Compresión de imágenes hiperespectrales con distorsión adaptable en lógica reconfigurale

Julián Caba[1], Dirk Stroobandt[2], María Díaz[3], Jesús Barba[1], Fernando Rincón[1], José Antonio de la Torre[1], Soledad Escolar[1], Sebastián López[3] y Juan Carlos López[1]

Resumen— **Las soluciones de compresión con pérdidas han crecido durante las últimas décadas debido al incremento de la tasa de datos de los sensores hiperespectrales de nueva generación, sin embargo las técnicas de compresión lineal incluyen información poco útil en regiones de poco interés para la aplicación final y al mismo tiempo escasa información en áreas de interés. En este trabajo, el compresor con pérdida HyperLCA ha sido extendido para incluir la característica de distorsión adaptativa en tiempo de ejecución, aportando varios ratios de compresión en un mismo escenario. La solución ha sido diseñada para mantener la posibilidad de desplegar la solución en dispositivos hardware reconfigurables (FPGAs). Los experimentos demuestran que la nueva versión del compresor es capaz de procesar 1024x1024 imágenes hiperespectrales y 180 bandas espectrales (377,5MB) en 0,935 segundos con un consumo de 1,145 vatios. Además, los resultados experimentales también revelan que nuestra arquitectura presenta un alto rendimiento (MSamples/s) y una notable eficiencia energética (MB/s por vatio), 10 y 6 veces superior a la mejor solución del estado del arte, respectivamente.**

Palabras clave— **FPGAs, hyperspectral imaging, anomaly detection, line-by-line performance, real-time, High-Level Synthesis, low-power.**

I. Introducción

La información recogida por sensores hiperespectrales proporciona una gran riqueza de información espectral, convirtiéndola en una de las principales candidatas para el análisis de áreas terrestres y, por ello, ha adquirido una importante relevancia, siendo ampliamente utilizada para una gran variedad de aplicaciones de teledetección como la agricultura de precisión, la cartografía geológica o la exploración minera. Sin embargo, la gran cantidad de datos recogidos por estos sensores requiere enormes recursos de almacenamiento a bordo o comunicaciones de gran ancho de banda, pero ambos son limitados. Además, los avances tecnológicos promueven la comercialización de sensores con mayores resoluciones espectrales y espaciales que hacen que el procesado de datos a bordo sea más desafiante [1], [2].

Tradicionalmente, la información es captada por sensores hiperespectrales montados sobre un vehículo aéreo no tripulado (UAV), donde el procesado de dicha información no suele realizarse debido principalmente por el bajo rendimiento y la limitada capacidad de energía de la plataforma que integra recursos de cómputo y de vuelo. Normalmente, los dispositivos que integran esta plataforma son dispositivos de bajo consumo y coste optimizado para procesar los datos captados, pero estos no cuentan con un alto rendimiento [3]. Existen varías alternativas que se han adoptado en los últimos años; 1) Las imágenes se descargan en la superficie terrestre para ser procesadas fuera de línea por sistemas de alto rendimiento; 2) En los UAVs las imágenes suelen almacenarse a bordo y procesarse cuando finaliza la misión de vuelo [4]. Recientemente, se han hecho algunos esfuerzos para transmitir las imágenes a tierra tan pronto como se capturan, pero se requiere una conexión punto a punto con un gran ancho de banda para reducir los grandes retrasos [5]. En cualquier caso, la transferencia de grandes volúmenes de datos es uno de los principales problemas que puede afectar al rendimiento global, así como el consumo energético consumido por la transmisión [6].

Dado que el ancho de banda es limitado, así como la memoria disponible en los dispositivos y los recursos de hardware dedicados, como los DSP, la forma de abordar este reto es utilizar técnicas de compresión de imágenes hiperespectrales a bordo. Aunque existe una gran variedad de algoritmos de compresión en la literatura, los algoritmos de compresión sin pérdidas son la opción preferente porque preservan la mayor parte de la información hiperespectral [2]. Las técnicas sin pérdidas producen datos no distorsionados después del proceso de descompresión, pero la relación de compresión no es lo suficientemente grande, sin embargo los métodos casi sin pérdidas permiten obtener valores más grandes de relación de compresión, introduciendo distorsiones que pueden ser controladas. Además, los sensores de última generación aumentan la velocidad de transmisión de datos, lo que requiere una mayor relación de compresión y/o reducir el tiempo de compresión para evitar la acumulación de datos sin comprimir y, por tanto, una transmisión eficiente [7]. Por lo tanto, los anchos de banda de comunicación limitados y los volúmenes de datos cada vez mayores obligan a pasar de las técnicas de compresión (casi) sin pérdidas a las técnicas de compresión con pérdidas, en las que se ha realizado un gran esfuerzo de investigación en los últimos años [2], [8].

En escenarios de sistemas móviles embebidos, los dispositivos de procesamiento paralelo, como las FPGAs, son adecuados para implementar algoritmos de compresión de imágenes hiperespectrales, debido al

[1]Dpto. Tecnologías y Sistemas de Información, UCLM, 13071 Ciudad Real, España. e-mail: {julian.caba, jesus.barba, fernando.rincon, joseantonio.torre, soledad.escolar, juancarlos.lopez}@uclm.es

[2]Ghent University, 9000 Ghent, Belgium. e-mail: dirk.stroobandt@ugent.be

[3]Instituto Universitario de Microelectrónica Aplicada (IUMA), 35017 Las Palmas de Gran Canaria, España. e-mail: {mdmartin, seblopez}@iuma.ulpgc.es

equilibrio entre el grado de paralelismo y el coste-energía, además del ahorro de costes que supone una solución basada en FPGAs con pocos recursos. Desafortunadamente, los limitados recursos computacionales de estos dispositivos suponen un nuevo reto cuando una solución se basa en este tipo de tecnologías, por lo que los esquemas de compresión de baja complejidad se erigen como la solución más práctica para escenarios tan restringidos [9], [10]. Sin embargo, la mayoría de los compresores con pérdida del estado del arte se basan en algoritmos existentes de compresión de imágenes 2D o vídeo, que se consideran de alta carga computacional, intensivos en requerimientos de memoria y de naturaleza no paralela [11]. Este hecho provoca que su uso sea limitado en entornos con recursos limitados, como es el caso de la compresión embarcada [12]. En este contexto, el Algoritmo de Compresión con Pérdidas para Sistemas de Imágenes Hiperespectrales (HyperLCA) [13] ha sido desarrollado como un compresor con pérdidas amigable con el hardware para imágenes hiperespectrales, que proporciona buenos ratios de compresión con una carga computacional razonable, ya que el proceso de compresión se basa en operaciones de transformación. Además, este algoritmo se ha diseñado para cumplir las limitaciones impuestas por los escáneres pushbroom/whiskbroom, considerando cada bloque de forma independiente, lo que se traduce en un menor uso de recursos de hardware [14]. Su idoneidad para aplicaciones de rendimiento en tiempo real ha sido analizada previamente en [15].

En este trabajo, se ha ampliado el algoritmo HyperLCA para incluir la función de distorsión adaptativa sin añadir una sobrecarga respecto a la implementación original, convirtiéndolo en un compresor inteligente al seleccionar los bloques relevantes cercanos a un patrón de firma predefinido. Por lo tanto, el algoritmo se ha adaptado para ejecutarse de forma más flexible con diferentes ratios de compresión sin modificar el conjunto de operaciones básicas realizadas en su versión original. Además, se ha llevado a cabo un análisis del HyperLCA basado en FPGA con función de distorsión adaptativa estableciendo diferentes reglas de la distorsión aplicada sobre el mismo escenario, lo que da lugar a una multiplicidad de relaciones de compresión dentro de la misma imagen y a una amplia gama de prestaciones de compresión de calidad. Esto significa que en la imagen comprimida, que contiene distorsiones de alta y baja tasa, se producen múltiples relaciones tasa-distorsión, de modo que no sólo se conservan perfectamente los píxeles hiperespectrales más diferentes; es decir, estos píxeles se encuentran dentro de la información extraída junto con píxeles extra relevantes de bloques interesantes, lo que se traduce en pérdidas insignificantes, ya que algunos bloques comprimidos tienen menos distorsión. Este hecho beneficia a muchas aplicaciones de imágenes hiperespectrales en las que la resolución espectral es decisiva; los bloques que contienen un gran número de similitudes pueden comprimirse con un criterio diferente al de los bloques con menores

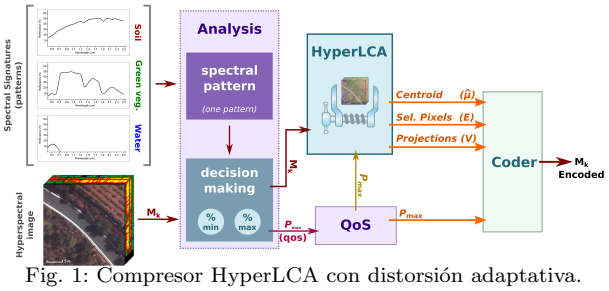

Fig. 1: Compresor HyperLCA con distorsión adaptativa.

similitudes con el patrón. Los bloques relevantes se seleccionan en tiempo de ejecución comparando píxel a píxel con un patrón de firma buscando similitudes en cada una de las bandas del espectro. Por último, la arquitectura se ha comparado con la versión anterior implementada en [16] y otros compresores del estado del arte en términos de rendimiento y utilización de recursos hardware. En este contexto, la mayor motivación de este trabajo es demostrar las afirmaciones anteriormente mencionadas sobre el HyperLCA con control de calidad, y por lo tanto, el objetivo es contribuir a la comunidad científica con un compresor inteligente con pérdida para imágenes hiperespectrales, que enriquezca aquellos bloques de mayor interés en un escenario inesperado/desconocido, mediante el uso de una solución de coste optimizado y energéticamente eficiente basada en tecnología FPGA.

II. Algoritmo HyperLCA

El algoritmo HyperLCA es un compresor con pérdidas basado en transformadas para imágenes hiperespectrales, cuya versión original se ha modificado para convertirla en una versión apta para hardware. Así, se ha rediseñado el algoritmo para conseguir altas relaciones tasa de compresión-distorsión, así como se ha disminuido la carga computacional debido al alto nivel de paralelismo implementado para aplicaciones basadas en sensores pushbroom/whiskbroom. Todos los cambios realizados permiten el uso de arquitecturas de procesamiento paralelo, como GPUs o FPGAs, sobre las que se ha implementado el algoritmo, [15] y [16], respectivamente. Además, el algoritmo HyperLCA ha sido especialmente diseñado para trabajar dentro de rangos numéricos específicos y utilizar aritmética entera, especificando la precisión necesaria para que las operaciones sean adecuadas en arquitecturas paralelas, tal y como se ha comentado previamente en [16], [14], [15].

La solución propuesta puede procesar independientemente bloques de la imagen con independencia de la alineación espacial de los mismos, lo que facilita la paralelización del proceso de compresión. Esto significa que los píxels hiperespectrales pueden procesarse sin dependencias. Sobre esta base, este trabajo incorpora la novedad de analizar cada píxel en tiempo de ejecución para determinar la tasa de distorsión que debe aplicarse al bloque que se está procesando, en lugar de establecer una tasa de compresión mínima deseada para todos los bloques capturados por el sensor, como hacen las versiones anteriores del algoritmo. La figura 1 muestra una visión

general de la nueva versión del algoritmo que representa las tres principales etapas de cálculo implicadas en el compresor HyperLCA con característica de distorsión adaptativa, compuestas por una una nueva etapa de *preprocesado o análisis de bloques*, que determinará el ratio de compresión a aplicar a cada bloque, una etapa de *transformación espectral*, cuyo core de operaciones es heredado de anteriores versiones y una etapa de *codificación*, que se ha adaptado de versiones anteriores para indicar el factor de distorsión aplicado a cada uno de los bloques.

A. Etapa 1: Preprocesamiento o análisis de bloques

En primer lugar, la imagen hiperespectral se divide en bloques compuestos por BS píxeles horizontales consecutivos ($\mathbf{M}_k$), también conocidos como líneas o bloques, que se procesarán. A continuación, se calculan los p_{max} píxeles más representativos dentro de un bloque ($\mathbf{M}_k$). Las implementaciones anteriores del algoritmo HyperLCA inicializan el parámetro p_{max} a partir de tres parámetros de entrada: CR (relación de compresión mínima deseada), N_{bits} (número de bits) y BS (tamaño del bloque) para determinar el número de transformaciones realizadas en el bloque hiperespectral. Sin embargo, en este trabajo, p_{max} no se fija en el momento del diseño, sino que se recalcula para cada bloque en función del número de píxeles similares con una firma de patrón hiperespectral. El equilibrio entre las métricas de calidad SNR, RMSE y MAD ha sido evaluado previamente en publicaciones anteriores combinando diferentes configuraciones de los parámetros de entrada (CR, N_{bits} y BS) [15]. En concreto, una de las mejores configuraciones es establecer los parámetros N_{bits} y BS en 12 y 1024, respectivamente. Estos valores se mantienen para calcular p_{max} en tiempo de ejecución, mientras que CR dependerá del número de píxeles hiperespectrales cercanos a la firma del patrón.

Por lo tanto, el compresor HyperLCA determina el número de transformaciones realizadas en el bloque que se está procesando, $\mathbf{M}_k$, como se muestra en la ecuación (1), donde DR se refiere al número de bits por píxel por banda; nb representa el número de bandas; BS es el número de píxeles horizontales consecutivos en un mismo bloque; CR se refiere a la relación de compresión deseada, definida como la relación entre el número de bits de la imagen original y los de los datos comprimidos; y N_{bits} es el número de bits que determina la precisión y el rango dinámico a utilizar para representar los valores de los datos comprimidos. Así, el número de transformaciones realizadas, p_{max}, determina directamente la máxima relación de compresión que se puede alcanzar con la configuración seleccionada, en la que valores más altos de p_{max} dan lugar a mejores imágenes reconstruidas, pero menores relaciones de compresión.

$$p_{\max} \leq \frac{DR \cdot nb \cdot (BS - CR)}{CR \cdot (DR \cdot nb + N_{\text{bits}} \cdot BS)} \qquad (1)$$

La relación de compresión (CR) utilizada en el cálculo de p_{max} para un bloque debe determinarse

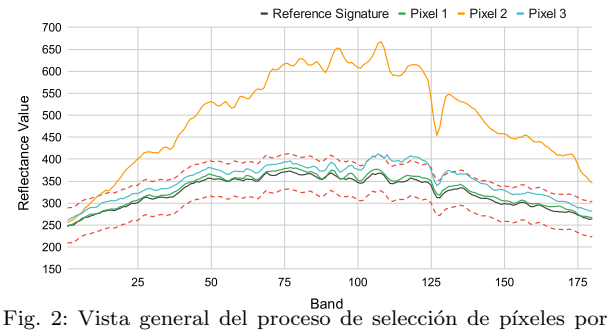

Fig. 2: Vista general del proceso de selección de píxeles por límites.

analizando cada uno de los píxeles de dicho bloque ($\mathbf{M}_k$), comparándolos uno a uno con una firma hiperespectral de referencia (patrón de referencia). Este proceso puede realizarse utilizando la distancia euclidiana con la firma de referencia y los píxeles hiperespectrales dentro de un bloque para determinar si el bloque, que está siendo procesado, contiene un alto porcentaje de píxeles cercanos a la referencia o, por el contrario, contiene pocas coincidencias. Sin embargo, el cálculo de la distancia euclídea es costoso en dispositivos reconfigurables y hace necesario buscar alternativas. En este sentido, este trabajo propone una solución amigable con el hardware mediante el uso de valores de band-limit, donde se define un valor *delta* para establecer los límites superior e inferior que dibujan la misma trama que la firma de referencia. Los resultados de este método son similares a los obtenidos por el método de la distancia euclídea.

La figura 2 muestra gráficamente el enfoque de límite de banda adoptado, donde se trazan tres píxeles hiperespectrales y una firma de referencia (línea negra). Además, las líneas rojas discontinuas definen los límites superior e inferior cuya tendencia es idéntica a la trazada por la firma de referencia. Por lo tanto, las firmas hiperespectrales que se encuentran dentro de ambos límites se consideran próximas a la referencia (línea verde de la figura 2), así como aquellos píxeles cuyo número de valores de reflectancia está fuera de los límites (línea azul de la figura 2); el número de valores fuera de los límites es configurable. Este hecho significa que las firmas hiperespectrales cuya reflectancia es muy similar a la de referencia, excepto en un pequeño número de bandas, también son consideradas próximas a la firma de referencia. Mientras tanto, los píxeles hiperespectrales cuyos valores de reflectancia más están fuera de los límites no se consideran (línea naranja de la figura 2).

Una vez que todos los píxeles hiperespectrales dentro de un bloque se definen como dentro o fuera de los límites, el CR se establece analizando el número de píxeles dentro de los límites. La figura 3 muestra tres ejemplos con diferentes ratios de compresión sobre el mismo escenario. En primer lugar, la figura 3a es una figura RGB original de un escenario de viñedos, mientras que la figura 3e muestra un patrón en el que los colores blanco y negro representan las firmas vegetales verdes y otros materiales, respectivamente. Desafortunadamente, el sensor utilizado en este trabajo captura la información hiperespectral

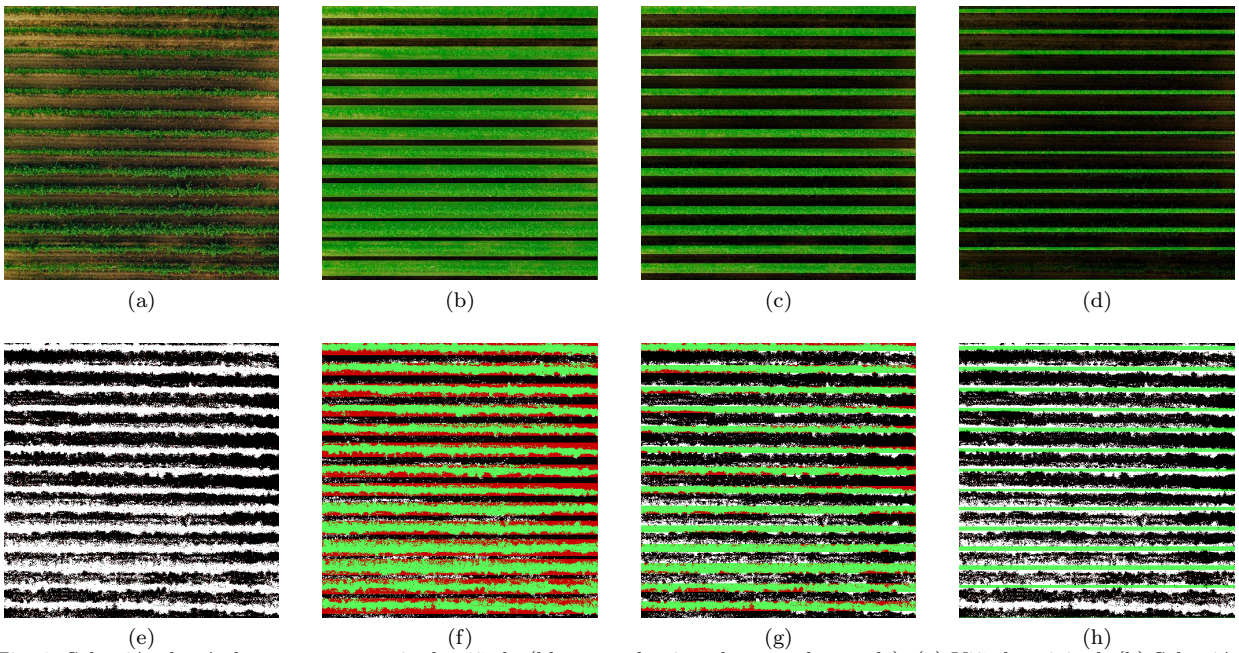

Fig. 3: Selección de píxeles en un escenario de viñedo (bloques seleccionados en color verde). (a) Viñedo original. (b) Selección de líneas con relación de compresión baja. (c) Selección de líneas con relación de compresión media. (d) Selección de líneas con una relación de compresión alta. (e) Patrón de selección de píxeles. (f) Diferencias entre los píxeles seleccionados en (b) y el patrón. (g) Diferencias entre los píxeles seleccionados en (c) y el patrón. (h) Diferencias entre los píxeles seleccionados en (d) y el patrón.

línea a línea, esto significa que no se obtiene inmediatamente toda la información hiperespectral de un escenario, por lo que las operaciones deben realizarse sobre una línea ($\mathbf{M}_k$) con BS píxeles hiperespectrales. Así, cuando se requiere una baja distorsión pero sólo en los bloques que contienen información útil para la aplicación final, el número de píxeles hiperespectrales próximos a la referencia debe ser pequeño. En este sentido, las figuras 3b, 3c y 3d resaltan en verde los bloques cuyo CR es bajo, es decir, estas líneas contienen información útil para la aplicación final, mientras que los bloques cuya información hiperespectral no es demasiado relevante se resaltan en negro. Por su parte, las figuras 3f, 3g y 3h destacan la pérdida, acierto y exceso de información hiperespectral respecto a la extracción de información ideal (figura 3e) en blanco, verde y rojo, respectivamente. El proceso de selección de bloques relevantes en los diferentes escenarios cumple los siguientes criterios: al menos 10 píxeles cercanos a la referencia para el escenario representado en la figura 3b, al menos 300 píxeles cercanos en el caso de la figura 3c y al menos 800 píxeles en el último caso (figura 3d).

B. Etapa 2: Transformación espectral

El compresor HyperLCA se encuentra entre los algoritmos basados en transformaciones que emplean una versión modificada del conocido método de ortogonalización Gram-Schmidt, ampliamente utilizado en compresión con pérdidas debido a su moderada complejidad en la que no se requiere reordenación de bandas. La idea básica de esta clase de algoritmos es mapear el dominio espacial de una imagen hiperespectral en su dominio de transformación [2]. En este sentido, la etapa *transformación espectral* selecciona

Algorithm 1 Transformación HyperLCA.

Inputs:
$\mathbf{M}_k = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{BS}]$, p_{max}
Outputs:
$\hat{\boldsymbol{\mu}}$; $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{p_{max}}]$; $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{p_{max}}]$
Algorithm:
Average pixel: $\hat{\boldsymbol{\mu}}$;
Centralized image: $\mathbf{C} = \mathbf{M}_k - \hat{\boldsymbol{\mu}} = [\mathbf{c}_1, \mathbf{c}_2, ...\mathbf{c}_{BS}]$;
for $n = 1$ **to** p_{max} **do**
  for $j = 1$ **to** BS **do**
    Brightness Calculation: $\mathbf{b}_j = \mathbf{c}'_j \cdot \mathbf{c}_j$;
  end for
  Maximum Brightness: $j_{max} = \text{argmax}(\mathbf{b}_j)$;
  Extracted pixels: $\mathbf{e}_n = \mathbf{r}_{j_{max}}$;
  $\mathbf{q}_n = \mathbf{c}_{j_{max}}$;
  $\mathbf{u}_n = \mathbf{q}_n / b_{j_{max}}$;
  Projection vector: $\mathbf{v}_n = \mathbf{u}'_n \cdot \mathbf{C}$;
  Information Subtraction: $\mathbf{C} = \mathbf{C} - \mathbf{q}_n \cdot \mathbf{v}_n$;
end for

los píxeles más diferentes de una imagen utilizando técnicas de proyección ortogonal. Así, los píxeles seleccionados se utilizan para proyectar la imagen con el fin de eliminar redundancias y obtener así una imagen espectral descorrelacionada y comprimida.

La *transformación espectral* se describe detalladamente en el algoritmo 1 y es la única etapa donde el core de operaciones se mantiene respecto a anteriores versiones. Las entradas de esta etapa son el bloque hiperespectral a comprimir ($\mathbf{M}_k$), que es el mismo que está siendo analizado por la etapa *Preproceso*, y el número de píxeles más diferentes a extraer ($p_{\max}$), que es la salida de la etapa *Preproceso*, es decir, se en-

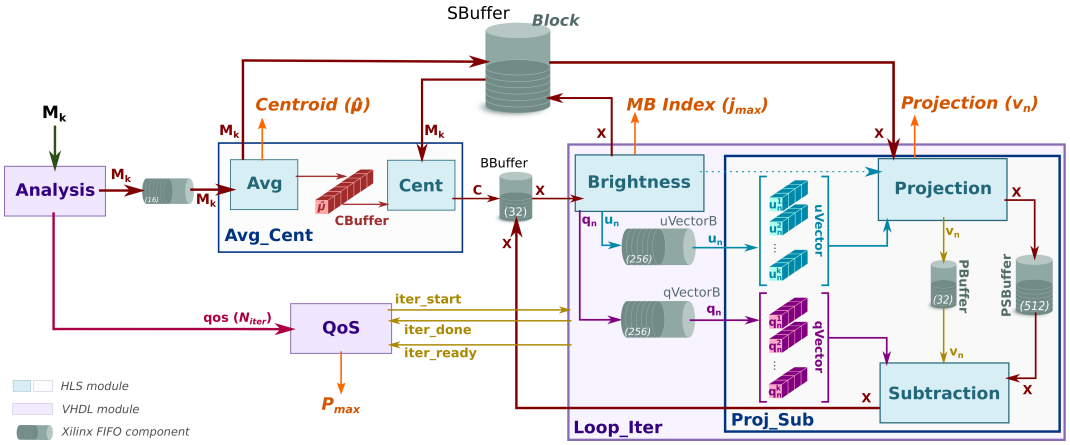

Fig. 4: Implementación hardware del algoritmo de compresión HyperLCA con distorsión adaptativa.

carga de determinar el número de transformaciones o proyecciones ortogonales a realizar sobre el bloque hiperespectral. Mientras que las salidas de esta etapa son el píxel medio ($\hat{\boldsymbol{\mu}}$) del bloque hiperespectral de entrada ($\mathbf{M}_k$), el conjunto de índices relacionados con los p_{rmmax} píxeles más diferentes ($\mathbf{E}$) y sus vectores de proyección ($\mathbf{V}$).

C. Etapa 3: Codificación

La última etapa del compresor HyperLCA realiza la entropía-codificación de los vectores recibidos de la etapa *Transformación Espectral*, donde el uso del algoritmo de Golomb-Rice [17] permite considerar cada vector independientemente del resto, es decir, las salidas de esta etapa pueden ser consumidas a medida que se reciben, por lo que ambas etapas se realizan en paralelo. Para ello, se calcula el parámetro de compresión (N) como valor medio del vector que se está procesando. Posteriormente, se dividen los elementos del vector por N y se obtienen el cociente (q) y el resto (r) de dicha operación. En segundo lugar, se calcula la menor potencia de 2 mayor que N como $b = log_2(N) + 1$. El cociente (q) se codifica utilizando código unario, mientras que el resto (r) se codifica como binario plano utilizando $b-1$ bits para valores de r inferiores a $2^b - N$, de lo contrario se codifica como $r + 2^b - N$ utilizando b bits.

D. Etapa 4: Generación del bitstream

Por último, se realiza la etapa *empaquetado de datos* para generar un único flujo de bits que contiene las salidas de las etapas de compresión anteriormente mencionadas. La figura 5 muestra gráficamente la estructura generada para el flujo de bits comprimido, que se divide en dos bloques jerárquicos.

En primer lugar, tiene lugar una cabecera global al principio de la cadena donde se recoge la información relativa a la imagen hiperespectral, incluidos los parámetros utilizados en el proceso de compresión; información espacial (número de columnas y filas) y espectral (número de bandas) de la imagen procesada, el tamaño de bloque utilizado (BS), el número de bits por píxel por banda (DR) y el número de bits utilizados para representar los valores de los datos comprimidos (N_{bits}).

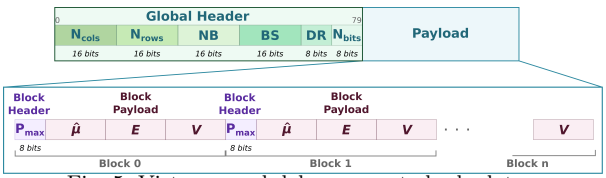

Fig. 5: Vista general del empaquetado de datos.

La carga útil se coloca después de esa cabecera global, que se divide en bloques que contienen dos campos: *cabecera de bloque* y *carga útil*. Debido a que los bloques de una imagen hiperespectral se comprimen con un número variado de transformaciones, es obligatoria una cabecera de 8 bits para determinar el p_{max} aplicado en cada bloque. Tras la *cabecera de bloque*, se encuentra la *carga útil* del mismo, que está compuesto por un único centroide independientemente de la distorsión aplicada sobre el bloque y a continuación los vectores $\mathbf{E}$ (píxeles más diferentes) y $\mathbf{V}$ (proyecciones), cuyo número viene denotado por la mencionada cabecera de 8 bits.

III. Implementación hardware (FPGA)

En este trabajo se han reutilizado aceleradores hardware especializados de [16] para construir la nueva arquitectura de la transformación HyperLCA, añadiendo el comportamiento dinámico necesario para incluir múltiples ratios de compresión. La figura 4 muestra una visión general de la arquitectura hardware implementada para las etapas de *preprocesamiento* y *transformación espectral*, donde los recuadros azules corresponden a los módulos descritos en HLS, y por tanto heredados del desarrollo anterior. Estos módulos están conectados a través de buffers de memoria y lógica a medida que permite orquestarlos para obtener el comportamiento deseado. Para mayor claridad, la correspondencia de las cajas azules con las operaciones realizadas en el algoritmo 1 es la siguiente: *Avg_Cent* corresponde a las líneas 1 y 2, mientras que *loop_iter* es el cuerpo del bucle principal que comprende desde la línea 3 a la 13, donde *brightness* se calcula en el bucle interior y en la línea 7 del Algoritmo 1, obteniéndose además vectores ortogonales ($\mathbf{q}$ y $\mathbf{u}$) una vez seleccionado el píxel más brillante. Mientras tanto, *proyection* y

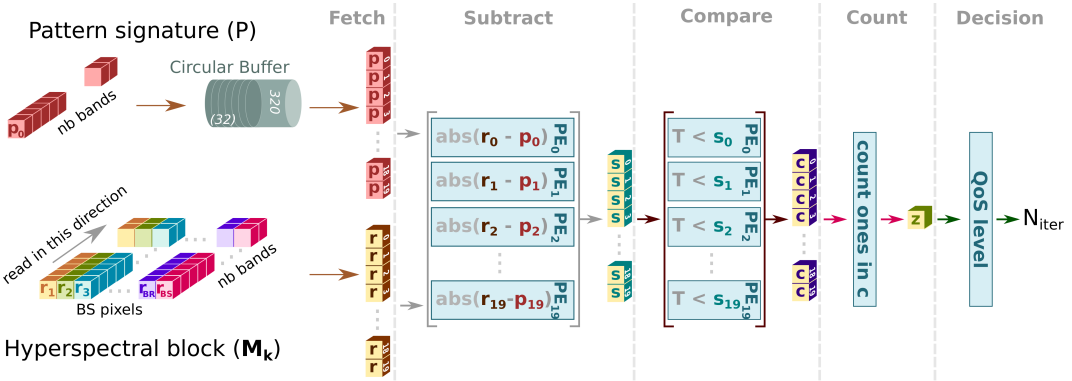

Fig. 6: Arquitectura basada en pipelines para calcular el número de transformaciones (PEs = 20).

subtraction coinciden con las líneas 11 y 12 del algoritmo 1, respectivamente. En busca de mejorar el rendimiento, la arquitectura introduce una mejora a través de la partición de los vectores ortogonales, **q** y **u**, utilizando VHDL en lugar de aplicar una directiva en la descripción HLS.

Hasta aquí, la arquitectura descrita se encarga de realizar el proceso de transformación sobre los bloques hiperespectrales ($\mathbf{M}_k$). Sin embargo, las operaciones de transformación realizadas, es decir, el número de ejecuciones del módulo *loop_iter*, se calcula ahora en tiempo de ejecución. El número de iteraciones está relacionado con la relación de compresión aplicada a un bloque, lo que también significa la cantidad de datos hiperespectrales que se representan para cada bloque. Así, el módulo *QoS* gestiona este proceso, además de notificar el valor p_{max} utilizado para codificar el bloque hiperespectral actual ($\mathbf{M}_k$). Por su parte, el módulo *Analysis* se encarga de calcular el número de iteraciones a realizar en función de la similitud entre los píxeles que componen un bloque hiperespectral y la firma del patrón.

Subtract. Una vez que los datos hiperespectrales están listos, la etapa *subtract* sustrae de forma absoluta del bloque hiperespectral actual ($\mathbf{M}_k$) la firma del patrón por bandas en paralelo, de forma que a la primera banda del píxel hiperespectral perteneciente al bloque procesado se le resta la primera banda del patrón almacenado, luego la segunda banda de ambos y así sucesivamente. Los resultados de esta etapa se almacenan en los registros s_i, que se representan como cuadros amarillos y verdes en la figura 6.

Compare. Los resultados obtenidos de cada PE en la etapa *subtract* se comparan con un valor constante, T, que es el valor *delta* predefinido. Así, si la constante T es menor que el resultado de la resta absoluta obtenida en la etapa anterior, significa que la banda correspondiente está dentro de los límites. Por el contrario, si la comparación da como resultado que T es mayor, implica que el valor de la banda está fuera de los límites. La salida de esta etapa es un cero o un uno cuando la banda está fuera o dentro de los límites, respectivamente. De forma similar a la etapa anterior, los resultados de las comparaciones también se almacenan en los registros c_i (cuadros amarillos y morados en la figura 6).

Count. En esta etapa, los registros que contienen un uno, es decir, la banda está dentro de los límites, se cuentan y se suman a una suma parcial, que se almacena en el registro z (ver figura 6).

Por último, la etapa *decision* se realiza después de comparar todo el píxel hiperespectral de un bloque con la firma del patrón. Determina si un píxel está próximo al patrón a partir del valor de la suma almacenado en el registro z y calculado por la etapa *count*. En este sentido, no es necesario que todos los valores de banda estén dentro de los límites, la solución permite que varias bandas estén fuera de ellos (ver línea azul en la figura 2). Además, la etapa *decision* establece el número de operaciones de transformación realizadas por la etapa *transformación espectral*. Se calcula contando el número de píxeles próximos a la firma del patrón (T_c) y, a continuación, se compara dicha suma con las reglas predefinidas por el usuario, que determina el grado de distorsión. Para ello, el usuario debe proporcionar dos constantes R_{min} y R_{max} para aplicar los siguientes casos, donde QoS_{max}, QoS_{min} y QoS_{med} son el número de iteraciones que también están predefinidas; un número alto significa mayor información espectral extraída.

$$\begin{cases} N_{iter} = QoS_{max}, & \text{if } T_c \geq R_{max} \\ N_{iter} = QoS_{min}, & \text{if } T_c \leq R_{min} \\ N_{iter} = QoS_{med}, & \text{otherwise} \end{cases}$$

IV. Resultados experimentales

En esta sección se evalúa el compresor con distorsión adaptativa mediante el análisis de la arquitectura basada en FPGA implementada en una ZC7Z020-CLG484 y el rendimiento alcanzado por la misma, comparando el rendimiento y el consumo de energía cuando varios PE trabajan en paralelo. En esta sección también se evalúa la precisión de la selección de píxeles cercanos mediante límites frente al uso de la distancia euclídea.

A. Dataset

El rendimiento de la arquitectura presentada ha sido evaluado mediante un conjunto de imágenes hiperespectrales, que fueron captadas por una plataforma aérea personalizada sobre diferentes zonas de cultivo en la isla de Gran Canaria (España). El conjunto de datos contiene 4 imágenes hiperespectrales recogidas sobre dos zonas de viñedos diferentes, cuyas

Tabla I: Comparación de los resultados obtenidos entre la distancia euclidiana y el método propuesto.

Limites	Errores	Verdaderos Positivos ($\mathbf{E} \cap \mathbf{L}$)	Falsos Negativos ($\mathbf{E} - \mathbf{L}$)	Falsos Positivos ($\mathbf{L} - \mathbf{E}$)	Verdaderos Negativos ($\mathbf{U} - (\mathbf{E} \cup \mathbf{L})$)	$\mathbf{E} \subseteq \mathbf{L}$	$\mathbf{L} \subseteq \mathbf{E}$
±15	20	49.093 %	50.906 %	0.0000 %	100.00 %	0.6835 %	100.00 %
	25	54.467 %	45.532 %	0.0000 %	100.00 %	0.7812 %	100.00 %
	30	60.003 %	39.996 %	0.0077 %	99.999 %	1.0745 %	99.804 %
±20	20	82.158 %	17.841 %	0.2145 %	99.992 %	6.3476 %	93.261 %
	25	89.024 %	10.975 %	0.9618 %	99.963 %	14.160 %	73.339 %
	30	94.686 %	5.3133 %	2.8799 %	99.879 %	28.711 %	41.601 %
±23	20	96.843 %	3.1564 %	7.6101 %	99.658 %	42.187 %	22.949 %
	25	99.256 %	0.7437 %	12.146 %	99.413 %	76.172 %	8.0078 %
	30	99.958 %	0.0418 %	17.789 %	99.074 %	98.339 %	2.6369 %
±24	20	98.703 %	1.2969 %	12.198 %	99.413 %	65.332 %	11.230 %
	25	99.823 %	0.1766 %	17.296 %	99.106 %	93.652 %	3.0273 %
	30	99.995 %	0.0046 %	22.758 %	98.739 %	99.804 %	1.3671 %
±25	20	99.526 %	0.4741 %	16.796 %	99.140 %	85.156 %	4.0039 %
	25	99.944 %	0.0557 %	22.292 %	98.773 %	97.949 %	1.5625 %
	30	100.00 %	0.0000 %	27.498 %	98.377 %	100.00 %	0.7812 %
±26	20	99.842 %	0.1580 %	21.478 %	98.831 %	94.443 %	2.2461 %
	25	99.991 %	0.0093 %	26.839 %	98.430 %	99.609 %	0.8789 %
	30	100.00 %	0.0000 %	31.697 %	98.014 %	100.00 %	0.6836 %
±27	20	99.939 %	0.0604 %	25.837 %	98.510 %	97.754 %	1.0742 %
	25	100.00 %	0.0000 %	30.963 %	98.081 %	100.00 %	0.6836 %
	30	100.00 %	0.0000 %	35.542 %	97.641 %	100.00 %	0.6836 %
±30	20	100.00 %	0.0000 %	37.044 %	97.482 %	100.00 %	0.4883 %
	25	100.00 %	0.0000 %	41.147 %	97.008 %	100.00 %	0.3906 %
	30	100.00 %	0.0000 %	44.861 %	96.519 %	100.00 %	0.2929 %

coordenadas exactas son 27°59′35,6″N 15°36′25,6″W y 27°59′15,2″N 15°35′51,9″W.

La plataforma de adquisición se compone de una cámara hiperespectral *Specim FX10* pushbroom en un dron DJI Matrice 600 [18]. El sensor de imagen captura 1024 píxeles espaciales por pista y hasta 224 bandas espectrales en el rango entre 400 y 1000 nm. Sin embargo, en los experimentos realizados sólo se trabaja con 180 bandas espectrales; se descartan las 10 primeras bandas espectrales así como las 34 últimas, debido a que la respuesta en los límites del espectro electromagnético es bajo.

B. Precisión del método de selección de píxeles

Se ha evaluado el método de selección de píxeles mediante límites superior e inferior propuesto en este trabajo, comparando los resultados con los obtenidos mediante la distancia euclídea. Para ello, el análisis se ha realizado utilizando las operaciones de la Teoría de Conjuntos, aplicando seis métricas diferentes para analizar el conjunto de píxeles seleccionados. Cabe destacar que el resultado obtenido, independientemente del método utilizado, es un conjunto de píxeles similares al patrón de referencia, por lo que el análisis debe guiarse por las similitudes entre conjuntos de resultados.

En este contexto, el conjunto universal, $\mathbf{U}$, está compuesto por el índice de píxeles dentro de un bloque hiperespectral ($\mathbf{M}_k$). Así, $\mathbf{U}$ se define como $\mathbf{U} = \{0, ..., 1023\}$. El conjunto de píxeles seleccionados por el algoritmo de distancia euclidiana se denota como $\mathbf{E}$ ($\mathbf{E} \subseteq \mathbf{U}$), mientras que los extraídos por el método de límites se representa como $\mathbf{L}$ ($\mathbf{L} \subseteq \mathbf{U}$). En las líneas siguientes se describen las seis métricas utilizadas para determinar el grado de similitud de los dos enfoques.

Verdaderos Positivos. Esta métrica analiza el número de píxeles comunes seleccionados por los dos algoritmos, es decir, la operación de intersección se aplica a los conjuntos $\mathbf{E}$ y $\mathbf{L}$, que se obtienen mediante la distancia euclídea y el método del límite, respectivamente. Un porcentaje alto significa que la mayoría de los píxeles seleccionados por la distancia euclídea también son elegidos por el método propuesto, pero no implica que sean muy similares; el conjunto $\mathbf{L}$ puede contener más píxeles que $\mathbf{E}$.

Falsos Negativos. En este caso, representa los píxeles dentro de $\mathbf{E}$ pero no en $\mathbf{L}$, o en otras palabras, denota los píxeles que el método propuesto debería haber incluido en su conjunto de soluciones pero que en realidad no están incluidos. Esta métrica y la métrica de verdaderos positivos proporcionan el grado de similitud entre conjuntos; un bajo porcentaje de falsos negativos y un alto porcentaje de verdaderos positivos significa que dos conjuntos son muy similares.

Falsos Positivos. Esta métrica mide el grado de píxeles extra considerados por el método del límite. También se calcula mediante la operación diferencia, pero ahora los píxeles comunes de $\mathbf{E}$ y $\mathbf{L}$ se extraen de $\mathbf{L}$ en lugar de $\mathbf{E}$. Un valor porcentual pequeño de esta métrica implica que la solución es ajustada, siempre que la métrica de verdaderos positivos tenga un valor alto y la métrica de falsos negativos tenga un valor bajo.

Verdaderos Negativos. En este caso, la métrica denota los píxeles no incluidos en las soluciones de ambos métodos, es decir, los píxeles que no están en $\mathbf{E}$ ni en $\mathbf{L}$. Se calcula mediante la unión de los conjuntos $\mathbf{E}$ y $\mathbf{L}$ y luego el resultado se resta (operación de diferencia) del conjunto universal ($\mathbf{U}$).

E es un subconjunto de L. Esta métrica determina si el conjunto obtenido a partir de la distancia euclídea es un subconjunto del conjunto obtenido por el método propuesto. Indica que la solución contiene todos los píxeles que el algoritmo de referencia.

L es un subconjunto de E. En este caso, denota la falta de píxeles en la solución porque el conjunto obtenido por el algoritmo de la distancia euclídea es mayor que el obtenido por el método del límite.

En la tabla I se indica el porcentaje de similitud entre la distancia euclídea y el método pruopuesto. Para ello, el conjunto obtenido por la distancia

Tabla II: Recursos hardware del algoritmo HyperLCA con función de distorsión adaptativa (Dispositivo: Xilinx ZynQ-7020).

PEs	BRAM18K	DSP48E	FlipFlops	LUTs
1	211 *(75.3 %)*	9 *(4.0 %)*	6,146 *(5.7 %)*	7,502 *(14.1 %)*
2	208 *(74.2 %)*	16 *(7.2 %)*	6,186 *(5.8 %)*	8,329 *(15.6 %)*
4	216 *(77.1 %)*	30 *(13.6 %)*	7,048 *(6.6 %)*	9,384 *(17.6 %)*
6	223 *(79.6 %)*	62 *(28.1 %)*	8,134 *(7.6 %)*	10,886 *(20.4 %)*
10	232 *(82.8 %)*	102 *(46.3 %)*	9,684 *(9.1 %)*	12,733 *(23.9 %)*
12	217 *(77.5 %)*	122 *(55.4 %)*	10,573 *(9.9 %)*	14,458 *(27.1 %)*
20	218 *(77.8 %)*	202 *(91.8 %)*	13,834 *(13.0 %)*	19,178 *(36.0 %)*

euclídea debe cumplir que ninguno de los miembros de dicho conjunto supere el valor de 200 con respecto a la distancia a la referencia. Por su parte, el método de límites define los límites superior e inferior en función de los valores espectrales de la firma patrón con una única variable (columna *limites*), también define el número máximo de bandas espectrales cuyos valores quedan fuera de los límites (columna *errores*). Estos parámetros son configurables y, para el análisis actual, los valores seleccionados se enumeran en las dos primeras columnas de la tabla I. Las mejores configuraciones son aquellas con un alto porcentaje de verdaderos positivos y un bajo porcentaje de falsos positivos y negativos, por lo que la mejor configuración es aquella cuyos parámetros *Límites* y *Errores* están configurados con ±25 y 30, respectivamente.

C. Análisis hardware

La arquitectura propuesta se ha implementado en un dispositivo de coste optimizado (FPGA ZC7Z020-CLG484) en comparación con otras arquitecturas del mismo fabricante, como Kintex o UltraScale/UltraScale+. En este sentido, el dispositivo seleccionado ofrece un buen equilibrio en tres aspectos: coste, consumo energético y rendimiento. Sin embargo, para lograr un buen equilibrio de ratio en rendimiento por vatio, es necesario invertir esfuerzos de ingeniería en la parte arquitectónica como consecuencia de los limitados recursos del SoC ZynQ.

La tabla II lista los recursos lógicos programables de acuerdo con el número de elementos de procesamiento (PE) instanciados en cada configuración tras la fase de post-implementación. El tamaño del bloque hiperespectral (BS) se fija en 1024 píxeles hiperespectrales, mientras que el tamaño espacial es de 180 bandas. El número de PEs que se pueden instanciar es de 20 PEs, ya que el número de DSPs (Digital Signal Processing units) disponibles es de 220 en el dispositivo FPGA ZC7Z020-CLG484 y dicha configuración requiere 202 de estas unidades. Así, los $DSPs$ son el recurso limitante de la arquitectura propuesta, ya que la siguiente configuración posible instanciará 30 PEs, por lo que demandará 257 $DSPs$, es decir, se solicita un dispositivo con más recursos hardware. Cabe mencionar que el número de PEs debe ser divisor del número de bandas para aplicar correctamente las optimizaciones. Mientras tanto, el resto de recursos hardware no son críticos para la escalabilidad de la arquitectura, aunque las BRAMs están entre 74 % y 83 %, realmente depende del parámetro de tamaño de bloque (BS), cuyo valor se establece en tiempo de diseño.

Tabla III: MSamples/s alcanzadas por las versiones del acelerador HyperLCA con función de distorsión adaptativa.

PEs	1	2	4	6	10	12	20
100 MHz	58	120	236	332	494	562	777
143 MHz	87	180	352	495	734	835	1149

El rendimiento de cada configuración del acelerador hardware se representa en MSamples por segundo (MSamples/s), que depende del número de PEs instanciados y de la configuración de la frecuencia de reloj. En este sentido, los modelos RTL para los módulos HLS se generaron fijando la frecuencia de reloj objetivo en 100MHz. A continuación, se sintetizaron dos versiones del flujo de bits para dos configuraciones de reloj diferentes: 100 MHz y 143 MHz. En la tabla III se muestra el promedio de MSamples/s alcanzado por el acelerador hardware en función del número de PEs y la configuración de la frecuencia de reloj. Cabe mencionar que la métrica MSamples/s depende principalmente de dos parámetros de configuración del acelerador hardware: la firma del patrón y las reglas que determinan el ratio de compresión aplicado al bloque hiperespectral actual. Así, los valores de MSamples/s de la tabla III son una media de los resultados obtenidos de las cuatro imágenes sensadas por la plataforma UAV aplicando tres reglas, donde las versiones rápidas (143MHz) consiguen un 33 % mejor rendimiento que las lentas (100MHz).

Aunque las herramientas de análisis de energía de Vivado proporcionan resultados precisos sobre el consumo de energía después de la fase de post-implementación, hay otros componentes de hardware fuera del dispositivo FPGA que están involucrados en el proceso de compresión hiperespectral, como la memoria externa (RAM), que no se tienen en cuenta en los informes de energía. Desafortunadamente, la plataforma ZedBoard no tiene múltiples fuentes de alimentación para obtener información en tiempo real sobre el consumo de energía, por lo que no se puede realizar una medición de grano fino como hace la herramienta de estimación de energía AMD-Xilinx. Por otro lado, la herramienta de gestión de hardware Vivado supervisa la tensión de alimentación de los componentes del sistema de procesamiento (PS) y la lógica programable (PL), pero la frecuencia de actualización es demasiado baja. Por lo tanto, el consumo de energía de la arquitectura presentada en este documento se ha medido en la plataforma ZedBoard a través del puerto de detección de corriente (conector J21), utilizando un multímetro con resolución suficiente para mostrar la diferencia en el consumo de energía cuando se realiza un proceso de compresión [19]. La tabla IV enumera el consumo de potencia en vatios y la eficiencia energética en MMuestras/s por vatio (MSs/W) en las diferentes versiones del HyperLCA con función de distorsión.

La figura 7 muestra gráficamente una comparación del rendimiento y la compensación del consumo de energía entre las versiones de 100MHz (barras azules y línea morada) y 143MHz (barras rojas y línea na-

Tabla IV: Consumo de energía del diseño en ZedBoard medido mediante el conector J21.

	PEs	1	2	4	6	10	12	20
100MHz	Vatios (W)	3.624	3.612	3.636	3.642	3.654	3.660	3.720
	MSs/W	16	33	64	91	135	153	208
143MHz	Vatios (W)	3.660	3.648	3.660	3.696	3.714	3.720	3.768
	MSs/W	23	49	96	133	197	224	304

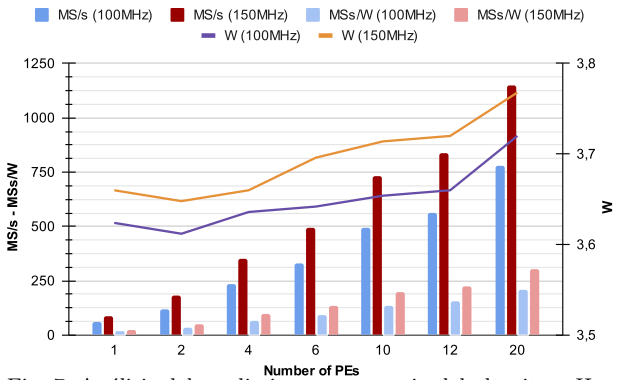

Fig. 7: Análisis del rendimiento y potencia del algoritmo HyperLCA con función de distorsión adaptativa en sus versiones de 100 MHz y 143 MHz a través del conector J21.

ranja) del algoritmo HyperLCA con función de distorsión adaptativa. El consumo de energía tiene un comportamiento similar en ambas versiones, cuyas variaciones dependen del número de PEs, mientras que la eficiencia energética en la versión de 143MHz es un 30 % superior a la versión de 100MHz.

D. Comparativa con otros compresores

La tabla V muestra un resumen detallado del rendimiento alcanzado por las arquitecturas del estado del arte y la presentada en este trabajo, utilizando los valores obtenidos tras la fase de post-implementación para analizar las arquitecturas en el mismo punto. Además, la tabla también muestra en la penúltima columna la potencia en chip representada en vatios. Lamentablemente, algunas propuestas del estado del arte no proporcionan dicha información, por lo que se ha estimado con la herramienta Xilinx Power Estimator (XPE). El uso de recursos embebidos especializados de la FPGA no tiene un gran impacto en el consumo de potencia, pero depende del número de regiones de reloj que estén activas, la cantidad de recursos y la tecnología FPGA utilizada. En este sentido, la arquitectura presentada por D. Keymulen en [22] requiere más potencia que la presentada por D. Báscones et al. en [25], donde ambas propuestas utilizan el mismo dispositivo FPGA y VHDL para describir la arquitectura; la diferencia radica en el número de recursos FPGA utilizados para cada propuesta. Podemos concluir que la arquitectura presentada en este trabajo supera al resto de propuestas del estado del arte en el balance MB/s por vatio (MBS/W), donde la versión más rápida del compresor HyperLCA con función de distorsión, es decir, la versión que contiene 20 PEs trabajando en paralelo, es entre 4,9× y 5,8× mejor que la arquitectura presentada por D. Báscones et al. en [25], cuando la frecuencia de reloj se configura a 100MHz y 143MHz, respectivamente.

Cabe mencionar que la implementación anterior del compresor HyperLCA es entre 3,6× y 4,1× más lenta que la versión con función de distorsión adaptativa. Además, la configuración de la frecuencia de reloj tiene poca influencia en la tasa de MB/s por vatio, se incrementa 54,66 MBS/W cuando la frecuencia de reloj se establece en 143MHz. Por tanto, la arquitectura presentada en este trabajo es capaz de comprimir la citada imagen hiperespectral de 1024x1024 de tamaño espacial y 180 bandas espectrales en 0,935s con un consumo de 1,145W.

V. Conclusiones

En este trabajo se ha abordado la inclusión de la característica de distorsión adaptativa en el algoritmo HyperLCA con el fin de aumentar la información espectral en aquellas regiones de interés, cuya información espectral contenga una cantidad de píxeles cercana a una firma de patrón hiperespectral predefinida. De este modo, la propuesta puede adaptarse al escenario que se esté procesando, por ejemplo, podría recogerse más información espectral de la vegetación que del suelo en un viñedo. Además, el compresor puede configurarse para aumentar la información en las líneas que contienen píxeles anómalos, por ejemplo, barcos en medio del mar.

El conjunto de operaciones hiperespectrales centrales se hereda de trabajos anteriores [16], [14] donde se comprobó la idoneidad de la tecnología FPGA para este tipo de aplicaciones, por lo que se ahorra una cantidad significativa de tiempo. Sin embargo, la nueva arquitectura incluye nuevas optimizaciones que permiten instanciar más PEs trabajando en paralelo, que a su ver permite aumentar el rendimiento, mientras que la función de distorsión adaptativa introduce una pequeña sobrecarga en términos de tamaño debido a la necesidad de incluir una cabecera de 8 bits por cada muestra que se comprime. Además, la comparación con otras arquitecturas basadas en FPGA revela que la arquitectura propuesta es una solución eficiente en coste-energía sin reducir la calidad de compresión, cuya pérdida de información espectral puede reducirse incrementando el parámetro p_{max}, con lo que se consigue una mejor calidad de compresión.

Agradecimientos

Esta investigación está parcialmente financiada por el Ministerio de Economía y Competitividad (MINECO) del Gobierno de España a través de los proyectos TALENT (PID2020-116417RB-C4, subproyectos 1 y 4) y MIRATAR (TED2021-132149B-C41) y por el programa Europeo Horizonte 2020 bajo el proyecto SHAPES (GA Nº 857159).

Referencias

[1] Antonio Plaza, Jon Atli Benediktsson, Joseph W Boardman, Jason Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Mathieu Fauvel, Paolo Gamba,

Tabla V: Comparación de rendimiento con otras implementaciones de compresores hiperespectrales basadas en FPGA.

Propuesta	Imagen Hiperespectral	Muestras	Lineas	Bandas	Rendimiento (MSamples/s)	MB/s	Vatios (W)	MBS/W
L. Santos et al.[20]	-	16	16	256	30.25	0.236	2.029*	0.116
A. García et al.[21]	AVIRIS (Jasper Ridge)	614	512	224	27.7	5.449	2.022*	2.695
D. Keymulen (1 core)[22]	AVIRIS (Jasper Ridge)	614	512	224	9	4.722	7.84	0.602
D. Keymulen (15 cores)[22]	AVIRIS (Jasper Ridge)	614	512	224	95	49.842	11.4	4.372
D. Fernández et al.[23]	AVIRIS (Jasper Ridge)	614	512	224	80.29	9.004	2.46*	3.661
D. Báscones et al.[24]	AVIRIS	512	512	256	119.96	29.99	2.732	10.977
D. Báscones et al.[25]	AVIRIS	512	512	256	162.3	40.575	0.714	56.828
Y. Barrios et al. (1 core) [26]	AVIRIS	512	512	256	0.4	0.1	0.6*	0.167
Y. Barrios et al. (8 cores)[26]	AVIRIS	512	512	256	1.7	0.425	0.9*	0.472
J. Caba et al. (12 PEs) [16]	Own *(Specim FX10)*	1024	1024	180	342	120.234	1.6	75.146
J. Caba et al. (12 PEs) [16]	Own *(Specim FX10)*	1024	1024	180	511	179.648	2.22	80.923
Our (20 PEs - 100MHz)	Own *(Specim FX10)*	1024	1024	180	777	273.164	0.993	275.089
Our (20 PEs - 143MHz)	Own *(Specim FX10)*	1024	1024	180	1,149	403.945	1.225	329.751

*Obtenido mediante la herramienta Xilinx Power Estimator (XPE).

Anthony Gualtieri, et al., "Recent advances in techniques for hyperspectral image processing," *Remote sensing of environment*, vol. 113, pp. S110–S122, 2009.

[2] Amal Altamimi and Belgacem Ben Youssef, "A systematic review of hardware-accelerated compression of remotely sensed hyperspectral images," *Sensors*, vol. 22, no. 1, 2022.

[3] Miloš Radosavljević, Branko Brkljač, Predrag Lugonja, Vladimir Crnojević, Željen Trpovski, Zixiang Xiong, and Dejan Vukobratović, "Lossy compression of multispectral satellite images with application to crop thematic mapping: A hevc comparative study," *Remote Sensing*, vol. 12, no. 10, pp. 1590, 2020.

[4] Fred Ortenberg, PS Thenkabail, JG Lyon, and A Huete, "Hyperspectral sensor characteristics: airborne, spaceborne, hand-held, and truck-mounted; integration of hyperspectral data with lidar," *Hyperspectral Remote sensing of vegetation*, pp. 39–68, 2011.

[5] José M. Melián, Adán Jiménez, María Díaz, Alejandro Morales, Pablo Horstrand, Raúl Guerra, Sebastián López, and José F. López, "Real-Time Hyperspectral Data Transmission for UAV-Based Acquisition Platforms," *Remote Sensing*, vol. 13, no. 5, 2021.

[6] Elodie Morin, Mickael Maman, Roberto Guizzetti, and Andrzej Duda, "Comparison of the Device Lifetime in Wireless Networks for the Internet of Things," *IEEE Access*, vol. 5, pp. 7097–7114, 2017.

[7] Bormin Huang, *Satellite data compression*, Springer Science & Business Media, 2011.

[8] Yaman Dua, Vinod Kumar, and Ravi Shankar Singh, "Comprehensive review of hyperspectral image compression algorithms," *Optical Engineering*, vol. 59, no. 9, pp. 1 – 39, 2020.

[9] Aaron B Kiely, Matthew Klimesh, Ian Blanes, Jonathan Ligo, Enrico Magli, Nazeeh Aranki, Michael Burl, Roberto Camarero, Michael Cheng, Sam Dolinar, et al., "The new ccsds standard for low-complexity lossless and near-lossless multispectral and hyperspectral image compression," 2018.

[10] Estanislau Augé, Jose Enrique Sánchez, Aaron B Kiely, Ian Blanes, and Joan Serra-Sagrista, "Performance impact of parameter tuning on the CCSDS-123 lossless multi-and hyperspectral image compression standard," *Journal of Applied Remote Sensing*, vol. 7, no. 1, pp. 074594, 2013.

[11] Lucana Santos, Enrico Magli, Raffaele Vitulli, José F López, and Roberto Sarmiento, "Highly-parallel GPU architecture for lossy hyperspectral image compression," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 2, pp. 670–681, 2013.

[12] Yubal Barrios, Antonio J Sánchez, Lucana Santos, and Roberto Sarmiento, "SHyLoC 2.0: A Versatile Hardware Solution for On-Board Data and Hyperspectral Image Compression on Future Space Missions," *Ieee Access*, vol. 8, pp. 54269–54287, 2020.

[13] Raúl Guerra, Yubal Barrios, María Díaz, Lucana Santos, Sebastián López, and Roberto Sarmiento, "A new algorithm for the on-board compression of hyperspectral images," *Remote Sensing*, vol. 10, no. 3, pp. 428, 2018.

[14] Raúl Guerra, Yubal Barrios, María Díaz, Abelardo Baez, Sebastián López, and Roberto Sarmiento, "A hardware-friendly hyperspectral lossy compressor for next-generation space-grade field programmable gate arrays," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 12, pp. 4813–4828, 2019.

[15] María Díaz, Raúl Guerra, Pablo Horstrand, Ernestina Martel, Sebastián López, José F. López, and Sarmiento Roberto, "Real-Time Hyperspectral Image Compression Onto Embedded GPUs," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 1–18, 2019.

[16] Julián Caba, María Díaz, Jesús Barba, Raúl Guerra, and Jose A. de la Torre and Sebastián López, "FPGA-Based On-Board Hyperspectral Imaging Compression: Benchmarking Performance and Energy Efficiency against GPU Implementations," *Remote Sensing*, vol. 12, no. 22, 2020.

[17] Paul G Howard and Jeffrey Scott Vitter, "Fast and efficient lossless image compression," in *Data Compression Conference, 1993. DCC'93.* IEEE, 1993, pp. 351–360.

[18] Pablo Horstrand, Raúl Guerra, Aythami Rodríguez, María Díaz, Sebastián López, and José Fco López, "A UAV platform based on a hyperspectral sensor for image capturing and on-board processing," *IEEE Access*, vol. 7, pp. 66919–66938, 2019.

[19] Velleman, "User Manual: DVM1200 - Multimeter with USB interface," Available Online: `https://www.velleman.eu/downloads/1/dvm1200gbnlfresdplit.pdf`, (Accessed on 26 May 2023).

[20] Lucana Santos, José Fco. López, Roberto Sarmiento, and Raffaele Vitulli, "FPGA implementation of a lossy compression algorithm for hyperspectral images with a high-level synthesis tool," in *2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, 2013, pp. 107–114.

[21] Aday García, Lucana Santos, Sebastián López, Gustavo Marrero, José Fco. López, and Roberto Sarmiento, "High level modular implementation of a lossy hyperspectral image compression algorithm on a FPGA," in *2013 5th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2013, pp. 1–4.

[22] Didier Keymeulen, "FPGA implementation of lossless and lossy compression of space-based multispectral and hyperspectral imagery," 2016.

[23] Daniel Fernández, Carlos González, Daniel Mozos, and Sebastián Lopez, "FPGA implementation of the principal component analysis algorithm for dimensionality reduction of hyperspectral images," *Journal of Real-Time Image Processing*, vol. 16, 2019.

[24] Daniel Báscones, Carlos González, and Daniel Mozos, "An Extremely Pipelined FPGA Implementation of a Lossy Hyperspectral Image Compression Algorithm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 10, pp. 7435–7447, 2020.

[25] Daniel Báscones, Carlos González, and Daniel Mozos, "An FPGA Accelerator for Real-Time Lossy Compression of Hyperspectral Images," *Remote Sensing*, vol. 12, no. 16, 2020.

[26] Yubal Barrios, Alfonso Rodríguez, Antonio Sánchez, Arturo Pérez, Sebastián López, Andrés Otero, Eduardo de la Torre, and Roberto Sarmiento, "Lossy Hyperspectral Image Compression on a Reconfigurable and Fault-Tolerant FPGA-Based Adaptive Computing Platform," *Electronics*, vol. 9, no. 10, 2020.