

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA



**DISEÑO DE LA ARQUITECTURA DE UN EXTRACTOR DE ENDMEMBERS
DE IMÁGENES HIPERESPECTRALES SOBRE UN FPGA EN TIEMPO REAL**

Tesis para optar el Título de Ingeniero Electrónico, que presenta el bachiller:

Christian Jair Luis Peña

ASESOR: MSc. Ing. Christian Enrique Cano Salazar

Lima, 2018

A mi familia, a mi asesor y a mis amigos del grupo de microelectrónica, de Bilingual y del colegio.



Resumen

El presente trabajo consiste en el diseño hardware de un extractor de endmembers para imágenes hiperespectrales en tiempo real empleando el algoritmo N-FINDR. Para comprobar la eficiencia de la arquitectura se utilizó la imagen hiperespectral Cuprite la cual tiene un tamaño de 350×350 y fue capturada por el sensor aerotransportado AVIRIS, el cual escanea una columna de 512 píxeles en 8.3ms. Por ende, el procesamiento de la referida imagen se realizará en menos de 1.98 segundos para alcanzar el tiempo real.

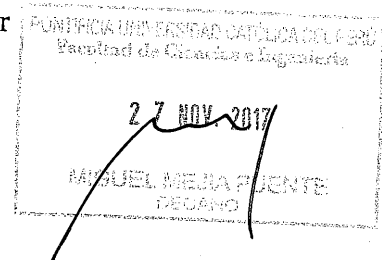
En primer lugar, el algoritmo fue analizado por medio del entorno de programación MATLAB® con el fin de identificar los procesos más costosos computacionalmente para optimizarlos. Además, se realizó el estudio de una nueva forma de eliminación de píxeles en el análisis por medio de un pre-procesamiento con la intención de reducir el tiempo de ejecución del algoritmo. Posteriormente, se analizó el proceso más costoso computacionalmente y se propuso un diseño algorítmico para mejorar la velocidad del proceso. En segundo lugar, se realizó la síntesis comportamental de la aplicación software con la finalidad de obtener una arquitectura hardware del sistema. La arquitectura fue descrita utilizando el lenguaje de descripción de hardware Verilog.

Finalmente, el diseño se verificó y validó mediante la herramienta ISim de Xilinx, a través del uso de testbenches, realizando la síntesis de la arquitectura diseñada sobre un FPGA Virtex 4 utilizando el software ISE de la empresa Xilinx obteniendo una frecuencia de operación estimada de 69.4Mhz, que representa un 64 % de mejora, respecto de la referencia [1], llegando a procesar una imagen hiperespectral en 17.98 segundos. Sin embargo, con esta frecuencia no es posible alcanzar el procesamiento en tiempo real esperado utilizando la familia Virtex 4. La arquitectura diseñada, fue optimizada utilizando paralelismo de operaciones, lo cual hace que se incremente el área de diseño, excediendo el límite de slices disponibles en el modelo Virtex 4 utilizando en la referencia [1], por ello se identificó mediante las hojas de datos de la familia Virtex que el FPGA más idóneo para soportar la arquitectura diseñada es la Virtex 7 modelo XC7VX980T que supera los 71,096 slices que requiere la presente arquitectura, obteniendo una frecuencia de operación de 112.819MHz.



TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

Título : Diseño de la arquitectura de un extractor de endmembers de imágenes hiperespectrales sobre un FPGA en tiempo real
Área : Circuitos y Sistemas # 1402
Asesor : MSc. Ing. Christian Enrique Cano Salazar
Alumno : Christian Jair Luis Peña
Código : 20120250
Fecha : 02/11/2017



Descripción y Objetivos

En los últimos años los sensores hiperespectrales han adquirido gran importancia en el rubro de la teledetección. Una de las principales aplicaciones se da en el ámbito de la geología, permitiendo la clasificación de tipos de rocas y estudio de recursos minerales, entre otros.

En el presente trabajo de tesis se propone la arquitectura hardware de un extractor de endmembers de imágenes hiperespectrales empleando el algoritmo N-FINDR. Con este fin se implementará, como primera etapa, el citado algoritmo en software, por medio de MATLAB. Luego se procederá con la síntesis comportamental del algoritmo en software, con lo cual se obtendrá la arquitectura en hardware. La descripción de dicha arquitectura será realizada en el lenguaje de descripción de hardware Verilog usando el software ISE de la compañía Xilinx y para ser sintetizada en un FPGA Virtex-4. Posteriormente, se realizará la verificación y validación funcional de la descripción mediante el uso de la simulación por Testbench en Verilog por medio de la herramienta ISim de la compañía Xilinx.

Objetivo principal del presente trabajo es diseñar en FPGA la arquitectura de un módulo de extracción de endmembers de imágenes hiperespectrales empleando el algoritmo N-FINDR.

Los objetivos específicos son los siguiente:

- Implementación del algoritmo N-FINDR en software.
- Diseño de la arquitectura hardware del módulo de N-FINDR.
- Descripción de la arquitectura en hardware utilizando el lenguaje de descripción de hardware Verilog, para ser sintetizada sobre un dispositivo FPGA Virtex-4.
- Verificación funcional de la arquitectura hardware, validándola con los resultados de las pruebas hechas en software.
- Optimización de la frecuencia de operación de la arquitectura para operar en tiempo real.

8

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

M. Sc. Ing. **WILLY CARRERA SORIA**
Coordinador de la Especialidad de Ingeniería Electrónica

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

Título : Diseño de la arquitectura de un extractor de endmembers de imágenes hiperespectrales sobre un FPGA en tiempo real

Índice

Introducción

1. Marco problemática
2. Marco teórico
3. Aplicación en software y diseño en hardware de un extractor de endmembers de imágenes hiperespectrales
4. Resultados

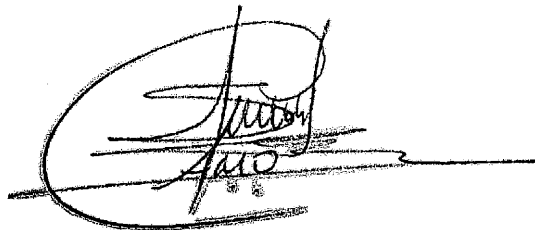
Conclusiones

Recomendaciones

Bibliografía

Anexos

Máximo : 100 páginas



PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA


M. Sc. Ing. **WILLY CARRERA SORIA**
Coordinador de la Especialidad de Ingeniería Electrónica



Índice general

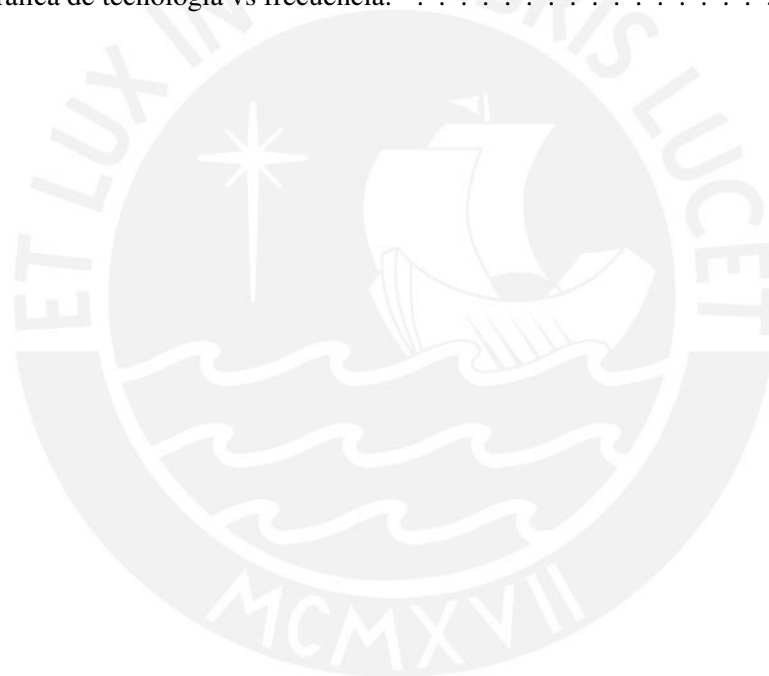
Introducción	1
1. Marco problemático	3
1.1. Descripción y formulación del problema	3
1.2. Importancia y justificación del estudio	5
1.3. Objetivos	6
1.3.1. Objetivo General	6
1.3.2. Objetivos Específicos	6
2. Marco teórico	8
2.1. Imágenes hiperespectrales: Conceptos	8
2.1.1. Radiancia y reflectancia	8
2.1.2. Sensor hiperespectral	9
2.1.3. Imagen hiperespectral	9
2.1.4. Pixel puro y pixel mezcla	10
2.1.5. Imagen hiperespectral Cuprite	10
2.2. Proceso de desmezclado espectral	11
2.2.1. Módulo de extracción de endmember	12
2.2.2. Comparación entre los algoritmos existentes	12
2.3. N-FINDR	13
2.4. Dispositivo Lógico Programable FPGA	16
2.4.1. Características de un circuito digital	16
3. Aplicación en software y diseño en hardware de un extractor de endmembers de imágenes hiperespectrales	18
3.1. Implementación software	18
3.1.1. Introducción al entorno de programación MATLAB®	18

3.1.2.	Estructura del código	19
3.1.3.	Resultados	19
3.1.4.	Análisis de los resultados	20
3.1.5.	Análisis de pruebas en software	21
3.2.	Diseño de la arquitectura hardware	24
3.2.1.	Diagrama de bloques	24
3.2.2.	Módulo N-FINDR	24
3.2.2.1.	Algoritmo del cálculo de la determinante	25
3.2.3.	Método propuesto	25
3.2.4.	Diseño del módulo de cálculo de área	28
3.2.5.	Diseño del módulo N-FINDR	28
3.2.6.	Módulo DMA	31
3.2.7.	Módulo control	32
3.2.8.	Diagrama de integración completa del módulo de extracción de endmembers	33
3.2.9.	Análisis de potencia de la codificación de las máquinas de estados	33
4.	Resultados	35
4.1.	Análisis del error generado en la arquitectura	35
4.2.	Simulaciones de los módulos de las arquitecturas diseñadas	37
4.3.	Simulación del bloque N-FINDR	37
4.4.	Simulación del bloque DMA	39
4.5.	Simulación del bloque Control	39
4.6.	Simulación del sistema de extracción de endmembers	40
4.7.	Evaluación de los endmembers	41
4.8.	Resultados de la síntesis de la arquitectura diseñada	44
	Conclusiones	47
	Recomendaciones	49
	Bibliografía	51

Índice de figuras

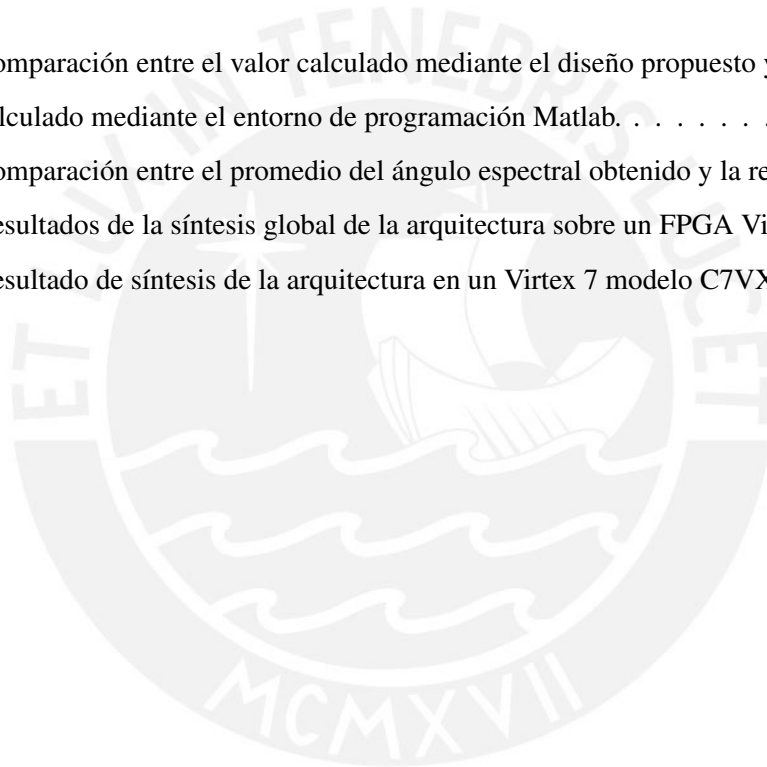
2.1. Toma de datos de un sensor hiperespectral	9
2.2. Foto aérea de Egipto.	10
2.3. Imagen de la zona minera de Cuprite	11
2.4. Cadena de desmezclado espectral.	12
2.5. Cadena de desmezclado espectral.	12
2.6. Diagrama de flujo de N-FINDR.	14
2.7. Algoritmo N-FINDR.	15
3.1. Diagrama de la estructura del código.	19
3.2. Gráfica de tiempo de ejecución para el procesamiento de la imagen Cuprite vs cantidad de endmembers	20
3.3. Resultado de utilizar el algoritmo N-FINDR tras la reducción dimensional de la imagen Cuprite a 2 dimensiones.	21
3.4. Resultado de utilizar el algoritmo N-FINDR quitando del análisis a los puntos que tienen una distancia menor a una desviación estándar.	22
3.5. Tiempo de ejecución del algoritmo N-FINDR con eliminación de píxeles centrales.	23
3.6. Porcentaje de ahorro de tiempo utilizando la metodología planteada y porcentaje de píxeles eliminados.	23
3.7. Diagrama de bloques planteado para la arquitectura hardware del sistema.	24
3.8. Ejemplo del método propuesto para cancelar términos.	27
3.9. Diagrama de bloques del módulo de cálculo de área.	29
3.10. Diagrama de bloques del módulo N-FINDR.	30
3.11. Diagrama de bloques del módulo DMA	31
3.12. Diagrama de bloques del módulo control.	32
3.13. Diagrama de bloques del módulo de extracción de endmember	33

4.1. Gráfica del error en el cálculo de la multiplicación de la diagonal principal vs la cantidad de bits decimales.	36
4.2. Histograma del error generado en la arquitectura propuesta	37
4.3. Simulación del módulo de cálculo de la determinante	38
4.4. Simulación del bloque N-FINDR	39
4.5. Simulación del módulo DMA.	39
4.6. Simulación del módulo control.	40
4.7. Simulación del sistema completo	40
4.8. Comparación de los endmembers	41
4.9. Comparación de los resultados utilizando el ángulo espectral	43
4.10. Reporte de tiempos y frecuencia de la síntesis en un FPGA Virtex 4.	45
4.11. Gráfica de tecnología vs frecuencia.	45



Índice de cuadros

2.1. Costo computacional de los algoritmos de extracción de endmember	13
3.1. Resultados de la potencia dinámica disipada por cada bloque	34
4.1. Comparación entre el valor calculado mediante el diseño propuesto y el valor real cálculado mediante el entorno de programación Matlab.	36
4.2. Comparación entre el promedio del ángulo espectral obtenido y la referencia [1].	43
4.3. Resultados de la síntesis global de la arquitectura sobre un FPGA Virtex 4.	44
4.4. Resultado de síntesis de la arquitectura en un Virtex 7 modelo C7VX980T.	46



Introducción

La minería en el Perú representa el 50 % de las divisas según el IPE¹. No obstante, se pierden muchas oportunidades debido a la dificultad del estudio que acarrea algunas zonas. Asimismo, existe una gran preocupación por la contaminación y por la explotación de ríos y lagos, mayormente producida por la minería ilegal. Sin embargo, existe una tecnología emergente actual conocida como imágenes hiperespectrales que permitiría abordar ambos problemas a través de la exploración con drones equipados con sensores hiperespectrales, estos capturan imágenes con alta resolución espacial y espectral que al ser procesadas permitirá analizar diversas propiedades físicas de un área geológica, así como la identificación porcentual de minerales en la zona analizada [2]. No obstante, procesar este tipo de imágenes, en tiempo real, requiere de una tecnología computacional avanzada, por ello es que actualmente solo se toman imágenes y luego se transfieren a un ordenador para su procesamiento. El procesamiento y análisis en tiempo real de estas imágenes permitirían tomar decisiones en menor tiempo, lo que resultará en acciones inmediatas, en caso se tratase de una potencial contaminación ambiental.

Las imágenes hiperespectrales permiten identificar los minerales por su longitud de onda a través de la extracción de los endmembers de la imagen. Sin embargo, esto involucra un alto grado de complejidad computacional debido a las operaciones de multiplicación y división en punto flotante que a su vez conlleva a un alto consumo de energía usando los procesadores convencionales. En la referencia [3] afirman que el uso de instrucciones de punto flotante personalizadas en FPGA disipa el 70 % menos de potencia que las configuraciones basadas en software. Siendo estos programas basados en software no viable para equipos que funcionan con baterías como los drones. Por los motivos expuestos, los FPGAs representan la mejor alternativa debido a su bajo consumo energético y alto desempeño computacional [4, 5, 6].

El presente trabajo propone diseñar la arquitectura hardware de un extractor de endmembers, en tiempo real, de imágenes hiperespectrales sobre un FPGA utilizando el algoritmo N-FINDR. Para ello, en capítulo 1 se trata la importancia de las imágenes hiperespectrales. En el capítulo 2 se

¹Instituto Peruano de Economía

detalla los conceptos teóricos que involucran las imágenes hiperespectrales y su procesamiento. En el capítulo 3 explica el diseño software y hardware que se desarrolló para el algoritmo N-FINDR. Finalmente, en el capítulo 4 se muestra los resultados obtenidos.



Capítulo 1

Marco problemático

La solución a los problemas que ves en tu vida es vivir en tal forma que desaparezca lo problemático.

Ludwig Wittgenstein

1.1. Descripción y formulación del problema

En los últimos años los sensores hiperespectrales han adquirido gran importancia en el rubro de la teledetección. Una de las principales aplicaciones de estos sensores es en el ámbito de la geografía donde destaca la facilidad de este medio para realizar mapeos geológicos, clasificación de tipos de rocas, y estudio de recursos minerales, entre otros [7]. Sin embargo, la transmisión de información en tiempo real de las distintas geografías requiere de una gran tasa de transferencia de datos. Por ejemplo, el sensor AVIRIS de la NASA, en el año 2016, alcanzó una tasa de adquisición de 17MB/s. Esto repercute en el alto grado de consumo de energía para realizar este procedimiento reduciendo el tiempo de vida efectivo de una batería, especialmente en equipos móviles como drones. Por ello, se necesita transmitir esta información de forma eficiente para así mejorar el rendimiento y tiempo de vida efectivo de la batería de un dron. Esta necesidad ha generado nuevos desafíos en el ámbito del procesamiento de imágenes hiperespectrales debido a la necesidad de procesar la información en tiempo real y tomar acciones y/o decisiones en el menor tiempo posible. Para realizar el procesamiento de este gran volumen de información que generan las imágenes hiperespectrales actualmente se opta por el uso de clústeres y sistemas multiprocesadores (GPUs) debido a su alto rendimiento computacional. Sin embargo, debido a la gran cantidad de energía que consumen estos dispositivos no es recomendable su implementación en dispositivos móviles

no tripulados como los drones. Es por ello que se han explorado nuevas tecnologías para abordar estas limitantes en lo que respecta a nuevos sistemas de procesamiento, poseyendo los FPGAs una tecnología de gran rendimiento y estándares de consumo de energía sumamente bajos respecto a los procesadores convencionales [4, 5, 6].

El procesamiento de imágenes hiperespectrales consta principalmente de dos etapas: extracción de endmembers y estimación de abundancias [1]. No obstante, se pueden incluir otras etapas como la etapa de corrección atmosférica la cual realiza la transformación de las unidades de radiancia (cantidad de luz emitida en todo el espectro de frecuencias) a reflectancia, la cual es una propiedad intrínseca de los materiales [1]. Existen algunos estudios y análisis respecto a algoritmos que buscan procesar las imágenes hiperespectrales. Sin embargo, uno de los más resaltante es el estudio realizado en [1, 8] donde se demostró que para el caso del algoritmo PPI (Pixel Purex Index) el bloque de corrección atmosférica es intrascendente pues los resultados obtenidos sin y con este módulo son muy parecidos. Asimismo, el bloque de reducción dimensional no siempre es requerido puesto que el uso de este bloque depende del algoritmo de extracción de endmember a utilizar. Por ejemplo, el algoritmo N-FINDR [9] necesita del bloque de reducción dimensional para su implementación mientras que el algoritmo PPI no lo necesita. El bloque de extracción de endmember tiene como finalidad extraer las firmas espectrales (endmembers) presentes en la imagen, los cuales son patrones de frecuencia únicos por cada material. Finalmente, el bloque de estimación de abundancias parte de los endmembers extraídos para calcular su porcentaje de estos en cada pixel de la imagen y así determinar el porcentaje total.

El presente trabajo se enfocará en el bloque de extracción de endmembers. Este bloque tiene como objetivo extraer el mejor conjunto de espectros para los endmembers para así mitigar el ruido que es generado por el procesamiento. Este ruido generado se calcula a través de la ecuación 1.1:

$$P_i = \sum_{j=0}^m a_{ij} E_j + n_i \quad (1.1)$$

Donde P_i es el espectro del píxel i , E_j es el espectro del endmember j , a_{ij} es la abundancia del endmember E_j en el pixel i y n es el ruido presente en el pixel i . De esta ecuación se deduce que el ruido generado en cada píxel depende de los espectros de los endmembers y de su respectiva abundancia en el pixel. Por ello, la elección correcta del conjunto de espectros es fundamental para el procesamiento. Actualmente existen varios algoritmos que resuelven este problema [9, 10, 11, 12, 13, 14] y cada uno de estos algoritmos sobresale en algún aspecto frente a los demás. Por ejemplo VCA es un algoritmo rápido de baja carga computacional [14] mientras

que N-FINDR es más lento pero obtiene mejores resultados que el VCA tomando como fuente una imagen hiperespectral real [15].

Actualmente se están desarrollando nuevos algoritmos para la extracción de endmembers y estos poseen ventajas sustanciales con respecto a sus predecesores. Estas ventajas se pueden observar en la carga computacional que requieren y la calidad de los endmembers extraídos, entre otros. Un ejemplo de estos algoritmos nuevos es el EE-DFA (Endmember extraction-discrete firefly algorithm) [15]. Asimismo, existen trabajos de investigación enfocados en crear nuevos algoritmos partir de los algoritmos existentes, como el MVCA (Modified vertex component analysis) [16] que es una versión modificada de VCA, y SGA (Simplex growing algorithm) que es una versión modificada de N-FINDR. El SGA logró solucionar algunos problemas del anterior algoritmo, como el factor aleatorio en el algoritmo N-FINDR, sin afectar la carga computacional [12], lo cual puede ser de gran importancia en aplicaciones que involucren dispositivos móviles.

En la actualidad existen varias implementaciones de estos algoritmos en FPGA [17, 18, 19, 20, 21]. Muchas de estas implementaciones trabajan en tiempo real logrando así superar a implementaciones en entornos software como Matlab o C++. Por citar un caso, la implementación en FPGA de MSVA logra procesar la imagen hiperespectral Cuprite en 0.84 segundos mientras que el software Matlab utilizando el mismo algoritmo con un procesador Intel Core i3 trabajando a 3.5 Ghz se demora 163.38 segundos [21]. Adicionalmente, el consumo computacional, de hasta 2.45 Gflops para procesar la imagen hiperespectral Cuprite, y energético, requerido por el procesador es alto, si se quiere pensar en una potencial implementación sobre dispositivos móviles como drones.

1.2. Importancia y justificación del estudio

A lo largo de la historia, la minería jugó un rol importante dentro de la economía del Perú. Asimismo, en este rubro el Perú destaca en la producción de plata y zinc llegando a ser el segundo productor a nivel mundial de estos materiales. Sin embargo, la práctica irresponsable y desregulada de esta actividad ha ocasionado graves daños al medio ambiente produciendo así el rechazo a esta actividad por gran parte la población peruana. Un ejemplo de este problema es la minería ilegal en los ríos de la Amazonía. Se estima que cada año se produce en Madre de Dios entre 16 000 Kg a 18 000 Kg de oro y por cada Kg de oro extraído se utilizan 2.8 Kg de mercurio [22]. Por otro lado, cada año se pierde numerosas oportunidades mineras debido a la dificultad del estudio geográfico de las zonas de difícil acceso.

Una solución a estos problemas es el uso de drones equipados con sensores hiperespectrales

[7]. La principal ventaja de estos sensores con respecto a los otros es la enorme cantidad de información que brindan acerca de la imagen. Con estos datos se puede obtener no solo los materiales que están en la imagen, sino también se puede estimar la abundancia de estos permitiendo de este modo realizar el estudio sobre los minerales en un determinado territorio de forma rápida y eficiente. Actualmente en el Perú ya se hace uso de drones equipados con sensores hiperspectrales. Sin embargo, estos vehículos aéreos requieren volver a la estación base para descargar la información a una computadora para su posterior procesamiento. Uno de los inconvenientes de transmitir estos datos desde un dron en movimiento es la gran tasa de transmisión que requieren debido al gran volumen de información a ser transmitida. Debido a este problema se requiere del procesamiento de la información durante el vuelo del dron para así transmitir solo los datos comprimidos y útiles en tiempo real. La alternativa más viable para realizar este complejo procesamiento utilizando la menor cantidad de energía es el FPGA. Además de la ventaja energética que posee frente a los sistemas multiprocesadores, que podrían realizar también este procesamiento, el FPGA es versátil; es decir, se puede rediseñar la arquitectura del FPGA innumerables veces para así poder implementar futuras mejoras al diseño.

El diseño de la arquitectura propuesta en la presente tesis solo se probará sobre la imagen hiperspectral de la región minera de Cuprite por ser la imagen predeterminada para hacer las comparaciones en esta área. Sin embargo, el algoritmo de extracción de endmembers N-FINDR también puede ser utilizado para cualquier otra aplicación que requiera el procesamiento de imágenes hiperspectrales como identificación contaminantes en la agricultura, entre otras [7].

1.3. Objetivos

1.3.1. Objetivo General

Realizar el diseño de la arquitectura hardware para el módulo de extracción de endmembers de imágenes hiperspectrales usando el algoritmo N-FINDR sobre un FPGA.

1.3.2. Objetivos Específicos

- Implementación del algoritmo N-FINDR en software.
- Diseño de la arquitectura del módulo N-FINDR.
- Descripción de la arquitectura en hardware utilizando el lenguaje de descripción de hardware Verilog, para ser sintetizada en un dispositivo FPGA Virtex-4.

- Verificación funcional de la arquitectura hardware, validándola con los resultados de las pruebas hechas en software.
- Optimización de la frecuencia de operación de la arquitectura para operar en tiempo real.



Capítulo 2

Marco teórico

Cada día sabemos más
y entendemos menos.

Albert Einstein

Debido a la gran cantidad de información que posee una imagen hiperespectral es necesario realizar su procesamiento con el fin de obtener la información relevante de estas imágenes. Esta información se traduce en mapas de abundancia. Estos mapas son importantes ya que revelan la concentración geográfica de cada material presente en la imagen hiperespectral. Sin embargo, este proceso acarrea una serie de tareas de gran complejidad computacional. Por tanto, este capítulo se enfoca en las nociones básicas para entender el procesamiento de imágenes hiperespectrales y en explicar el funcionamiento del algoritmo N-FINDR para extracción de endmembers.

2.1. Imágenes hiperespectrales: Conceptos

2.1.1. Radiancia y reflectancia

La radiancia es una magnitud física vectorial que mide la cantidad de energía electromagnética emitida o que pasa por un cuerpo. Por otro lado, la reflectancia mide la cantidad de energía electromagnética que es reflejada por un objeto. Esta última propiedad es intrínseca del material y si se obtiene la suficiente información acerca de esta, es posible caracterizar a los materiales[2].

2.1.2. Sensor hiperespectral

A continuación, se detalla el funcionamiento del sensor hiperespectral, ver Figura 2.1. El sol emite ondas electromagnéticas a la tierra para ello estas ondas pasan por la atmósfera la cual permite el paso de ciertas frecuencias y mitiga a otras. Esta disminución de potencia en estas bandas de frecuencia se debe principalmente al vapor de agua, el cual absorbe la energía de estas bandas. Luego, al llegar a la tierra estas ondas se reflejan según la firma espectral del material al cual impactan para finalmente volver a pasar por la atmósfera y ser sentido por la cámara hiperespectral. La señal sensada por la cámara es la radiancia.

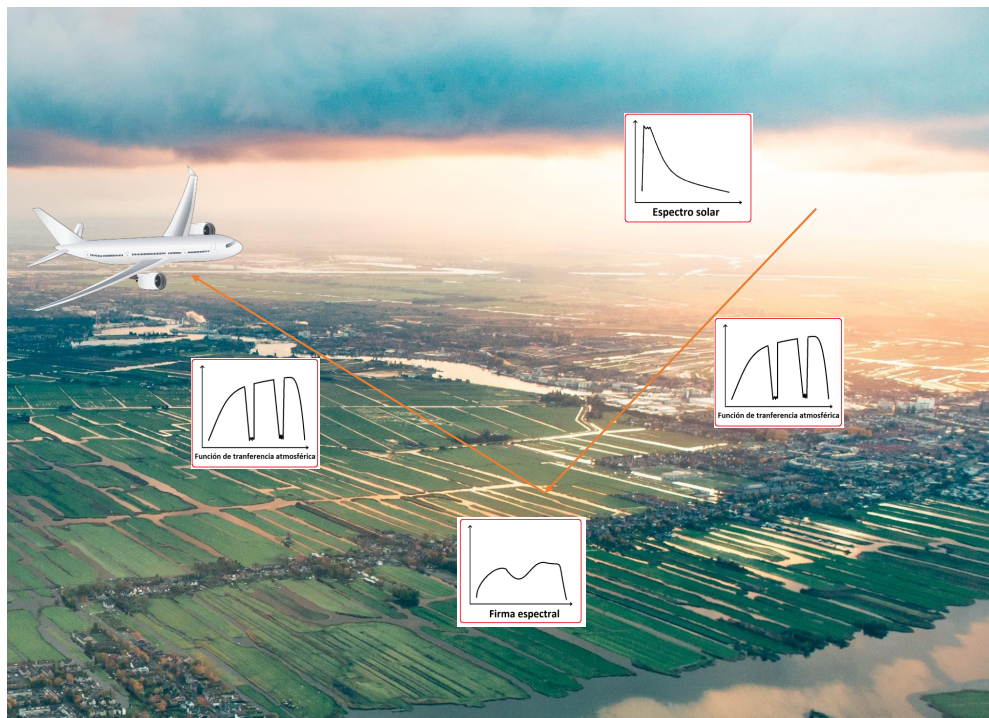


Figura 2.1: Toma de datos de un sensor hiperespectral. Inspirado en [2].

Para obtener la reflectancia del material, firma espectral, es necesario aplicar una corrección a la señal sensada debido a que esta señal fue modificada por la atmósfera. Esta corrección es conocida como corrección atmosférica y permite pasar los datos de valores de radiancia a reflectancia.

2.1.3. Imagen hiperespectral

Una imagen hiperespectral consta de una serie de cientos de imágenes tomadas en distintas bandas de frecuencia en el espectro electromagnético. Estas imágenes poseen una gran resolución espectral, de 0.94nm en el caso del sensor Aviris, lo cual permite detectar firmas espectrales.

2.1.4. Pixel puro y pixel mezcla

En la Figura 2.2 se observa los píxeles puros enmarcados en rojo. Estos píxeles se caracterizan por solo contar con una característica espectral(en este caso vegetación, arena o agua) mientras que los pixeles enmarcados en blanco, pixeles mezcla, son formados por más de un píxel puro. En la figura se observa que estos píxeles mezcla son la combinación del píxel puro de agua con el píxel puro de arena o del píxel puro de vegetación con el píxel puro de arena.



Figura 2.2: Foto aérea de Egipto.

Estos píxeles puros, también llamados endmembers, tiene la propiedad de caracterizar a toda la imagen. Por ello cada píxel de la imagen debe de ser descrita mediante una combinación lineal, con coeficientes no negativos y suma unitaria, de estos pixeles.

2.1.5. Imagen hiperespectral Cuprite

En la Figura 2.3 se muestra la zona minera de Cuprite ubicada en Nevada, Estados Unidos. Esta imagen es la más utilizada para comprobar la eficiencia de los algoritmos en el procesamiento de imágenes hiperespectrales. Esta imagen ha sido ampliamente estudiada y documentada en la librería U.S Geological Survey y está disponible para su descarga en la referencia [23].

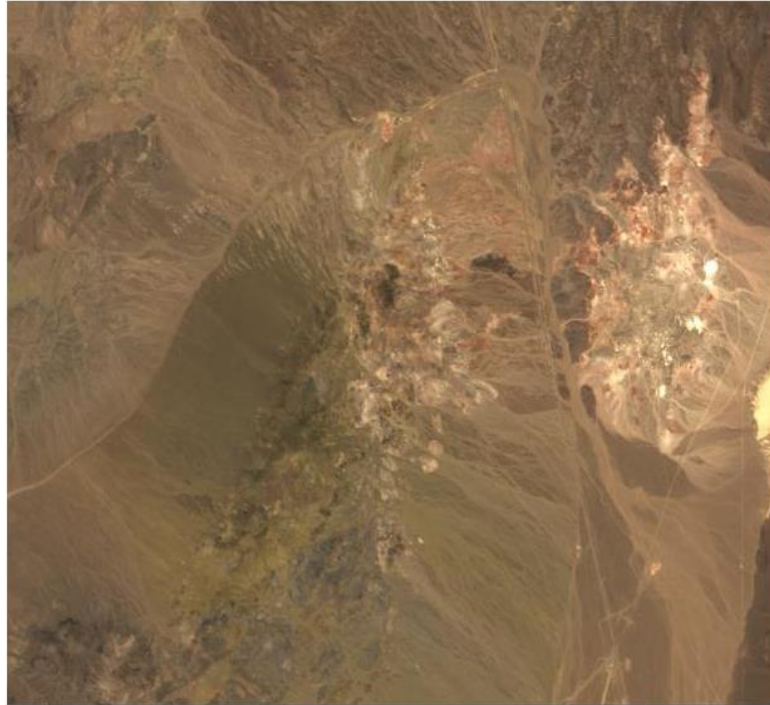


Figura 2.3: Imagen de la zona minera de Cuprite. Esta imagen se obtuvo utilizando las bandas de la imagen hiperespectral número 30, 20 y 12 que corresponde a longitudes de onda de $671.9 \mu\text{m}$, $578.1 \mu\text{m}$ y $503.1 \mu\text{m}$ respectivamente. Esta imagen hiperespectral está disponible para su descarga en la página web <https://aviris.jpl.nasa.gov>

2.2. Proceso de desmezclado espectral

El proceso de desmezclado espectral consiste en la descomposición de la imagen hiperespectral en su equivalente representado sólo por endmembers y sus respectivas abundancias en la imagen. Para ello se diferencian claramente tres etapas (Ver figura 2.4): La estimación de la cantidad de endmembers, la extracción de los endmembers y la estimación de la abundancia de cada endmember.

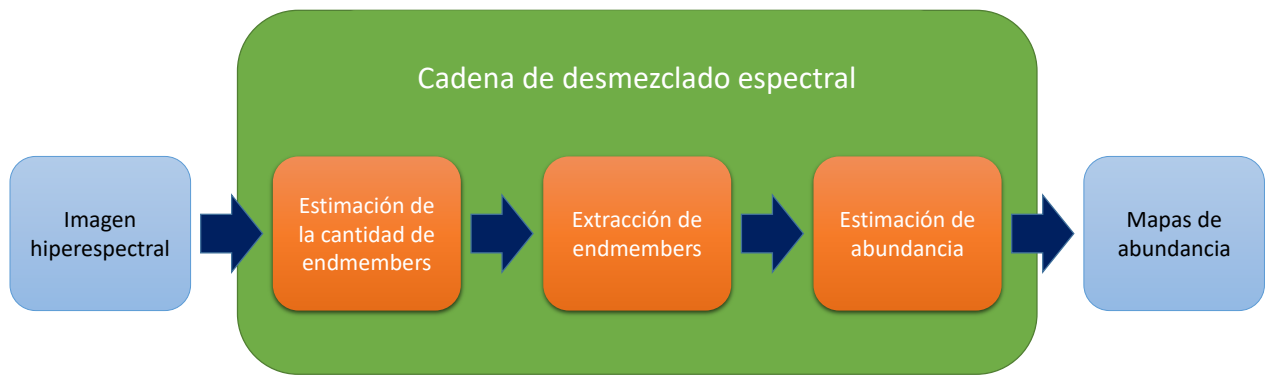


Figura 2.4: Cadena de desmezclado espectral.

2.2.1. Módulo de extracción de endmember

Este módulo tiene como objetivo detectar a los píxeles puros presentes en la imagen. Esta tarea puede ser realizada mediante dos sub-bloques (Ver Figura 2.5). El bloque de reducción dimensional, el cual reduce la dimensionalidad de la imagen según la cantidad de endmembers deseados, teniendo como ventaja reducir el costo computacional del bloque. Sin embargo, al realizar la reducción dimensional se pierde información. Existen diversos algoritmos que pueden realizar esta reducción y el más popular es el PCA (Principal Component Analysis). El segundo bloque realiza la búsqueda de los píxeles puros a través de los datos que ya han sido reducidos dimensionalmente. Cabe mencionar que no todos los algoritmos de extracción de endmembers necesitan del bloque de reducción dimensional [20, 21].

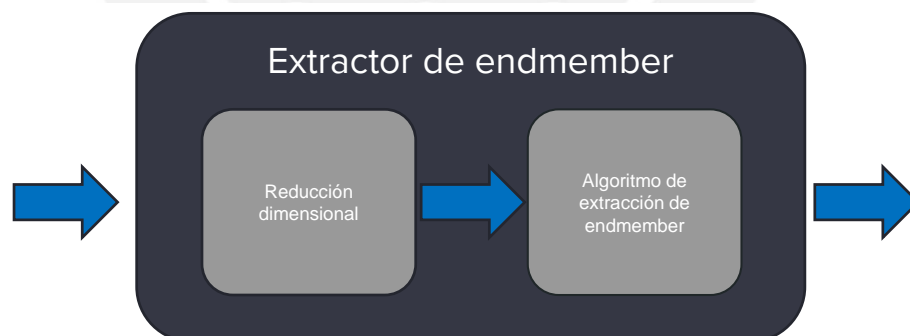


Figura 2.5: Cadena de desmezclado espectral.

2.2.2. Comparación entre los algoritmos existentes

Actualmente existen diversos algoritmos de extracción de endmember los cuales han sido desarrollados para superar los distintos problemas que existen en esta parte de la cadena de desmezclado espectral. Por citar uno de los problemas que se genera es la suposición de píxeles

puros dentro de la imagen. Gran parte de los algoritmos de extracción de endmember asumen la existencia de píxeles puros dentro de la imagen, es decir seleccionan como pixel puro a algún píxel dentro de la imagen. No obstante, esta premisa no siempre es cierta y puede que los píxeles puros no estén presentes en la imagen.

Una de las características más importantes de comparación es el costo computacional. En la Tabla 2.1 se observa el costo computacional de algunos de los algoritmos de extracción de endmembers. Se puede notar que existen diversos factores que influyen sobre este parámetro como la cantidad de endmembers que se desea conseguir(p) y la cantidad de píxeles en la imagen(N). En el caso de algoritmo PPI, este también depende de la cantidad de vectores aleatorios (skewers) a utilizar y en el caso del N-FINDR, la variable “ n ” tiene un valor entre 2.3 y 2.9 [12, 15].

Tabla 2.1: Costo computacional del algoritmo Vertex component analysis(VCA) ,N-FINDR , Simplex growing algorithm(SGA), Pixel purity index(PPI) y The sequential maximum angle convex cone(SMACC).

Algoritmo	Complejidad computacional (en número de flops)
VCA	$2p^2 N$
N-FINDR	$p^{n+1} N$
SGA	$\sum_{n=2}^p n^n N$
PPI	$2N sp$
SMACC	$p^2 N$

2.3. N-FINDR

Uno de los algoritmos más populares actualmente es el N-FINDR el cual se basa en formar el simplex, región N dimensional, del mayor volumen posible donde los vértices del simplex son los endmembers. Con el fin de formar este simplex N -dimensional es necesario realizar una reducción dimensional a la cantidad de endmembers deseados menos uno[9].

En la Figura 2.6 se muestra el diagrama de flujo del algoritmo; mientras que, Figura 2.7 se muestra el desarrollo de este. Primero se debe aplicar la reducción dimensional, en el ejemplo la reducción se realiza a dos dimensiones. Es decir cada punto de la muestra representa a un píxel de la imagen cuya ordenada y abscisa son determinados por el valor en sus dos bandas. El algoritmo comienza con la selección aleatoria los primeros endmembers para luego empezar el análisis con

el primer punto. Se reemplaza este primer punto por cada uno de los endmembers y se queda con el simplex de mayor volumen entre los simplex formados y el simplex original. Este proceso se vuelve a repetir para todos los píxeles de la imagen y como resultado final se obtiene el simplex de mayor volumen formado. Cada vértice de este simplex representa a un píxel que fue seleccionado como endmember[9].

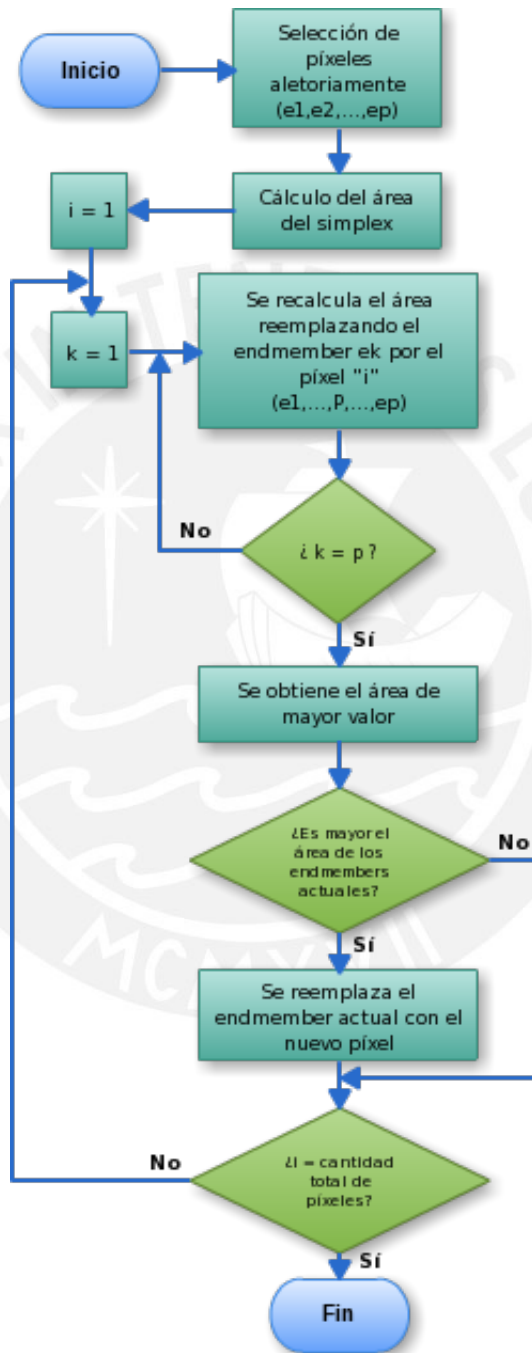


Figura 2.6: Diagrama de flujo de N-FINDR.

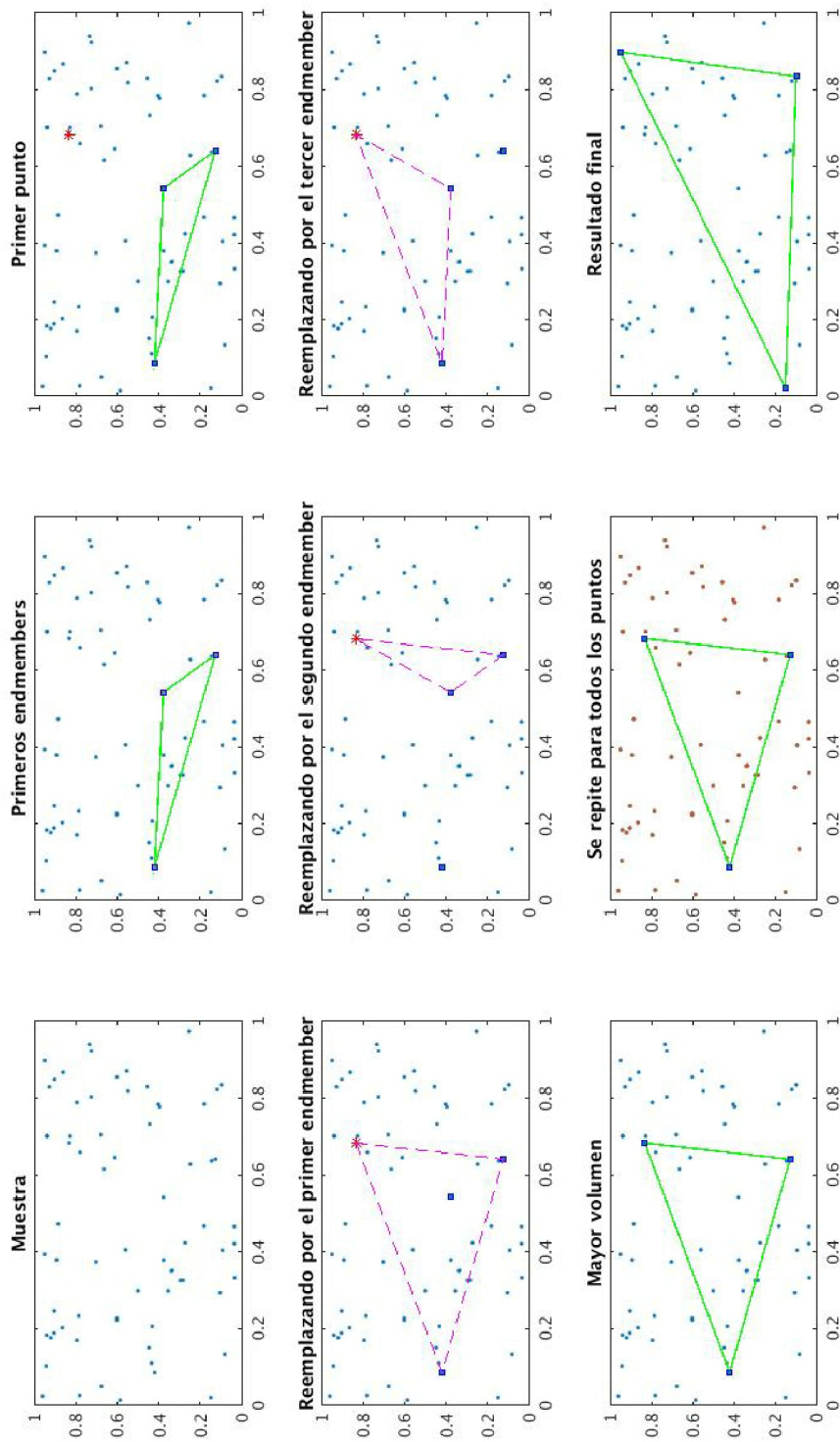


Figura 2.7: Algoritmo N-FINDR.

2.4. Dispositivo Lógico Programable FPGA

El FPGA(Field-programmable gate array) está compuesto por un arreglo de compuertas lógicas e interruptores programables. Este circuito integrado se caracteriza por el paralelismo y reconfigurabilidad que presenta. La propiedad del paralelismo significa que toda la lógica descrita en este dispositivo trabaja al mismo tiempo; mientras que, la reconfigurabilidad es la propiedad de volver a reconfigurar los elementos lógicos de este circuito en más de una ocasión. Existen varias tecnologías que utilizan los FPGA para guardar su configuración como por ejemplo la SRAM (Static Random Access Memory) y la EEPROM [24, 25, 26]. En la tecnología SRAM la configuración del FPGA se guarda en una memoria volátil; es decir, que el FPGA pierde su configuración al ser desenergizado. Por otro lado, los FPGA basados en memorias no volátiles EEPROM/FLASH no pierde su configuración al ser desenergizado. Cabe mencionar que no todos los FPGAs poseen la propiedad de reconfiguración como es el caso de los FPGA que utilizan la tecnología antifusible y solo se pueden configurar una única vez.

Debido a la gran cantidad de funciones que se pueden implementar en paralelo en los FPGA estos son ampliamente utilizados en aplicaciones que requieren de alto rendimiento computacional como es el caso las aplicaciones automovilísticas, médicas y de defensa. Asimismo, los FPGAs son utilizados para el prototipado de circuitos digitales. Por otro lado, los FPGA también son utilizados en las aplicaciones que requieren reconfiguración de los elementos lógicos cuando el sistema está en marcha, a este tipo de reconfiguración se le conoce como reconfiguración dinámica.

2.4.1. Características de un circuito digital

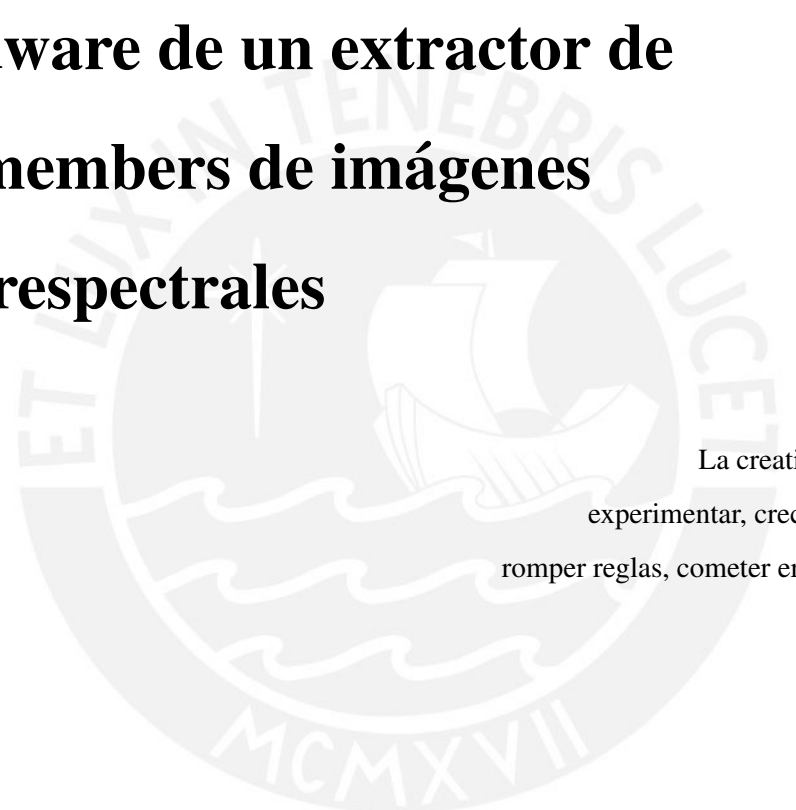
Los circuitos digitales se caracterizan a través de tres parámetros importantes: el área del circuito, la máxima frecuencia de operación y la potencia disipada. El área del circuito depende de la cantidad de transistores que se utilizan para la implementación de este y de la tecnología en la que se implementa. Este parámetro aproxima el precio del circuito integrado en caso se implemente en un ASICs. En el caso de los FPGAs de Xilinx esta característica se mide en slices, la cual es la unidad básica de construcción en esta compañía .La máxima frecuencia de operación es la frecuencia máxima en la que opera el circuito sin problemas de sincronización. Esta frecuencia varía según la tecnología en la que se implementa el diseño y se calcula utilizando el camino crítico del circuito, el cual es determinado por el tiempo que demora permutar los transistores en la lógica combinacional entre flip flops. Por último, la potencia disipada del circuito se determina mediante la suma de la potencia estática, producida por las corrientes de fuga de los transistores en el estado de corte, y la potencia dinámica, producida por la

permutación de los transistores. Entre estas dos potencias por lo general la dinámica es la mayor y representa el 85 % del total.



Capítulo 3

Aplicación en software y diseño en hardware de un extractor de endmembers de imágenes hiperespectrales



La creatividad es inventar,
experimentar, crecer, tomar riesgos,
romper reglas, cometer errores y divertirse.

Mary Lou Cook

3.1. Implementación software

Esta primera parte tiene como objetivo hacer un análisis del algoritmo N-FINDR con el propósito de verificar la funcionalidad del algoritmo e identificar etapas del mismo que pueden ser mejorados. Luego de esta verificación en software se realizó el diseño de la arquitectura a través de la síntesis comportamental usando la metodología ASMD.

3.1.1. Introducción al entorno de programación MATLAB®

Uno de los software de programación más populares para realizar investigación es MATLAB®. Este programa está basado en matrices y fue diseñado para realizar cálculos

científicos. Adicionalmente, personas de diversas parte del mundo desarrollan sus toolbox en esta plataforma y los comparten en páginas web de código abierto lo cual hace que se puedan desarrollar soluciones complejas utilizando paquetes ya desarrollados por otros científicos.

3.1.2. Estructura del código

Para realizar las pruebas del algoritmo en MATLAB® se utiliza la estructura que se muestra en la Figura 3.1. La primera etapa, lectura de la imagen hiperespectral, recibe como entrada la imagen hiperespectral con extensión .rfl y la codificación de esta imagen (número de bits por muestra, número de píxeles, entre otros). La segunda etapa, extracción de bandas de baja relación señal-ruido, retira las bandas de frecuencia donde la relación señal ruido es de 0 o que están próximo a este valor. La tercera etapa, Reducción dimensional, utiliza el algoritmo PCA(Principal Component Analysis) para esta tarea. Finalmente, se utiliza el algoritmo N-FINDR para la extracción de los endmembers.



Figura 3.1: Diagrama de la estructura del código.

3.1.3. Resultados

Como se observa en la Figura 3.2 el tiempo que demora en ejecutar el algoritmo crece conforme aumenta de la cantidad de endmembers que se desea obtener. Esto se debe principalmente a que al aumentar la cantidad de endmembers también aumenta la complejidad del cálculo del volumen del simplex y la cantidad de veces que se debe de calcular este volumen.

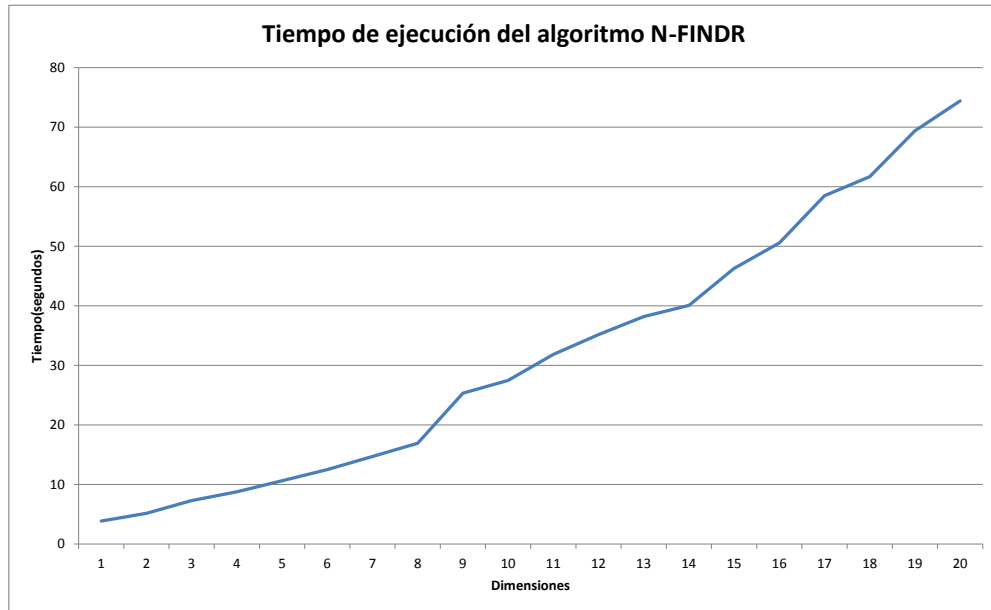


Figura 3.2: Gráfica de tiempo de ejecución para el procesamiento de la imagen Cuprite vs cantidad de endmembers utilizando un procesador i7.

3.1.4. Análisis de los resultados

En la Figura 3.3 se observa que los pixeles que tienen mayor posibilidad de ser seleccionados como endmembers se ubican en los bordes de la figura y que existe una gran concentración de puntos ubicados dentro de la figura formada que son datos que no se deben de analizar debido a que siempre serán descartados por generar áreas menores a los ubicados en los bordes de la figura.

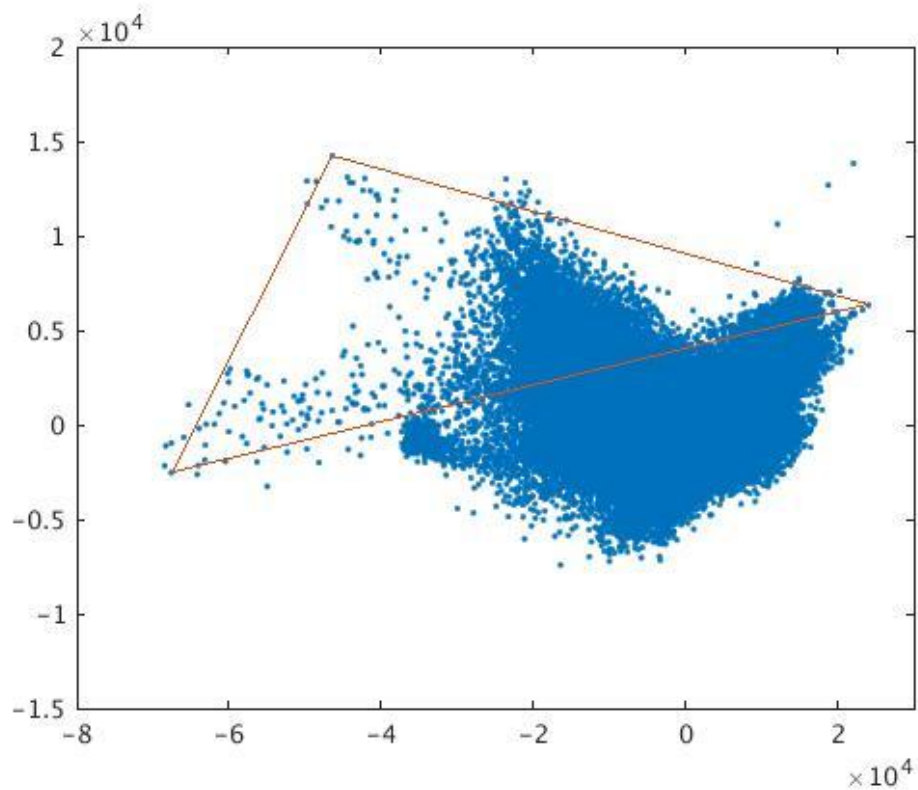


Figura 3.3: Resultado de utilizar el algoritmo N-FINDR tras la reducción dimensional de la imagen Cuprite a 2 dimensiones.

3.1.5. Análisis de pruebas en software

Debido a que el análisis de todos los puntos supone un costo computacional muy elevado el cual va aumentando conforme aumenta la dimensionalidad por ello se plantea descartar del análisis a los píxeles que estén cerca al centroide de la figura como se observa en la Figura 3.4, donde se descartan estos píxeles sin afectar en el desarrollo del algoritmo. Es importante destacar que con este método se logra descartar aproximadamente un 46 % de píxeles reduciendo el costo computacional global a casi el 44 % como se observa en las Figuras 3.5 y 3.6.

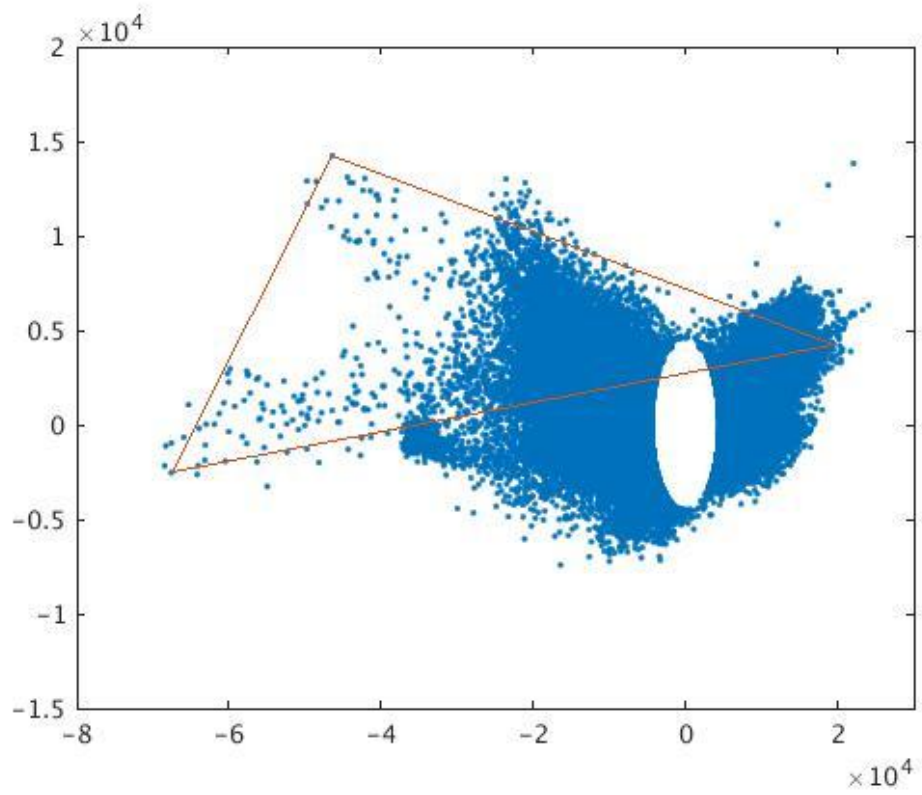


Figura 3.4: Resultado de utilizar el algoritmo N-FINDR quitando del análisis a los puntos que tienen una distancia menor a una desviación estándar.

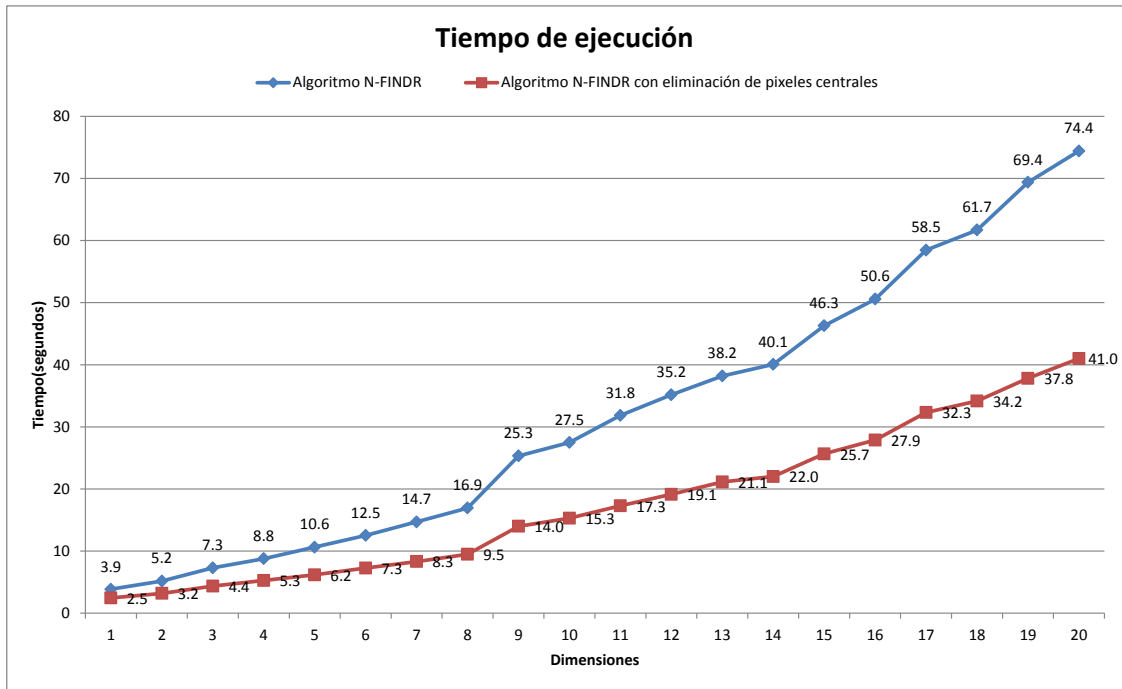


Figura 3.5: Tiempo de ejecución del algoritmo N-FINDR con eliminación de píxeles centrales.

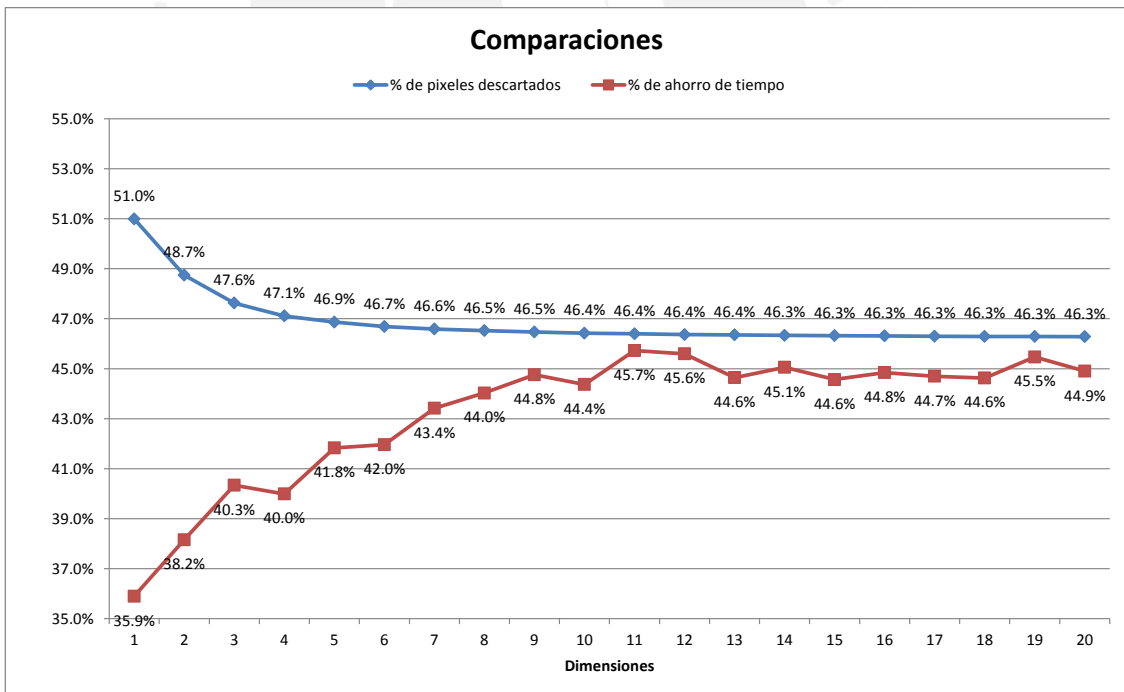


Figura 3.6: Porcentaje de ahorro de tiempo utilizando la metodología planteada y porcentaje de píxeles eliminados.

3.2. Diseño de la arquitectura hardware

Para el desarrollo de la arquitectura se plantea un bloque por cada función que debe de realizar el sistema (administración de la memoria, eliminación de píxeles centrales y desarrollo del algoritmo N-FINDR) y se utiliza el método de síntesis comportamental para el desarrollo de la arquitectura del eliminador de píxeles centrales y el algoritmo N-FINDR.

3.2.1. Diagrama de bloques

En la Figura 3.7 se observa el diagrama de bloques planteado para el sistema. En la memoria se almacena la imagen hiperespectral, ya reducida dimensionalmente. El bloque DMA(Direct memory access), tiene como función manejar toda comunicación con la memoria; en otras palabras, este bloque extrae los datos de la imagen hiperespectral y los dirige hacia el resto de los bloques según el sistema de control. El bloque Eliminador de píxeles centrales, es el primer bloque a ser ejecutado y determina los píxeles que deben ser analizados por el algoritmo N-FINDR. Al finalizar este análisis se procede a activar el bloque N-FINDR que cumple la función descrita por su nombre y entrega como resultado la posición de los endmembers elegidos.

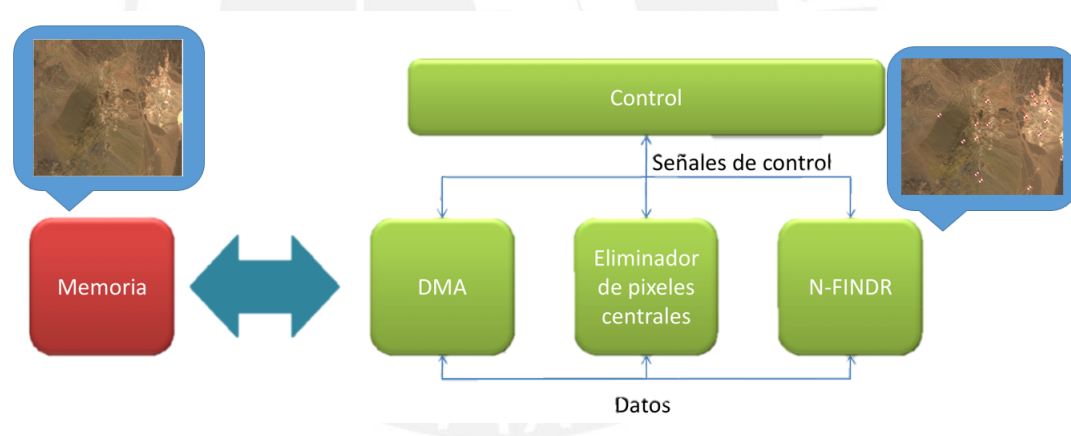


Figura 3.7: Diagrama de bloques planteado para la arquitectura hardware del sistema.

3.2.2. Módulo N-FINDR

Este módulo tiene como finalidad la recepción de los primeros endmembers y el análisis de los píxeles mediante el algoritmo N-FINDR. La etapa más compleja de este módulo es el cálculo de la determinante por ello en este trabajo se plantea dos mejoras para este proceso. Esta subsección está organizada de la siguiente manera. El apartado 3.2.2.1 presenta el algoritmo utilizado para el cálculo de la determinante. El apartado 3.2.2.2 describe el método usado en el diseño con el objetivo de incrementar la máxima frecuencia de operación del algoritmo. El apartado 3.2.2.3

muestra el diseño de la arquitectura hardware de módulo de cálculo de la determinante. Finalmente, el apartado 3.2.2.4 se presenta el diagrama de bloques de módulo N-FINDR.

3.2.2.1. Algoritmo del cálculo de la determinante

El cálculo de la determinante se puede realizar mediante diversas maneras pero los métodos más conocidos para este cálculo son mediante el teorema de Laplace o el método de triangulación. El primer método, el teorema de Laplace, posee un costo computacional elevado; mientras que, el método de triangulación requiere un menor costo computacional. Por ello, en el presente trabajo se implementa el método de triangulación como en la referencia [1]. El algoritmo de triangulación (ver Algoritmo 1) consta de 2 etapas. La primera etapa se encarga de transformar la matriz actual a una matriz triangular mediante propiedades matriciales. Esta transformación se realiza restando un factor de la columna superior a las columnas inferiores para así conseguir anular los términos no deseados. Sin embargo, para que este factor exista es necesario asegurar que el elemento de la columna que pertenece a la diagonal principal sea diferente a cero. En la segunda etapa se multiplica todos los elementos de la diagonal principal para obtener así la determinante. Finalmente, para obtener un valor proporcional al volumen del simplex es necesario aplicar la función valor absoluto a la determinante calculada. La fórmula para el cálculo del volumen está dada por la ecuación 3.1 donde d_i es el elemento i de la diagonal principal.

$$V(E) = \left| \prod_{i=1}^p d_i \right| = \prod_{i=1}^p |d_i| \quad (3.1)$$

3.2.3. Método propuesto

El uso de multiplicadores y divisores como se observa en el paso 1.B y 2 del Algoritmo 1 tiene como consecuencia una baja frecuencia de operación. Debido a esto, es necesario buscar un algoritmo equivalente, al mencionado, que evite el uso de multiplicadores y/o divisores, los cuales generan una baja frecuencia de operación. El problema encontrado en el paso 1.B trae como consecuencia, además del inconveniente ya descrito anteriormente, un gran margen de error cuando se trabaja con la codificación de número con signo sin decimales. Para evitar este margen de error elevado junto al problema de la baja frecuencia de operación, en este trabajo se propone un método basado en la multiplicación y división solo con potencias de 2 en conjunto con sumadores y restadores. El método planteado se observa en la figura 3.8 y tiene como primer paso el cálculo del valor absoluto del término que se desea eliminar así como del término que se utilizará para eliminar. Luego se calcula la distancia en bits “k” de los bits más significativos de

Algoritmo 1 Método de triangulación para el cálculo del valor absoluto de la determinante de una matriz cuadrada de tamaño p [1].

```
//Step 1
for ( $i = 1; i < p; i = i + 1$ ) do
  //Step 1.A
  if ( $A[i][i] = 0$ ) then
     $j = i + 1;$ 
    while ( $A[i][i] = 0$ ) do
      if ( $A[j][i] \neq 0$ ) then
         $temp = A[i];$ 
         $A[i] = A[j];$ 
         $A[j] = temp;$ 
      end if
       $j = j + 1$ 
      if ( $j > p$ ) then
        return 0;
      end if
    end while
  end if
  //Step 1.B
  for ( $j = i + 1; j \leq p; j = j + 1$ ) do
     $k = A[j][i] / A[i][i];$ 
     $A[j] = A[j] - k * A[i];$ 
  end for
end for
//Step 2
 $abs\_det = 1;$ 
for ( $i = 1; i \leq p; i = i + 1$ ) do
   $abs\_det = abs\_det * |A[i][i]|;$ 
end for
return  $abs\_det;$ 
```

entre ambos términos. Posteriormente, se multiplica 2^k a la columna del término que se utiliza para eliminar y se resta con la columna del término a eliminar. Este proceso se repite hasta eliminar el término que se desea.

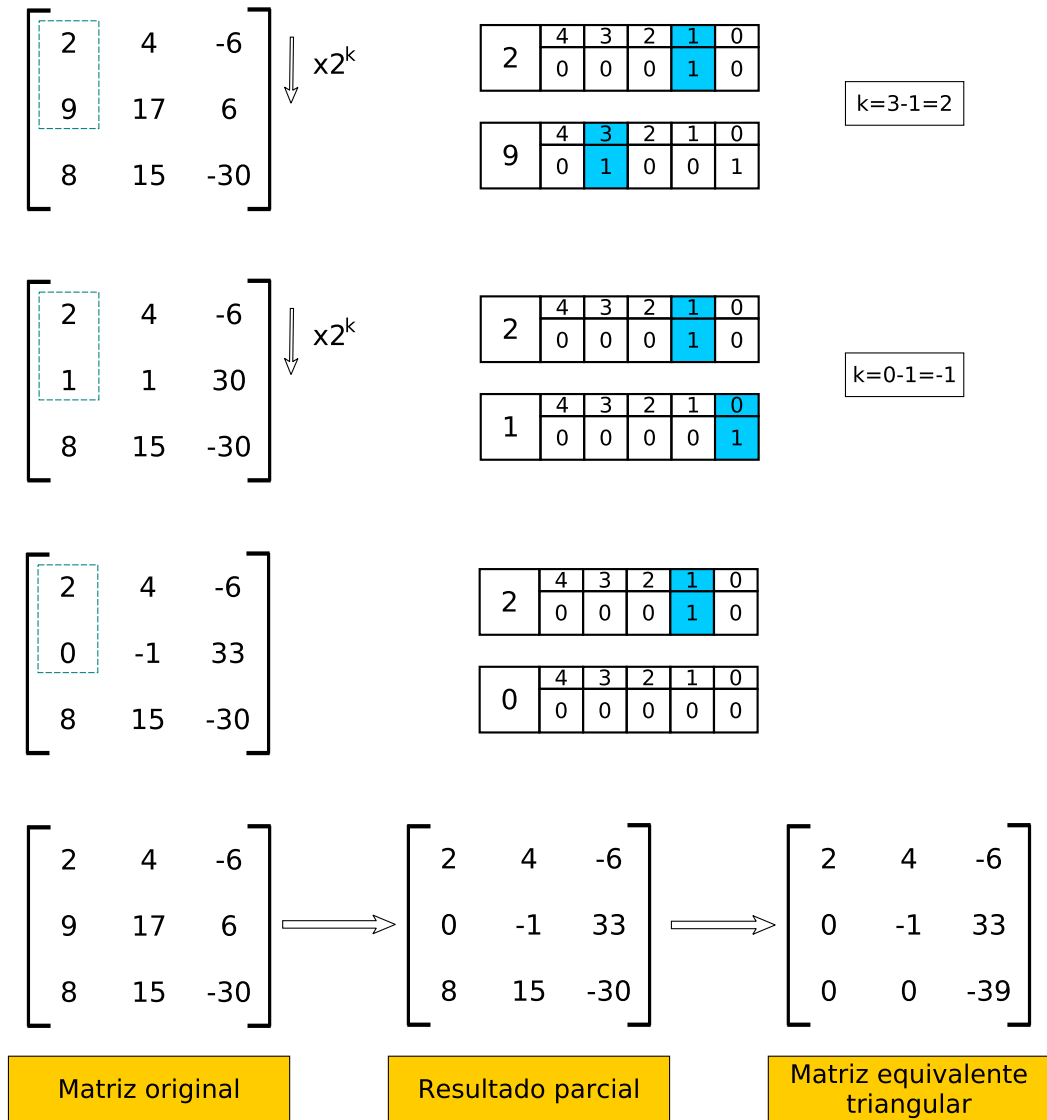


Figura 3.8: Ejemplo del método propuesto para cancelar términos.

En el paso 2, el principal problema que se presenta es la excesiva cantidad de bits que se requiere para representar el valor de la determinante, por ejemplo para el análisis de la imagen Cuprite, donde se desea detectar 21 endmembers el valor de la determinante alcanzó un valor de $9,8512 \times 10^{68}$ que requeriría un total de 231 bits para ser representado en codificación número con signo o precisión doble en codificación de punto flotante (64 bits), lo cual no es práctico para un FPGA de gama media. Debido a esto, es necesario aplicar la función logaritmo al valor absoluto de la determinante con la finalidad de facilitar el cálculo de la multiplicación de la diagonal principal

y para reducir la cantidad de bits requeridos para la representación de la determinante como se muestra en la ecuación 3.2.

$$\log_2(V(E)) = \log_2\left(\prod_{i=1}^p |d_i|\right) = \sum_{i=1}^p \log_2(|d_i|) \quad (3.2)$$

3.2.4. Diseño del módulo de cálculo de área

El módulo del cálculo del área se divide en 7 etapas (Ver Figura 3.9). La etapa de registro de la matriz se encarga de guardar y modificar la matriz actual. Por ello, recibe todas las posibles modificaciones que podría sufrir la matriz y además posee una señal de habilitación para actualizar la matriz. La etapa de cambio de columna tiene como finalidad verificar que el término que pertenece a la diagonal principal sea diferente a 0. Mientras que, la etapa de procesamiento realiza el paso 1.B del Algoritmo 1 mediante el método explicado en el apartado 3.2.2.3.

Por otro lado, el índice i de la etapa de índices tiene como función representar el índice del lazo de todo el paso 1. La etapa de selección tiene como objetivo extraer de la matriz la fila que se están analizando así como la fila que se debería de cambiar en el paso 1.A. La etapa de multiplicación realiza el paso 2 del Algoritmo 1 mediante el uso del logaritmo. Finalmente, la etapa de la máquina de estados genera todas las señales de control para seguir el flujo del algoritmo.

3.2.5. Diseño del módulo N-FINDR

El módulo N-FINDR se compone de cinco etapas (Ver Figura 3.10). La etapa de registro de la matriz guarda el valor de las bandas de los actuales endmembers y va reemplazando cada uno de ellos por el valor de las bandas del pixel que se está analizando. La etapa de cálculo de la determinante recalcula el área del simplex considerando la nueva posición que ha tomado el pixel dentro de la matriz. La etapa de análisis compara si el nuevo volumen es mayor al volumen generado por los actuales endmembers y genera un valor lógico de uno en la señal “cambio” si es que se ha detectado esta condición. La etapa de registro de posiciones guarda las posiciones de los actuales endmembers. Finalmente, la etapa de la máquina de estados controla el flujo de todo el módulo y posee la entrada “inicializar” para guardar la primera selección aleatoria de los endmembers, así como la señal “nuevo dato”, la cual indica que se tiene un nuevo pixel para analizar.

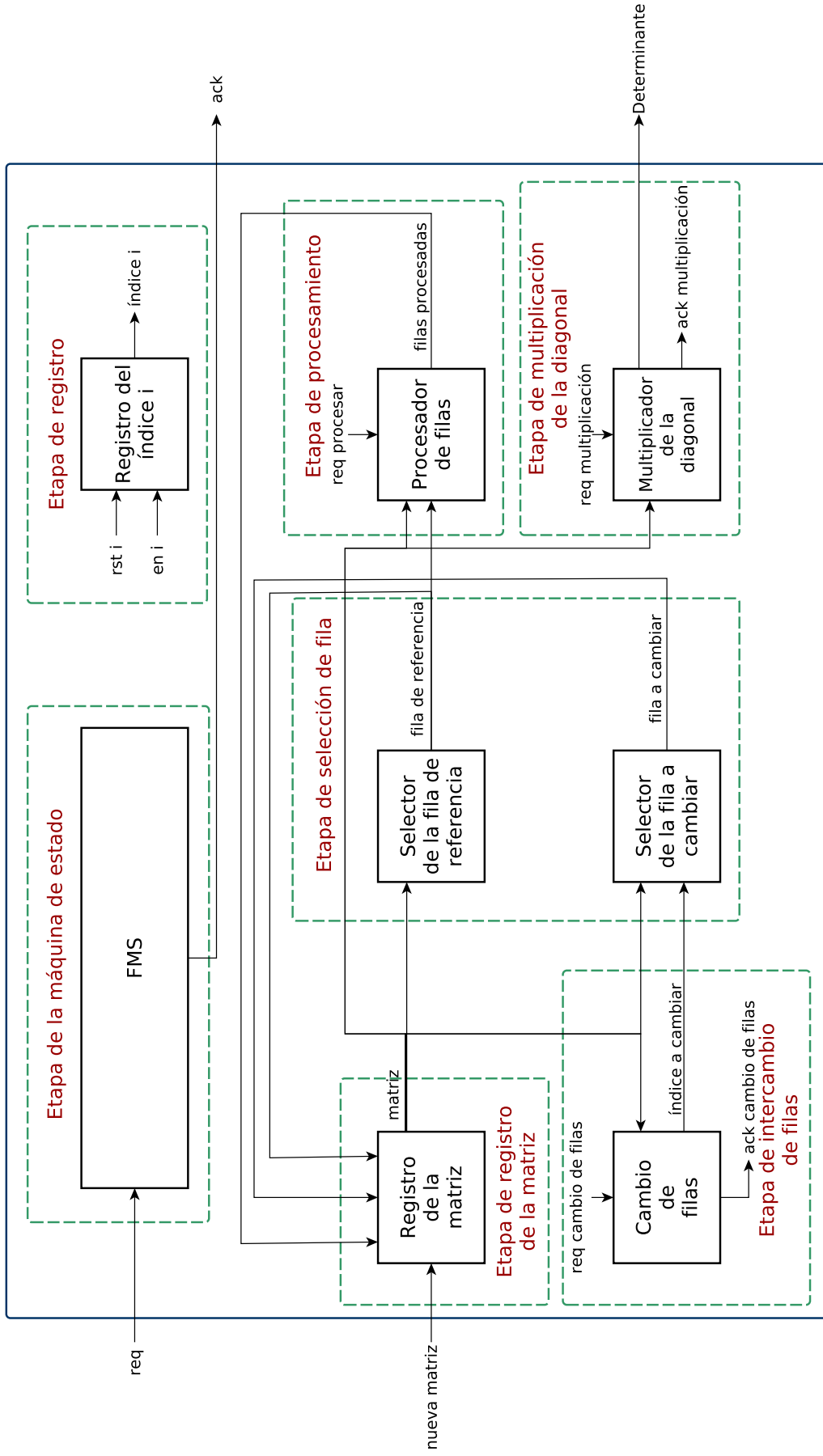


Figura 3.9: Diagrama de bloques del módulo de cálculo de área.

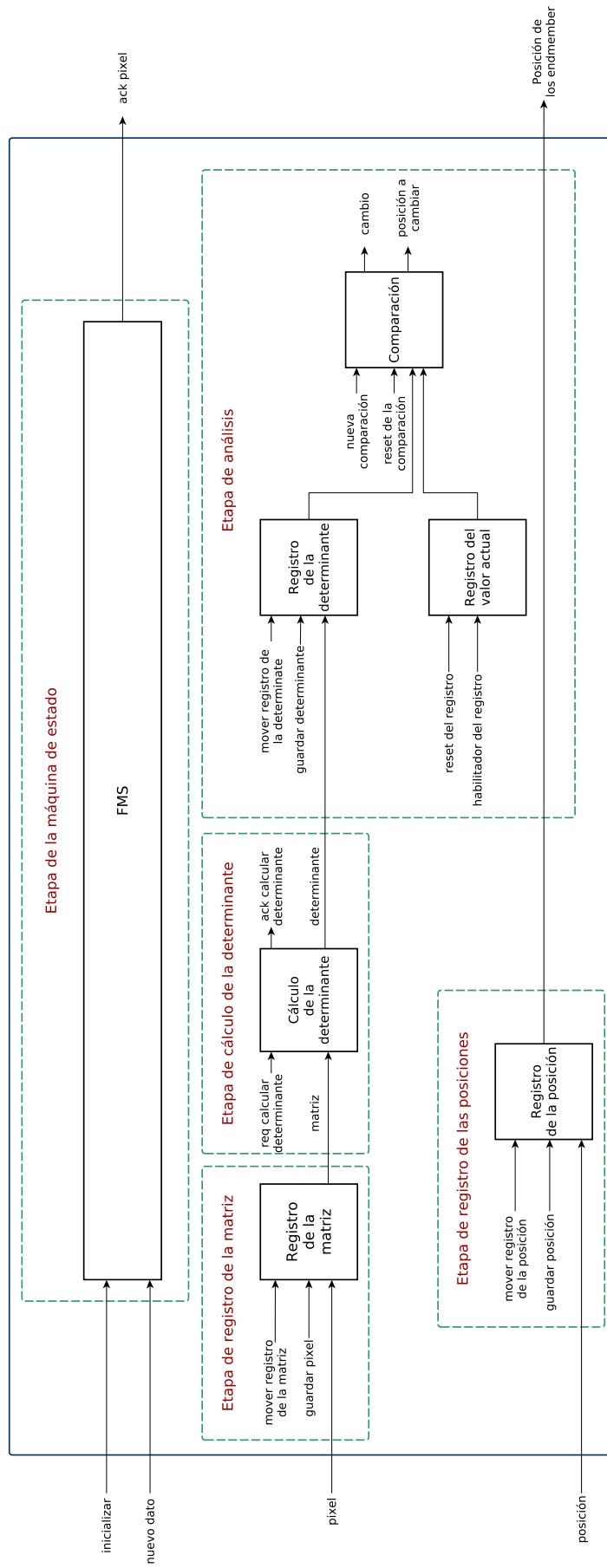


Figura 3.10: Diagrama de bloques del módulo N-FINDR.

3.2.6. Módulo DMA

Para el desarrollo del módulo DMA se consideró el uso de una memoria SRAM la cual tiene un tamaño de palabra de 32 bits y una capacidad mínima de 9.8MB. Para el caso de 21 endmembers, este módulo extrae los datos de la memoria y los agrupa de 20 en 20. En la Figura 3.11 se observa el diagrama de bloques del módulo DMA el cual consta 4 etapas. La etapa de la dirección inicial tiene como objetivo ubicar la posición de inicio de lectura para ello es necesario multiplicar la posición del pixel por 20, esto se logra sumando esta posición desplazada 4 bits a la derecha(x16) con la misma posición desplazada 2 bits a la derecha(x4). La etapa de cuenta está compuesta por un contador el cual barre las 20 posiciones de memoria donde esta los datos del pixel. A continuación la etapa de registro almacena los valores que son leídos de la memoria y activa la señal “lleno” cuando se llena completamente, esto significa que la lectura del pixel está completa. Por último, la etapa de la máquina de estados tiene como función manejar el flujo del módulo, logrando así que primero se fija la posición inicial y luego se recorre las direcciones de memoria del pixel mientras que se guardan los datos leídos en el registro serie hasta que se llene.

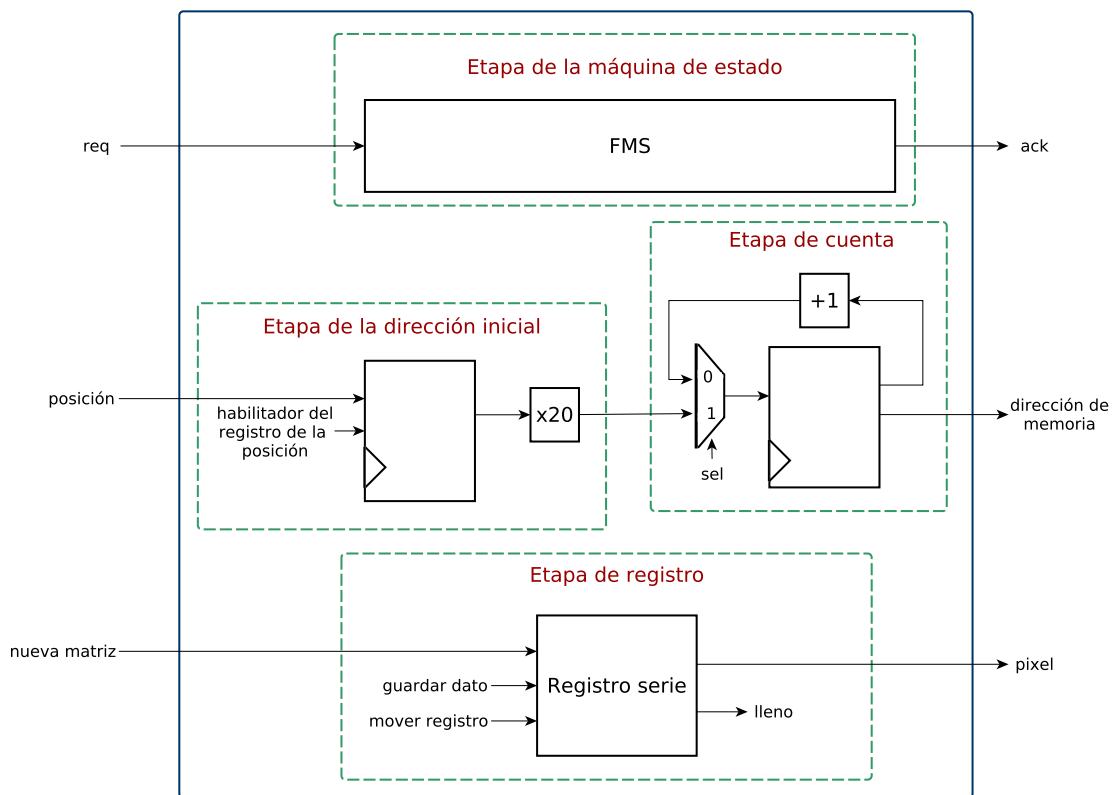


Figura 3.11: Diagrama de bloques del módulo DMA .

3.2.7. Módulo control

Este módulo se encarga de generar las señales de control para el bloque DMA y para el bloque N-FINDR. Con el fin de realizar las tareas en paralelo, este bloque envía a procesar un pixel al módulo N-FINDR mientras que ordena al módulo DMA leer el siguiente pixel. Como se observa en la Figura 3.12, este módulo posee dos contadores en la etapa de generación de la posición. El generador de números aleatorios de esta etapa entrega la primera selección de los endmembers mientras que el bloque contador actúa como un contador genérico el cual recorre todos los pixeles para su análisis en el bloque N-FINDR. La etapa de registro tiene como función retener la información del pixel actual, mientras que es procesado. La etapa de la máquina de estados tiene como salida las señales de habilitación y de reinicio síncrono de los contadores, además controla el flujo de los demás bloques analizando si han terminado su labor a través de las señales Ack N-FINDR y Ack memoria, activando las señales de req N-FINDR y req memoria para solicitar el funcionamiento de estos módulos.

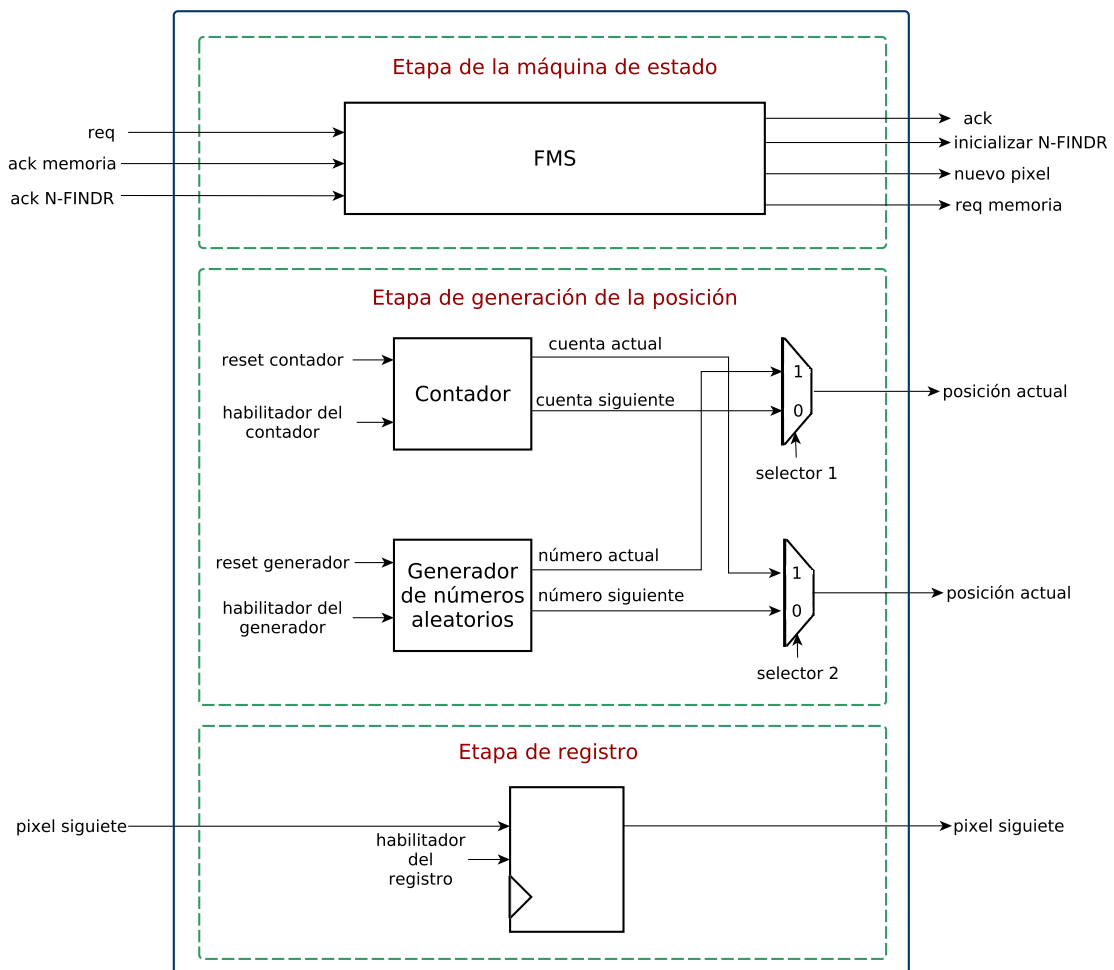


Figura 3.12: Diagrama de bloques del módulo control.

3.2.8. Diagrama de integración completa del módulo de extracción de endmembers

En la Figura 3.13 se observa la interconexión entre los tres bloques anteriormente descritos.

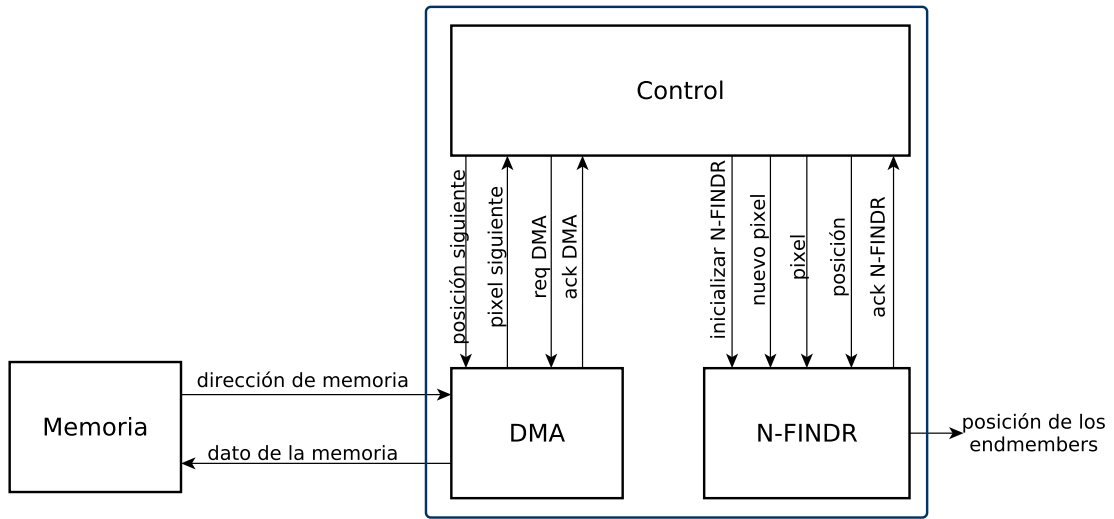


Figura 3.13: Diagrama de bloques del módulo de extracción de endmember .

3.2.9. Análisis de potencia de la codificación de las máquinas de estados

Para determinar la codificación que se debe usar en cada máquina de estado de la arquitectura se realizó un análisis de potencia utilizando cuatro diferentes tipos de codificación: binario, gray, Johnson y one hot. Para esto se realizó simulaciones de los circuitos mediante el software ISim con la finalidad de guardar los cambios producidos en las señales en un archivo de extensión saif. Posteriormente utilizando el software XPower Analyzer de la empresa Xilinx se estimó la potencia dinámica utilizando el archivo saif creado en el paso anterior. Los resultados del estudio se pueden observar en esta tabla 3.1. En la tabla se observa que la codificación adecuada para los bloques de cálculo de la determinante y DMA es one hot; mientras que, para los bloques N-FINDR y control se debe de usar la codificación Johnson.

Tabla 3.1: Resultados de la potencia dinámica disipada por los bloques. Para el bloque de cálculo de la determinante y el bloque N-FINDR se realizó el análisis con una reducción dimensional a dos bandas.

Bloque	Binario	Gray	Jhonson	One hot
Cálculo de la determinante	173mW	170mW	179mW	168mW
N-FINDR	147mW	143mW	136mW	149mW
DMA	49mW	52mW	52mW	26mW
Control	87mW	81mW	78mW	82mW

Capítulo 4

Resultados

Nuestra recompensa se encuentra en el esfuerzo
y no en el resultado.

Un esfuerzo total es una victoria completa.

Mahatma Gandhi

Las arquitecturas mostradas anteriormente fueron descritas mediante el lenguaje de descripción de hardware Verilog y simuladas mediante el programa ModelSim. Además, el diseño fue sintetizado para un FPGA Virtex 4 a través del programa ISE de la compañía Xilinx. En esta sección se muestra los resultados obtenidos de la síntesis y simulación de la arquitectura propuesta y esta organizada de la siguiente manera. La sección 4.1 muestra el análisis de error utilizando los métodos planteados en la sección 3.2.2.2. La sección 4.2 presenta los resultados de la simulación de los diversos bloques que involucrados en el desarrollo del extractor de endmember. Finalmente, la sección 4.3 muestra los resultados de síntesis de la arquitectura.

4.1. Análisis del error generado en la arquitectura

En esta subsección se analiza el error generado por el uso de los métodos planteados en los pasos 1.B y 2 del Algoritmo 1.

Para el paso 1.B se planteó un método que evita el uso de multiplicadores y divisores para obtener una mayor frecuencia de operación. Sin embargo, el uso de la codificación de números sin signo generó un error debido a que se perdían los decimales al dividir entre las potencias de 2. El error promedio generado por la pérdida de los decimales fue de 2.5789 %, en la diagonal principal.

En el paso 2, el error es generado por la aproximación del logaritmo a un número en punto fijo. Como se observa en la Figura 4.1, si se aumenta la cantidad de bits el error en la aproximación disminuye. Sin embargo, se observa que no existe una mejora sustancial con respecto al error después del uso de 10 bits.

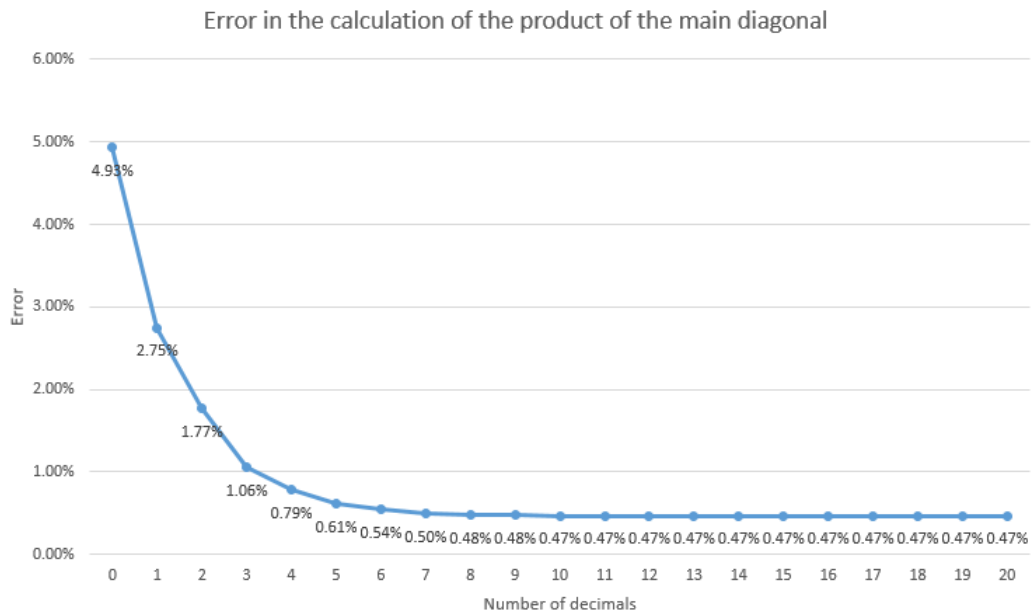


Figura 4.1: Gráfica del error en el cálculo de la multiplicación de la diagonal principal vs la cantidad de bits decimales.

En la Tabla 4.1 se muestra el desempeño global del módulo de cálculo de la determinante mediante cinco muestras que se tomaron de la imagen Cuprite. Además se realizó un histograma del error generado en la arquitectura y esta se muestra en la Figura 4.2. Mediante este análisis se obtuvo como media y desviación estándar del error 1.2342 % y 0.3524 % respectivamente.

Tabla 4.1: Comparación entre el valor calculado mediante el diseño propuesto y el valor real calculado mediante el entorno de programación Matlab.

Valor calculado mediante la arquitectura hardware	201.375	218	192.375	221.25	208.875
Valor real calculado mediante Matlab	202.895	220.036	194.375	223.617	210.901

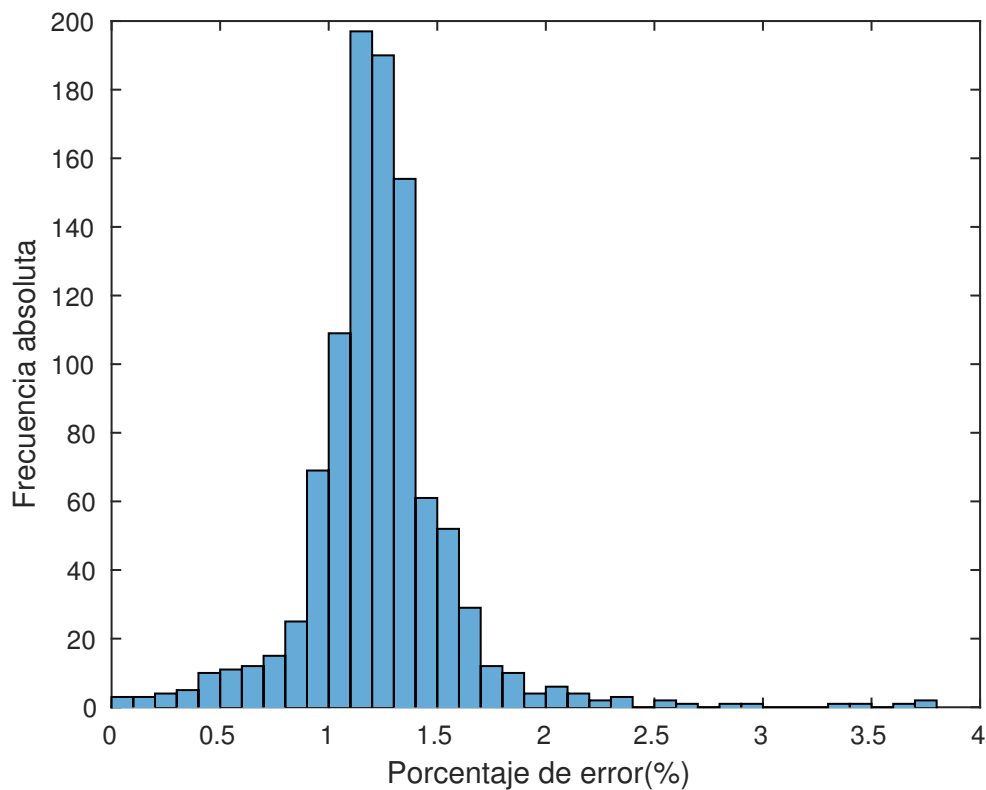


Figura 4.2: Histograma del error generado en la arquitectura propuesta usando como muestra mil matrices de 21x21 recolectadas de la imagen Cuprite.

4.2. Simulaciones de los módulos de las arquitecturas diseñadas

Para la verificación de los módulos diseñados se desarrolló diversas simulaciones por testbench y sus resultados se muestran en la presente sección. Esta sección está organizada de la siguiente manera. En el apartado 4.2.1 se muestra la simulación del bloque N-FINDR y de los sub bloques que están involucrados. En el apartado 4.2.2 se presenta los resultados de la simulación de DMA. El apartado 4.3 muestra la simulación de las señales de control generadas por bloque control. Finalmente, en el apartado 4.4 se presenta la simulación del sistema de extracción de endmembers.

4.3. Simulación del bloque N-FINDR

El circuito crítico del módulo N-FINDR y que determina el rendimiento de la arquitectura es el sub bloque de cálculo de la determinante. En la Figura 4.3 se observa la simulación de este sub bloque dividida en dos secciones. En la primera sección, señales de la entidad, se observa todas las señales de entrada y salida del módulo donde las señales de “req” y “ack” son de control e

indican el inicio y el fin del procesamiento de la arquitectura. Por otro lado, la señal matriz es un vector la cual tiene toda la información de la matriz a procesar. Finalmente, la señal “determinante” muestra el resultado de la operación. Por otro lado, en la sección “señales internas” se encuentran las señales de control internas como las señales req_intercambiador, ack_intercambiador, req_procesar, ack_procesar, req_multiplicar_diagonal y ack_multiplicar_diagonal las cuales estan asociados al inicio y fin del paso 1.A, 1.B y 2 respectivamente del Algoritmo 1. Asimismo, la señal “índice i” indica la columna actual de referencia; es decir, apartir de la columna “i” aún falta realizar el procesamiento.

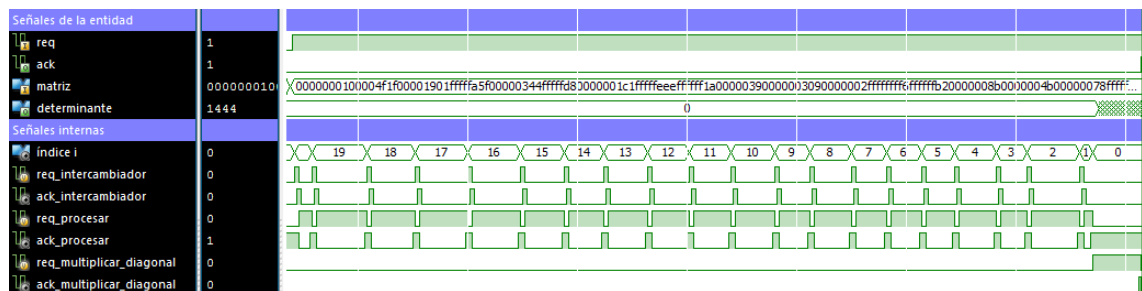
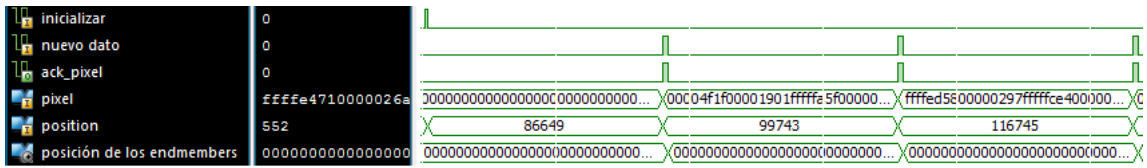
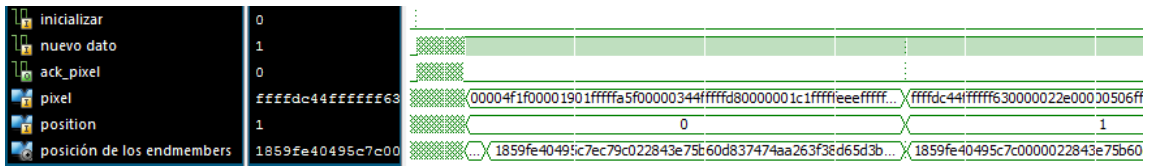


Figura 4.3: Simulación del módulo de cálculo de la determinante. El resultado que se muestra al finalizar la simulación en la señal “determinante” esta en una codificación de punto fijo y posee 3 bit de decimales, por ende, la determinante es 180.5.

El funcionamiento del bloque N-FINDR se divide en dos etapas. La primera etapa es la inicialización de la matrix, donde se selecciona aleatoriamente los primeros endmembers y se guarda su información dentro de la matriz. En la Figura 4.4a se muestra este proceso el cual empieza cuando la señal “inicializar” se activa y el sistema empieza a guardar los pixeles cuando detecta un nuevo dato a través de la señal “nuevo dato” . La información del pixel guardado esta compuesto por dos datos: la información en las bandas del pixel y su posición dentro de la imagen, las cuales están en las señales “pixel” y “posicion”, respectivamente. Mientras que, la segunda etapa es el análisis del pixel y consiste en reemplazar cada uno de los endmembers por el pixel a analizar y recalculer el volumen formado para determinar si este cambio favorece o no al incremento del volumen. En la Figura 4.4b se observa el proceso de análisis, el cual inicia cuando la señal “nuevo dato” se activa. Seguidamente, la matriz empieza a cambiar debido al ingreso del pixel a analizar y se recalcula el volumen variando la posición de este dato dentro de la matriz. Finalmente, al terminar el análisis de todas las posibles combinaciones, se compara todos los resultados con el valor actual de la determinante para comprobar si el ingreso de este dato influye positivamente al incremento del volumen. En la figura 4.4b se muestra que el pixel si cumple la condición y reemplaza a uno de los endmembers lo cual se observa cuando la señal de “ack pixel”



(a) Etapa de inicialización de la matriz



(b) Etapa de análisis de píxeles

Figura 4.4: Simulación del bloque N-FINDR

se activa y la posición de los endmembers cambia.

4.4. Simulación del bloque DMA

El bloque DMA interactúa con una memoria SRAM, por ende, para su simulación se infirió una memoria de este tipo. En la Figura 4.5 se muestra las señales “dirección de memoria” y “dato de la memoria” que sirven de interconexión con la memoria. La primera señal se relaciona con la dirección del dato que se va a leer desde la memoria; mientras que, la segunda señal es el dato que se extrae desde la dirección referida. Por otra parte, la señal “posición del pixel” indica el pixel que se debe de leer desde la memoria y el resultado de la lectura se observa en la señal “pixel”. Para leer un pixel, en este trabajo, es necesario leer 20 espacios de memoria debido a que cada pixel esta compuesto por 20 bandas.

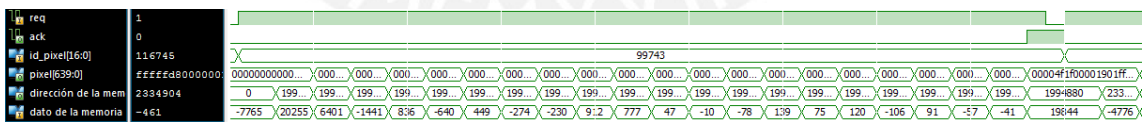


Figura 4.5: Simulación del módulo DMA.

4.5. Simulación del bloque Control

En la Figura 4.6 se muestra las señales que recibe y envía el bloque control a los bloques N-FINDR y DMA. Asimismo, se observa las señales de control “req” y “ack” los cuales determinan el inicio y fin del proceso de extracción de endmembers, respectivamente. En la sección “N-FINDR” y “DMA” se observan las señales de interconexión con los bloques

referidos. Además, se muestra que el bloque de control esta inicializando la matriz de los endmembers en el bloque N-FINDR, mientras que, el bloque DMA esta leyendo la memoria para enviarle la primera selección de endmembers al bloque N-FINDR.

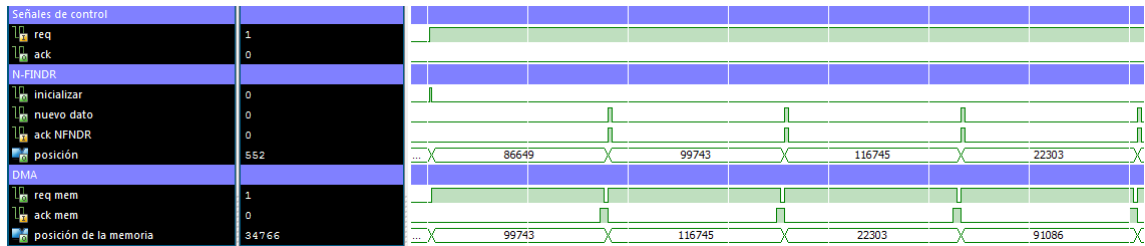
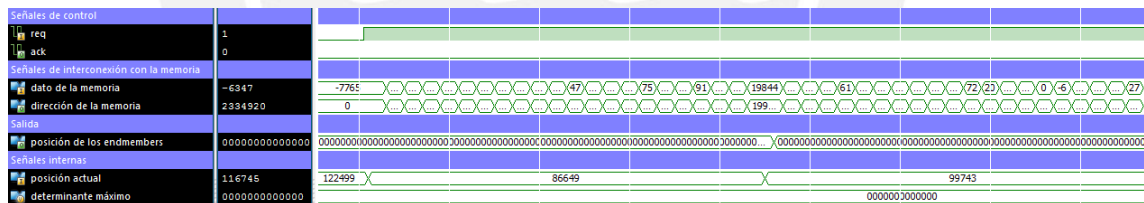


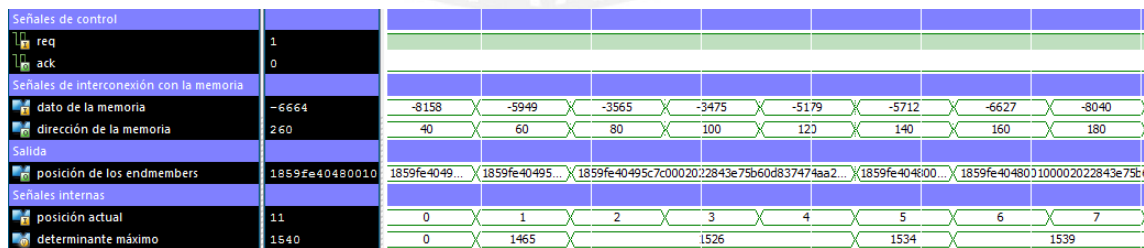
Figura 4.6: Simulación del módulo control.

4.6. Simulación del sistema de extracción de endmembers

En la Figura 4.7 se observa la simulación del sistema completo. En la primera sección, Figura 4.7a, se observa el proceso de selección aleatoria de los primeros endmembers. En la sección “señales internas” está la señal “posición actual” la cual indica la posición aleatoria del pixel que se esta leyendo desde la memoria y guardando en la matriz del bloque N-FINDR. En el segundo gráfico, en la Figura 4.7b se observa el proceso de análisis para los primeros ocho pixeles y la señal “determinante máximo” muestra el volumen actual formado por los endmembers.



(a) Selección aleatoria de los primeros endmembers



(b) Simulación del sistema para los primeros ocho pixeles

Figura 4.7: Simulación del sistema completo

4.7. Evaluación de los endmembers

Mediante la arquitectura propuesta se encontraron varios conjuntos de endmembers. Sin embargo, para cuantizar la calidad de los endmembers extraídos es necesario compararlos con librería espectral de la USGS¹. Con este fin, en la Figura 4.8 se observa esta comparación mediante cinco minerales presentes en la imagen Cuprite según [27].

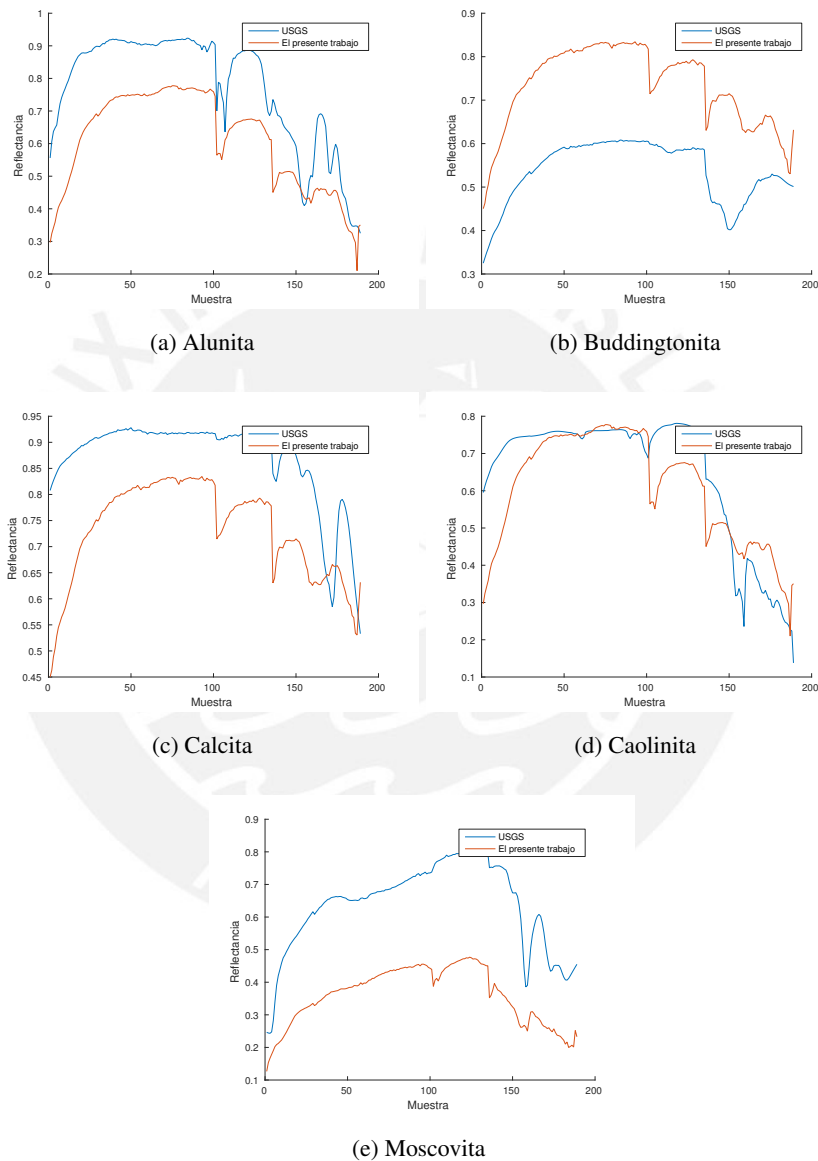


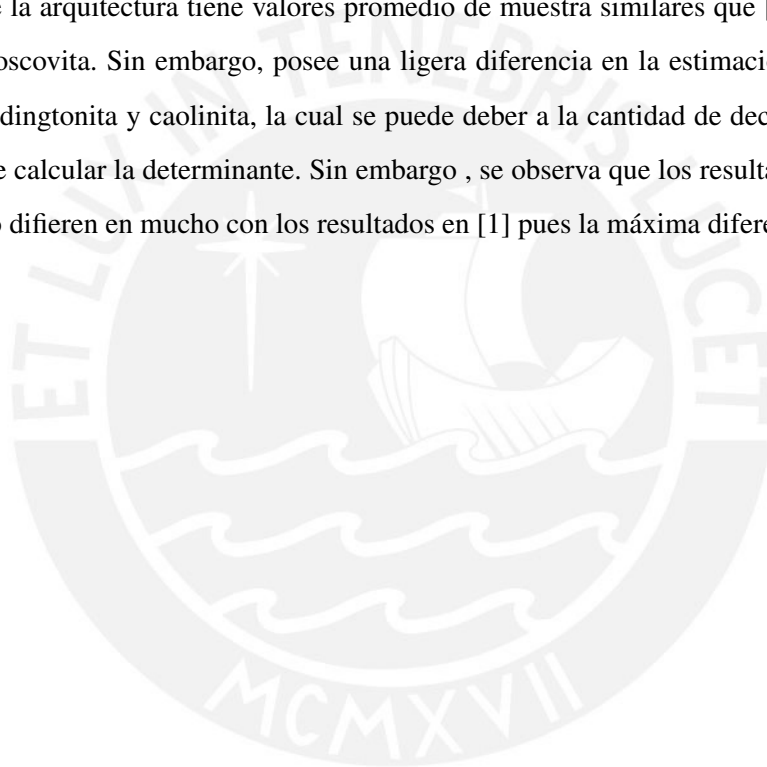
Figura 4.8: Comparación de los endmembers encontrados mediante la arquitectura usando el algoritmo N-FINDR y los minerales proporcionados por la librería espectral de la USGS. Además, para evitar el error generado por las bandas de baja relación señal ruido se descartaron estas del análisis.

¹United States Geological Survey

Con la finalidad de medir la similitud entre los endmembers detectados y los minerales presentes en la imagen Cuprite se usó el ángulo espectral entre estos dos vectores. El ángulo espectral se calcula a través de la ecuación 4.1:

$$AE(E, M) = \arccos \frac{\sum_{i=1}^k E_k \times M_k}{\sqrt{\sum_{i=1}^k E_k^2} \times \sqrt{\sum_{i=1}^k M_k^2}} \quad (4.1)$$

Donde E es el espectro del endmember, M es el espectro del mineral y AE es el ángulo espectral. Si el valor del ángulo espectral se acerca a cero esto indica que los vectores presentan similitud; mientras que, si el resultado es cercano a 90° indica que los vectores son totalmente diferentes. En la Figura 4.9, se observa los valores de las muestras tomadas y en la tabla 4.2 se observa que la arquitectura tiene valores promedio de muestra similares que [1] en los minerales calcita y moscovita. Sin embargo, posee una ligera diferencia en la estimación de los minerales alunita, buddingtonita y caolinita, la cual se puede deber a la cantidad de decimales utilizados al momento de calcular la determinante. Sin embargo, se observa que los resultados presentados en esta tesis no difieren en mucho con los resultados en [1] pues la máxima diferencia es menor a 2°.



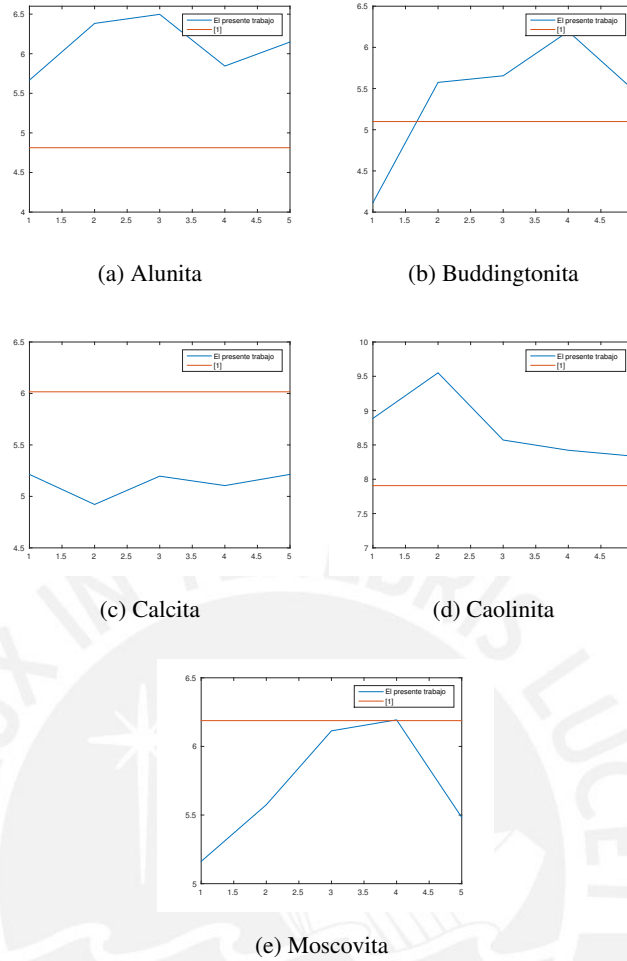


Figura 4.9: Comparación del resultado del ángulo espectral del trabajo y [1]. En las ascisas se tiene el número muestra; mientras que, en las ordenadas el ángulo espectral en grados sexagesimales.

Tabla 4.2: Comparación entre el promedio del ángulo espectral obtenido y la referencia [1].

Mineral	Promedio	[1]
Alunita	6.327	4.812
Buddingtonita	5.935	5.099
Calcita	5.367	6.016
Caolinita	9.127	7.906
Moscovita	5.937	6.187

4.8. Resultados de la síntesis de la arquitectura diseñada

La arquitectura propuesta fue sintetizada utilizando el software ISE de la compañía Xilinx para el FPGA 4vfx60ff672-12 de la familia Virtex-4. Aunque la arquitectura supero la capacidad máxima de este FPGA en cantidad de slices el programa ISE estimo la cantidad de slices necesarios para el diseño así como la frecuencia de operación máxima de este. El resultado se muestra en el Tabla 4.3.

Tabla 4.3: Resultados de la síntesis global de la arquitectura sobre un FPGA Virtex 4.

	Dr. Carlos González[1]	El presente trabajo de tesis
Frecuencia máxima de operación(MHZ)	42.3	69.4
Número de slices utilizados	25060	124900

Sin embargo, el modelo Virtex 4 modelo XC4VFX60 utilizado en la referencia [1] cuenta con 25,280 slices, siendo esto una limitante para la arquitectura diseñada en el presente trabajo, dado que para el diseño realizado en el presente trabajo se empleó el paralelismo de operaciones, lo cual incrementó el uso de recursos del FPGA haciendo que la cantidad de slices se incrementara de manera significativa, respecto de la referencia [1]. La frecuencia de operación de 69.4 Mhz es un valor estimado por el software ISE como se observa en la Figura 4.10 debido a que se excede el número de slices disponibles. Es por ello que se realizó un análisis de mediante hojas de datos de las familias Virtex 4, Virtex 5, Virtex 6 y Virtex 7 [28, 29, 30, 31], donde se pudo identificar que el FPGA Virtex 7 modelo XC7VX980T es el más idoneo para soportar el tamaño de la arquitectura diseñada, obteniendo una frecuencia de 112.819MHz. Asimismo, se realizó la síntesis en FPGA Virtex 5, Virtex 6 para verificar si existe una mejora en frecuencia conforme se utiliza un FPGA de mayor tecnología, lo cual puede ser observado en la figura 4.11.

```

=====
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
-----
-----+-----+-----+
Clock Signal          | Clock buffer (FF name) | Load |
-----+-----+-----+
clk                   | BUFGP                   | 59613 |
-----+-----+-----+

Timing Summary:
-----
Speed Grade: -12

Minimum period: 14.407ns (Maximum Frequency: 69.413MHz)
Minimum input arrival time before clock: 3.665ns
Maximum output required time after clock: 3.806ns
Maximum combinational path delay: No path found

```

Figura 4.10: Reporte de tiempos y frecuencia de la síntesis en un FPGA Virtex 4.

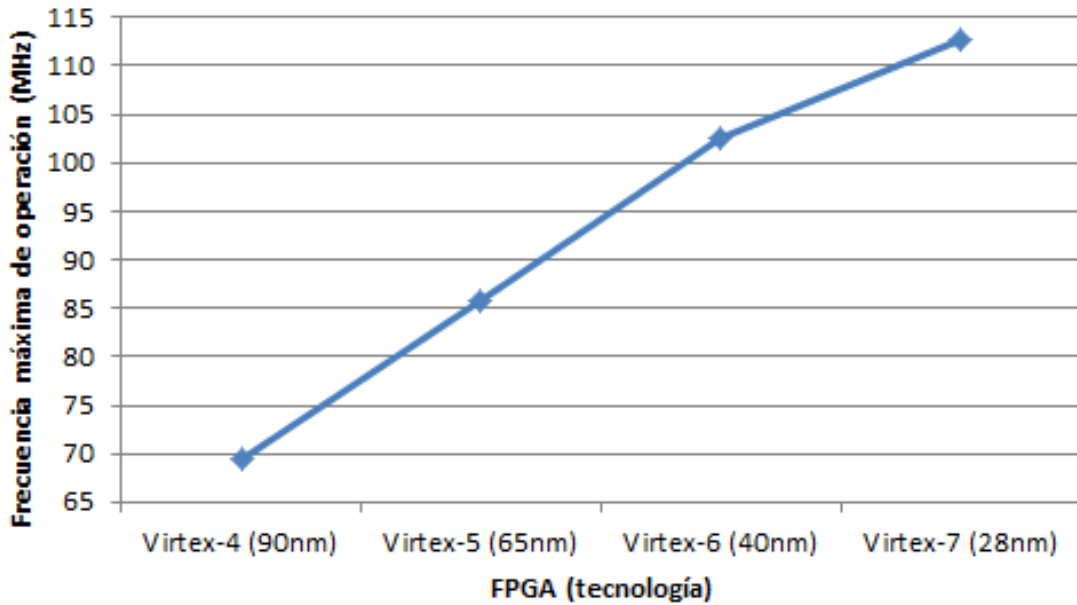


Figura 4.11: Gráfica de tecnología vs frecuencia.

En la referencia [1], se utilizó un FPGA Virtex 4 por ello se buscó sintetizar en un FPGA similar para hacer la correcta comparación. Sin embargo, la arquitectura diseñada superó la cantidad de slices que dicho FPGA tiene disponibles, debido a la utilización del paralelismo de

operaciones con el fin de incrementar la frecuencia de operación y buscar alcanzar el procesamiento en tiempo real. Sin embargo, utilizando el FPGA Virtex 4 no se logró alcanzar la síntesis por falta de slices disponibles, debido a que dentro de la arquitectura se implementó un bloque procesador por cada endmember a procesar. Es decir si se procesan 5 endmembers se tienen 5 procesadores y si se procesan 21 endmembers se tendrán 21 procesadores. Es por ello, que la presente arquitectura no es soportada por la Virtex 4 al procesar 21 endmember, pero si se procesan 8 endmembers como máximo si encajaría en una Virtex 4 sin problemas. Es por ello que se utilizó un FPGA Virtex 7 modelo XC7VX980T, pero esto hace que no sea comparable los resultados respecto de la referencia [1], pero si se logró verificar el correcto funcionamiento de la arquitectura diseñada obteniendo una mejora en frecuencia estimada del 64 %(69.4MHz) sobre una Virtex 4 y por otro lado utilizando una virtex 7 se obtiene una frecuencia exacta de 112.819MHz.

La síntesis de arquitectura se encuentra detallada en la Tabla 4.3, la fue realizada para procesar 21 endmembers. Sin embargo, la arquitectura propuesta puede adaptarse a cualquier cantidad de endmembers. Los resultados de la síntesis de la propuesta arquitectura según la cantidad de endmembers se muestran en la Tabla 4.4. Se puede observar que existe una proporcionalidad entre el número de slices del FPGA y el número de endmembers a procesar, es por ello, que no era posible sintetizar la arquitectura sobre una Virtex 4.

Tabla 4.4: Resultado de síntesis de la arquitectura en un Virtex 7 modelo C7VX980T.

Cantidad de endmembers	Slices	Frecuencia
5	3,264	128.883MHz
8	8,606	126.850MHz
11	14,960	117.725MHz
13	21,923	117.391MHz
16	36,443	115.694MHz
21	71,096	112.819MHz

Conclusiones

- Con el fin de aprovechar de manera óptima la característica del alto grado de paralelismo de operaciones se realizó el diseño de la arquitectura hardware, bajo esta premisa, alcanzando una frecuencia máxima de operación de 69.4MHz para el dispositivo FPGA Virtex 4 de la compañía Xilinx®. Por temas de seguridad en el funcionamiento de la arquitectura ante posibles variaciones del proceso al momento de realizar una futura implementación se ha restringido el análisis de frecuencia de operación a un 90 % la frecuencia máxima de operación para prevenir posibles variaciones de performance de funcionamiento ante una futura implementación en físico, lo cual permite procesar una secuencia de imágenes hiperespectrales de 350x350 pixeles a una tasa de 1 imagen cada 17.98 segundos.
- La validación del algoritmo N-FINDR y del módulo de extracción de endmembers sobre el entorno de programación MATLAB® permitió verificar que los resultados obtenidos por medio de la arquitectura hardware diseñada son válidos, cumpliendo con el objetivo de diseñar correctamente la arquitectura, bajo los requerimientos de diseño establecidos. Así también, se pudo identificar que la etapa del cálculo de la determinante, es sumamente crítico para el algoritmo, por lo cual se propuso una alternativa de diseño en hardware para mejorar la frecuencia de operación de la arquitectura. Además, el módulo de eliminación de pixeles centrales permitiría mejorar en un 45.7 % el tiempo de procesamiento.
- Un sensor AVIRIS tarda en escanear 512 pixeles un tiempo aproximado de 8.13ms[1]. Esto conduce a la necesidad de procesar un conjunto de datos hiperespectrales de 350 x 350 pixeles (resolución estándar) en un tiempo de 1.98 segundos para alcanzar un procesamiento en tiempo real. El desarrollo de un sistema de procesamiento de imágenes hiperespectrales basados en un FPGA Virtex4, en la presente tesis, no alcanzó el procesamiento en tiempo real, según los resultados mostrados. Dado que aún no es posible alcanzar la velocidad de adquisición de imágenes de un sensor hiperespectral y procesar así

los píxeles a medida que llegan al sistema. En las pruebas de simulación del presente trabajo, sobre el mismo modelo de placa utilizado en la referencia [1], se ha obtenido una frecuencia 64 % veces mayor a la obtenida en la referencia [1]. Sin embargo, el tiempo de procesamiento excede en 16 segundos dicho tiempo de 1.98 segundos necesario para alcanzar el tiempo real. Aunque la inclusión de placas FPGA mucho más modernas de tecnologías mejorarían considerablemente los resultados, como es el caso de una Virtex 7 modelo XC7VX980T, donde la frecuencia alcanzada de la presente arquitectura fue de 112.819MHz. Además, la adición del bloque “Eliminador de píxeles centrales” reduciría la cantidad de píxeles a analizar lo cual permitiría realizar este proceso en un menor tiempo.

- La frecuencia máxima de operación obtenida por la referencia [1], relacionada con la presente tesis, fue de 42.3 MHz, mientras que el presente trabajo obtuvo una frecuencia estimada máxima de operación de 69.4 Mhz sobre un FPGA Virtex 4, es decir, se logró una mejora aproximada del 64 % con respecto a lo obtenido por la referencia [1] en mención. Se cumplió con el propósito de mejorar la frecuencia de operación. Sin embargo, no existe un modelo de FPGA de la familia Virtex-4 que soporte la cantidad de slices requeridos por la arquitectura diseñada, debido a que dentro de la arquitectura se implementó un bloque procesador por cada endmember a procesar. Es decir si se procesan 5 endmembers se tienen 5 procesadores y si se procesan 21 endmembers se tendrán 21 procesadores. Es por ello, que la presente arquitectura no es soportada por la Virtex 4 al procesar 21 endmember, pero si se procesan 8 endmembers como máximo si encajaría en una Virtex 4 sin problemas. Es por ello, que mediante la utilización de las hojas de datos se identificó que el FPGA Virtex 7 soporta la cantidad de slices necesarios por la presente arquitectura, obteniendo una frecuencia de 112.81 Mhz. Con ello se puede concluir que al realizar la síntesis, de la presente arquitectura, sobre un FPGA de mayor tecnología se puede llegar a obtener mejores frecuencias de operación que puedan permitir alcanzar el procesamiento en tiempo real.

Recomendaciones

- El diseño propuesto puede ser implementado sobre una tecnología FPGA Virtex-4 o de mayores prestaciones con el proposito de validar y verificar el funcionamiento de la arquitectura propuesta. Sin embargo, es posible implementarse en versiones más modernas de FPGA para obtener mejoras en la velocidad de procesamiento.
- Se recomienda rediseñar el contador que recorre toda la imagen durante el análisis de los pixeles para que cuente aleatoriamente debido a que los pixeles contiguos poseen una alta correlación, lo cual hace que generen un menor volumen en el análisis, a comparación de un recorrido aleatorio.
- Para prestaciones del trabajo, en entornos espaciales, los circuitos se encuentran inmersos a una alta radiación solar y cósmica, el nivel lógico de los componentes puede cambiar originando problemas en el procesamiento. Por ello es necesario adaptar al circuito propuesto como módulos adicionales que puedan proteger el circuito, con el proposito de evitar un malfuncionamiento por temas de radación cósmica o solar, por ejemplo si se desea utilizar este circuito sobre un satélite espacial.
- Actualmente, el empleo de la salida paralelo, del registro de la matriz en el sub-bloque de cálculo de la determinante obliga a utilizar muchos recursos para seleccionar la columna que se desea procesar. Sin embargo, con el cambio de arquitectura en el diseño de serie-paralelo a serie-serie se esperaría obtener una mayor frecuencia de operación y una menor área.
- Se sugiere desarrollar los bloques restantes de la cadena de desmezclado espectral, como el de estimación de la cantidad de endmembers y estimación de abundancia, para completar todo el procesamiento hiperespectral y proceder a la implementación de esta cadena de desmezclado sobre un FPGA. Con estos bloques adicionales ya es posible su implementación y puesta sobre un dron, con la finalidad de generar los mapas de abundancia de los minerales que se encuentren en la imagen hiperespectral.

- Se aconseja realizar el análisis de potencia del circuito considerando diferentes cantidades de endmembers para determinar la energía que consume el procesamiento. Además, este análisis será determinante para calcular el tiempo de vuelo autónomo que posee el dron.



Bibliografía

- [1] C. González Calvo, *Procesamiento a bordo de imágenes hiperespectrales de la superficie terrestre mediante hardware reconfigurable*. Universidad Complutense de Madrid, 2012.
- [2] D. Manolakis, D. Marden, and G. A. Shaw, “Hyperspectral image processing for automatic target detection applications,” *Lincoln laboratory journal*, vol. 14, no. 1, pp. 79–116, 2003.
- [3] D. M. Cambre, E. Boemo, and E. Todorovich, “Arithmetic operations and their energy consumption in the nios ii embedded processor,” in *Reconfigurable Computing and FPGAs, 2008. ReConFig’08. International Conference on*, pp. 151–156, IEEE, 2008.
- [4] D. Llamocca, C. Carranza, and M. S. Pattichis, “Separable fir filtering in fpga and gpu implementations: Energy, performance, and accuracy considerations.,” in *FPL*, pp. 363–368, IEEE Computer Society, 2011.
- [5] M. Wielage, F. Cholewa, C. Fahnemann, P. Pirsch, and H. Blume, “High performance and low power architectures: Gpu vs. fpga for fast factorized backprojection,” in *Computing and Networking (CANDAR), 2017 Fifth International Symposium on*, pp. 351–357, IEEE, 2017.
- [6] H. R. Zohouri, N. Maruyama, A. Smith, M. Matsuda, and S. Matsuoka, “Evaluating and optimizing opencl kernels for high performance computing with fpgas,” in *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*, pp. 409–420, IEEE, 2016.
- [7] H. Grahn and P. Geladi, *Techniques and applications of hyperspectral image analysis*. John Wiley & Sons, 2007.
- [8] 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*, vol. 2010, no. 1, p. 969806, 2010.

- [9] M. E. Winter, "N-findr: An algorithm for fast autonomous spectral end-member determination in hyperspectral data," in *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pp. 266–275, International Society for Optics and Photonics, 1999.
- [10] J. Bowles, D. Gillis, and P. Palmadesso, "New improvements in the orasis algorithm," in *Aerospace Conference Proceedings, 2000 IEEE*, vol. 3, pp. 293–298, IEEE, 2000.
- [11] L. Miao and H. Qi, "Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization.," *IEEE Trans. Geoscience and Remote Sensing*, vol. 45, no. 3, pp. 765–777, 2007.
- [12] C.-I. Chang, C.-C. Wu, W. Liu, and Y. C. Ouyang, "A new growing method for simplex-based endmember extraction algorithm.," *IEEE Trans. Geoscience and Remote Sensing*, vol. 44, no. 10-1, pp. 2804–2819, 2006.
- [13] J. Gruninger, A. J. Ratkowski, and M. L. Hoke, "The sequential maximum angle convex cone (smacc) endmember model," tech. rep., SPECTRAL SCIENCES INC BURLINGTON MA, 2004.
- [14] J. M. P. Nascimento and J. M. B. Dias, "Vertex component analysis: a fast algorithm to unmix hyperspectral data.," *IEEE Trans. Geoscience and Remote Sensing*, vol. 43, no. 4, pp. 898–910, 2005.
- [15] C. Zhang, Q. Qin, T. Zhang, Y. Sun, and C. Chen, "Endmember extraction from hyperspectral image based on discrete firefly algorithm (ee-dfa)," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 126, pp. 108–119, 2017.
- [16] S. López, P. Horstrand, G. M. Callicó, J. F. López, and R. Sarmiento, "A low-computational-complexity algorithm for hyperspectral endmember extraction: Modified vertex component analysis.," *IEEE Geosci. Remote Sensing Lett.*, vol. 9, no. 3, pp. 502–506, 2012.
- [17] M. Hsueh and C.-I. Chang, "Field programmable gate arrays (fpga) for pixel purity index using blocks of skewers for endmember extraction in hyperspectral imagery.," *IJHPCA*, vol. 22, no. 4, pp. 408–423, 2008.
- [18] C. González, D. Mozos, J. Resano, and A. Plaza, "Fpga implementation of the n-findr algorithm for remotely sensed hyperspectral image analysis.," *IEEE Trans. Geoscience and Remote Sensing*, vol. 50, no. 2, pp. 374–388, 2012.

- [19] C.-I. Chang, W. Xiong, and C.-C. Wu, "Field-programmable gate array design of implementing simplex growing algorithm for hyperspectral endmember extraction.," *IEEE Trans. Geoscience and Remote Sensing*, vol. 51, no. 3-2, pp. 1693–1700, 2013.
- [20] J. Rosário, J. M. Nascimento, and M. Véstias, "Fpga-based architecture for hyperspectral endmember extraction," in *SPIE Remote Sensing*, pp. 924703–924703, International Society for Optics and Photonics, 2014.
- [21] C. Li, L. Gao, A. Plaza, and B. Zhang, "Fpga implementation of a maximum simplex volume algorithm for endmember extraction from remotely sensed hyperspectral images," *Journal of Real-Time Image Processing*, pp. 1–14, 2017.
- [22] D. A. de Sociedad Peruana, "La realidad de la minería ilegal en países amazónicos," *Lima: SPDA*, 2014.
- [23] J. P. laboratory, "Aviris." url<https://aviris.jpl.nasa.gov/>, 2017. Accedido 20-11-2017.
- [24] C. E. Cano Salazar, "Diseño de una arquitectura de un filtro digital de sobre muestreo de imágenes, en factor 2, de acuerdo al formato h. 264/svc sobre fpga," 2012.
- [25] C. Maxfield, *The design warrior's guide to FPGAs: devices, tools and flows*. Elsevier, 2004.
- [26] S. Brown, "Fundamentals of digital logic design with vhdl," 2010.
- [27] G. Swayze, R. N. Clark, F. Kruse, S. Sutley, and A. Gallagher, "Ground-truthing aviris mineral mapping at cuprite, nevada," 1992.
- [28] I. Xilinx, "Virtex-4 family overview," vol. 1, 2010.
- [29] I. Xilinx, "Virtex-5 family overview," vol. 1, 2015.
- [30] I. Xilinx, "Virtex-6 family overview," vol. 1, 2015.
- [31] I. Xilinx, "7 series fpgas data sheet: Overview," vol. 1, 2018.