDVC05] J. Resano, D.Mozos, D. Verkest, and F. Catthoor. A reconfiguration manager for dynamically reconfigurable hardware. *IEEE Design and Test of Computers*, 22(5):452–460, 2005.

[RLG+98] C.R. Rupp, M. Landguth, T. Garverick, E. Comersall, H. Holt, J.M. Arnold, and M. Gokhale. The NAPA adaptive processing architecture. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'98)*, pages 28–37, 1998.

[RMC05] J. Resano, D. Mozos, and F. Catthoor. A Hybrid Prefetch Scheduling Heuristic to Minimize at Run-Time the Reconfiguration Overhead of Dynamically Reconfigurable Hardware. *Proceedings of DATE '05*, pages 106–111, 2005.

[RSBK01] R. O. Reynolds, P. H. Smith, L. S. Bell, and H. U. Keller. Design of Mars lander cameras for Mars Pathfinder, Mars Surveyor '98 and Mars Surveyor '01. *IEEE Transactions on Instrumentation and Measurement*, 50(1):63–71, 2001.

[RSH05] G. K. Rauwerda, G. J. M. Smit, and P. M. Heysters. Implementation of multi-standard wireless communication receivers in a heterogeneous reconfigurable system-on-chip. *Proceedings of the 16th ProRISCWorkshop*, pages 421–427, 2005.

[RSMM06] S. Román, J. Septién, H. Mecha, and D. Mozos. Constant Complexity Management of 2D HW Multitasking in Run-Time Reconfigurable FPGAs. *Lecture Notes in Computer Science*, 3985:187–192, 2006.

[Sch97a] H. Schmit. Incremental Reconfiguration for Pipelined Applications. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'97)*, pages 16–18, 1997.

[Sch97b] R. A. Schowengerdt. *Remote Sensing: Models and Methods for Image Processing, 2nd ed.* Academic Press: New York, 1997.

[SMP99a] R. Sidhu, A. Mei, and V.K. Prassana. Genetic Programming using Self-Reconfigurable FPGAs. *Lecture Notes in Computer Science*, 1673:301–312, 1999.

[SMP99b] R. Sidhu, A. Mei, and V.K. Prassana. String Matching on Multicontext FPGAs using Self-Reconfiguration. *Proceedigns of the ACM International Symposium in Field Programmable Gate Arrays (FPGA'99)*, pages 217–226, 1999.

[SP01] R. Sidhu and V.K. Prassana. Fast Regular Expression Matching using FPGAs. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01)*, pages 698–709, 2001.

[ST04] C. F. Da Silva and A. M. Tokarnia. RECASTER: synthesis of fault-tolerant embedded systems based on dynamically reconfigurable FPGAs. *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS '04)*, pages 2003–2008, 2004.

[Ste86] S. R. Sternberg. Greyscale Morphology. *Computer Vision Graphics and Image Processing*, 35:283–305, 1986.

[SUP+09] M. E. Schaepman, S. L. Ustin, A. Plaza, T. H. Painter, J. Verrelst, and S. Liang. Earth system science related imaging spectroscopy - an assessment. *Remote Sensing of Environment*, 113:123–137, 2009.

[Sys00] Chameleon Systems. CS2000 Advance Product Specification. *http://www.isis.vanderbilt.edu/projects.asp*, 2000.

[TB01] R. Tessier and W. Burleson. Reconfigurable computing and digital signal processing: a survey. *Journal of VLSI Signal Processing*, 28:7–27, 2001.

[TCJW97] S. Trimberger, D. Carberry, A. Johnson, and J. Wong. A time-multiplexed FPGA. *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'97)*, pages 22–28, 1997.

[Tho] J. T. Thomson. Rad Hard FPGAs. *Available online: http://esl.eng.ohio-state.edu/~rstheory/iip/RadHardFPGA.doc.*

[TMS+03] J. Tabero, H. Mecha, J. Septién, S. Román, and D. Mozos. A Vertex-List Approach to 2D Hw Multitasking Management in RTR FPGAs. *Proceedings of DCIS*, pages 545–550, 2003.

[Tri98] Triscend. Configurable Processors: An Emerging Solution for Embedded System Design. *A White Paper*, 1998.

[Tul97] J. Tuley. Soft Computing Reconfigures Designer Options. *Embedded Systems*, page 76, 1997.

[Ver08] Verilog. IEEE Standard Verilog Hardware Description Language. *http://www.verilog.com/IEEEVerilog.html*, 2008.

[VHD06] VHDL. IEEE VHDL Analysis and Standarization Group. *http://www.vhdl.org/vasg*, 2006.

[VPVRP05] D. Valencia, A. Plaza, M. A. Vega-Rodríguez, and R. M. Pérez. FPGA design and implementation of a fast pixel purity index algorithm for endmember extraction in hyperspectral imagery. *Proceedings of SPIE*, 5995:599508.1–599508.10, 2005.

[VRJ00] M. Velez-Reyes and L.O. Jimenez. A fast canopy reflectance model to simulate realistic remote sensing scenarios. *In Algorithms for Multispectral, Hyperspectral and Ultraspectral Imagery VI, Proceedings of SPIE*, 4049:288–297, 2000.

[VRRPL03] M. Velez-Reyes, S. Rosario, A. Puetz, and R.B. Lockwood. Iterative algorithms for unmixing of hyperspectral imagery. *In Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery IX, Proceedings of SPIE*, 5093, 2003.

[Win03] M. E. Winter. N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. *Proc. SPIE Image Spectrometry V*, 3753:266–277, 2003.

[Win04] M. E. Winter. A proof of the N-FINDR algorithm for the automated detection of endmembers in a hyperspectral image. In *Proceedings of SPIE*, volume 5425, pages 31–41, 2004.

[WK07] M. Watanabe and F. Kobayashi. A 0.35 um CMOS 1,632–Gate–Count Zero–Overhead Dynamic Optically Reconfigurable Gate Array VLSI. *Proceedings of ASP-DAC '07*, pages 124–125, 2007.

[Xil] Xilinx. *Available online: http://www.xilinx.com/publications/prod_mktg/virtex5qv-product-table.pdf.*

[Xil00] Xilinx. Using Block SelectRAM + Memory in Spartan II FPGAs. *Xilinx Application Notes, http://www.xilinx.com*, 2000.

[Xil03] Xilinx. Virtex 2.5V Field Programmable Gate Array. *Xilinx Documentation, http://www.xilinx.com*, 2003.

[Xil09] Xilinx. Available online: http://www.xilinx.com/support/documentation/ip_documentation/ppc405_virtex4.pdf. 2009.

[ZCS08] Fang Zhong, D.W. Capson, and D.C. Schuurman. Parallel architecture for pca image feature detection using fpga. In *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*, pages 001341 –001344, May 2008.