IMPLEMENTACIÓN DE UN ALGORITMO DE TRANSFORMACIÓN WAVELET PARA LA DESCOMPRESIÓN DE DATOS SÍSMICOS EN UNA FPGA

EDHER FABIÁN SÁNCHEZ COLMENARES

Universidad Industrial de Santander

Facultad de Ingenierías Físico-mecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2014

IMPLEMENTACIÓN DE UN ALGORITMO DE TRANSFORMACIÓN WAVELET PARA LA DESCOMPRESIÓN DE DATOS SÍSMICOS EN UNA FPGA

Autor:

EDHER FABIÁN SÁNCHEZ COLMENARES

Trabajo de Grado para optar al título de

Ingeniero Electrónico

Director:

PhD (c) CARLOS AUGUSTO FAJARDO ARIZA

Codirector:

Ing. CARLOS ANDRÉS ANGULO JULIO

Universidad Industrial de Santander

Facultad de Ingenierías Físico-mecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2014

DEDICATORIA

A Dios por darme la sabiduría y guiarme durante todo el proceso de aprendizaje; a mis padres quienes me motivaron y ayudaron a construir este sueño; a mi familia, amigos y mi novia.

AGRADECIMIENTOS

Este trabajo es apoyado por la compañía Colombiana de Petróleos ECOPETROL y COLCIENCIAS como parte del proyecto de investigación No. 0266-2013 y 511/2010. El autor agradece al grupo de investigación CPS de la Universidad Industrial de Santander por su constante apoyo. A mis directores Carlos Angulo y Carlos Fajardo por su presencia incondicional, sus valiosos aportes y sugerencias.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

RESUMEN

TÍTULO: Implementación de un Algoritmo de Transformación Wavelet para la Descompresión de Datos Sísmicos en una FPGA[1]

AUTOR: Edher Fabián Sánchez Colmenares[2]

PALABRAS CLAVES: DWT, Transformación Discreta *Wavelet*, Descompresión, FPGA, Esquema *Lifting*

DESCRIPCIÓN:

Actualmente las exploraciones sísmicas producen una gran cantidad de datos, superando incluso los 100 Terabytes, los cuales deben ser procesados en las estaciones de cómputo locales para su respectivo análisis. Para realizar este procedimiento eficientemente, es necesario que la velocidad a la cual son transmitidos los datos sea acorde al procesamiento de estos, por consiguiente, la técnica de compresión de datos es apta para esta labor. Los algoritmos de compresión son deseables ya que ofrecen una reducción en términos de almacenamiento y ancho de banda de transmisión.

En este proyecto se desarrolló una arquitectura para la descompresión de datos sísmicos, basada en un algoritmo compuesto por tres etapas: Transformación Wavelet Bidimensional Inversa, Cuantificación Uniforme Inversa y Decodificación *Huffman*. Para la etapa de transformación se seleccionó el esquema *Lifting* utilizando un filtro biortogonal 5/3.

El diseño fue desarrollado en el software *ISE Design Suite13.2* de *Xilinx* usando VHDL y luego implementado en una FPGA Virtex 5 - XC5VFX70T. La arquitectura es capaz de descomprimir un conjunto de datos correspondiente a 44 trazas sísmicas con 2000 muestras cada una. Adicionalmente la arquitectura procesa dos datos a la vez y a su vez genera dos datos reconstruidos a una frecuencia máxima de operación de 75.87 MHz.

[1] Trabajo de Grado

[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: PhD (c) Carlos Augusto Fajardo Ariza. Codirector: Ing. Carlos Andrés Angulo Julio

ABSTRACT

TITLE: FPGA Implementation of a Wavelet Transform Algorithm for Seismic Data Decompression[3]

AUTHOR: Edher Fabián Sánchez Colmenares[4]

KEYWORDS: DWT, Discrete *Wavelet* Transform, Decompression, FPGA, *Lifting* Scheme

DESCRIPTION:

Nowadays, seismic exploration produces a vast amount of data, even exceeding one hundred Terabytes, which must be processed in local computer stations for its processing and analyzing respectively . To achieve this procedure with efficiency, it is necessary that the rate at which data is transmitted according to the processing thereof, thus the data compression technique is suitable for this task. Since compression algorithms reduce the amount of storage and transmission bandwidth, they are highly desired in several fields such as seismic data processing.

In this project we developed an architecture for seismic data decompression based on an algorithm composed of three stages: The first one is Inverse Bidimensional Wavelet Transform, the second one is Inverse Uniform Quantization and the third one is Huffman Decoding. For the transformation step the *Lifting* scheme was selected using a biorthogonal 5/3 filter.

The design was developed in the *ISE Design Suite 13.2* software from *Xilinx* using VHDL and then implemented on a Virtex 5 - XC5VFX70T FPGA. The architecture is capable of decompressing a data set corresponding to forty four seismic traces with two thousand samples each one. Additionally architecture processes two data simultaneously and so on generates two reconstructed data at a maximum operating frequency of 75.87 MHz.

[3] Degree project

[4] Faculty of Physical-Mechanical Engineering.Electrical, Electronics and Telecommunications School. Advisor: PhD (c) Carlos Augusto Fajardo Ariza. Co-Advisor: Carlos Andrés Angulo Julio

INTRODUCCIÓN

En la actualidad existen diversos métodos de exploración de hidrocarburos, siendo la reflexión sísmica uno de los más utilizados debido a que ofrece una excelente capacidad para examinar a grandes profundidades, así como para identificar los detalles necesarios del subsuelo [1][2]. Este proceso genera una enorme cantidad de datos (conocidos como trazas sísmicas) capaces de exceder los 100 Terabytes [2], los cuales deben ser transmitidos hacia las estaciones de cómputo locales para su respectivo procesamiento. Así mismo, la cantidad de datos con los que se debe trabajar aumenta con el paso del tiempo y es imprescindible mantener la productividad sin afectar los resultados.

Por otro lado, la velocidad actual para la transmisión de las trazas sísmicas hacia las estaciones no es lo suficientemente alta en comparación a la velocidad de procesamiento, causando que el proceso no sea muy eficiente [3], por consiguiente, es necesario buscar alternativas como la compresión de trazas sísmicas.

Hoy en día, la compresión es una técnica clave para la gestión de trazas sísmicas, ya que otorga una mayor flexibilidad en la administración del espacio de almacenamiento del servidor local o remoto, así como disminuye el tráfico de red, consiguiendo reducir el tiempo empleado en la transmisión [2][4]. Adicionalmente, la Transformación *Wavelet* Discreta (DWT del inglés *Discrete Wavelet Transform*) ha sido distintiva gracias a los trabajos realizados en diversas aplicaciones relacionadas con la geofísica, como eliminación de ruido, análisis de imágenes y compresión [1][5].

En este trabajo se presenta la implementación de un algoritmo de descompresión de trazas sísmicas en una FPGA, basado en una Decodificación *Huffman*, una Cuantificación Uniforme Inversa y una Transformación *Wavelet* 2D Inversa utilizando el esquema *Lifting* con un filtro biortogonal 5/3.

Este artículo está organizado de la siguiente forma: La sección II ofrece una breve descripción de los pasos empleados para la compresión de datos. La sección III muestra la arquitectura propuesta. La sección IV proporciona los resultados obtenidos y finalmente en la sección V se mencionan las conclusiones.

1. COMPRESIÓN DE DATOS

La compresión de datos consiste en transformar una cadena de datos de entrada (X) en otra cadena de datos de salida (X') con menor tamaño. La compresión de datos es importante principalmente por dos razones: (1) reduce la cantidad de bytes que se deben almacenar y (2) disminuye el tiempo empleado en la transferencia de datos [6].

Generalmente, un algoritmo de compresión está compuesto de las siguientes tres etapas:

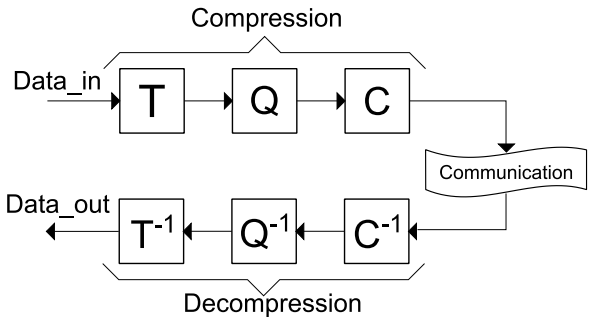
- Transformación (T): Esta etapa permite presentar los datos de una forma más compacta. En términos de la teoría de la información, esta etapa ayuda a disminuir la entropía en los datos.

- Cuantificación (Q): Reduce la cantidad de bits necesarios para representar cada uno de los datos. En esta etapa se produce pérdida de información.

- Codificación (C): Disminuye el número total de bits para representar todo el conjunto de datos.

Para realizar la descompresión se deben ejecutar los procesos inversos respecto a la codificación, cuantificación y transformación. La figura 1 muestra el esquema para la compresión y descompresión de datos.

Figura 1. Esquema Compresión - Descompresión de Datos



1.1 TRANSFORMACIÓN *WAVELET* DISCRETA (DWT)

El concepto matemático de la transformación es una herramienta poderosa que se emplea en muchas áreas, entre ellas la compresión de datos [6]. Como se mencionó anteriormente, la transformación de los datos es el primer paso para la compresión, debido a que tiene como finalidad obtener una representación más compacta de los datos.

Dada una señal que varía con el tiempo, se selecciona un intervalo y se procede a utilizar esta transformación para identificar y aislar las frecuencias que constituyen la señal en ese intervalo [6]. En el caso de una imagen, se debe aplicar una transformación *Wavelet* Bidimensional, obteniendo como resultado la identificación de las bajas frecuencias (correspondiente a las características más relevantes de ésta) y las altas frecuencias (representando los detalles de la imagen).

Existen dos opciones principales para la aplicación de la transformación, una basada en la convolución y otra en el esquema *lifting* [7].

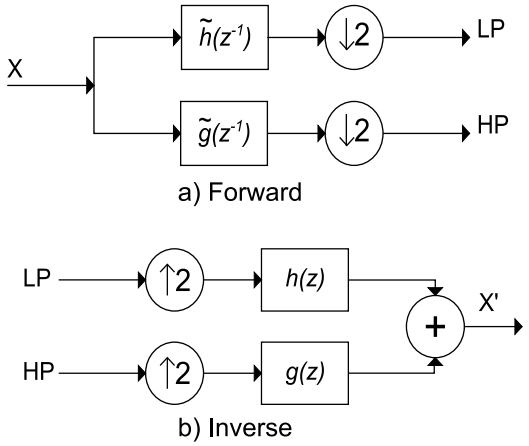
1.1.1 DWT 1D con base en la convolución

Consiste en aplicar dos filtros, $\tilde{h}$ pasa-bajos y $\tilde{g}$ pasa-altos, seguidos por un sobre muestreo (ver Figura 2a).Como resultado se obtiene la descomposición de la señal de entrada (X) en coeficientes de aproximación (LP) y coeficientes de detalle (HP). Este proceso es ineficiente, ya que la mitad de los coeficientes calculados son descartados en el muestreo y el filtrado se realiza a todos los datos.

1.1.2 DWT 1D basada en el esquema *lifting*

El esquema *lifting* reduce los requerimientos computacionales de la transformación, mediante la factorización de la matriz polifásica de los filtros W*avelet* en matrices elementales.

Figura 2. Transformación *Wavelet* Discreta 1D a) Directa b) Inversa



Adaptado de [1]

El análisis de la matriz polifásica está definido en el dominio Z de la siguiente manera:

$$\tilde{P}(z^{-1}) = \begin{bmatrix} H_e(z) & H_o(z) \\ G_e(z) & G_o(z) \end{bmatrix} \tag{1}$$

Donde $H_e(z)$ y $H_o(z)$ denotan los componentes par e impar del filtro pasa-bajos mientras que $G_e(z)$ y $G_o(z)$ denotan los componentes par e impar del filtro pasa-altos. Con base en lo anterior, la descomposición de la entrada Xpuede ser escrita:

$$\begin{bmatrix} LP(z) \\ HP(z) \end{bmatrix} = \tilde{P}(z^{-1}) \begin{bmatrix} X_e(z) \\ X_o(z) \end{bmatrix} \tag{2}$$

Donde $LP(z)$ y $HP(z)$ representan los coeficientes de aproximación y de detalle de la señal respectivamente, mientras que $X_e(z)$ y $X_o(z)$ son los componentes par e impar de $X(z)$.

En [1] se ha demostrado que cualquier par de filtros complementarios ($\tilde{h}$, $\tilde{g}$) se puede factorizar en una secuencia de matrices triangulares - superior e inferior - y una matriz diagonal. La representación de la matriz polifásica de un banco de filtro de *Wavelets* está dada por:

$$\tilde{P}(z) = \begin{bmatrix} \tilde{h}_e(z) & \tilde{h}_o(z) \\ \tilde{g}_e(z) & \tilde{g}_o(z) \end{bmatrix} \quad (3)$$

La matriz $\tilde{P}(z)$ se puede factorizar como:

$$\tilde{P}(z) = \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \prod_{i=1}^{m} \left(\begin{bmatrix} 1 & \tilde{s}_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \tilde{t}_i(z) & 1 \end{bmatrix} \right) \quad (4)$$

Donde $\tilde{s}_i(z)$ y $\tilde{t}_i(z)$ son polinomios de Laurent, m es el número de pasos *lifting*[1]y *K* es una constante diferente de cero.

Teniendo en cuenta lo anterior, el esquema de *lifting* se puede dividir en los siguientes pasos:

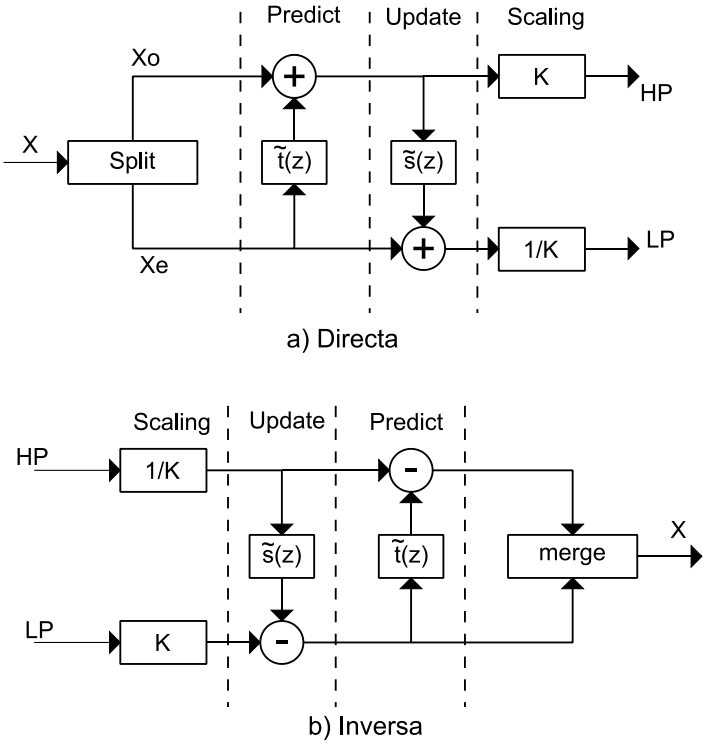
- La etapa de *Split*, donde la señal de entrada *X* es dividida en muestras pares X_e e impares X_o.
- La etapa de *Predict*, en la cual las muestras pares son multiplicadas por los coeficientes del polinomio $\tilde{t}_i(z)$ y se suman con las muestras impares.
- La etapa de *Update*, en que las nuevas muestras impares son multiplicadas por los coeficientes del polinomio $\tilde{s}_i(z)$ y se suman con las muestras pares.
- La etapa de *Scaling*, donde se multiplican las nuevas muestras pares e impares por *K* y *1/K* respectivamente.

La transformación inversa se realiza cambiando los signos de los coeficientes de los polinomios $\tilde{s}_i(z)$ y $\tilde{t}_i(z)$ e intercambiando los factores de escala *K* por *1/K* y viceversa. Los esquemas de transformación *lifting* directa e inversa se muestran en la Figura 3.

.

[1] Un paso *lifting* consiste en realizar las etapas *Update* y *Predict*.

Figura 3. DWT 1D usando esquema *Lifting* a) Directa b) Inversa



Adaptado de [2]

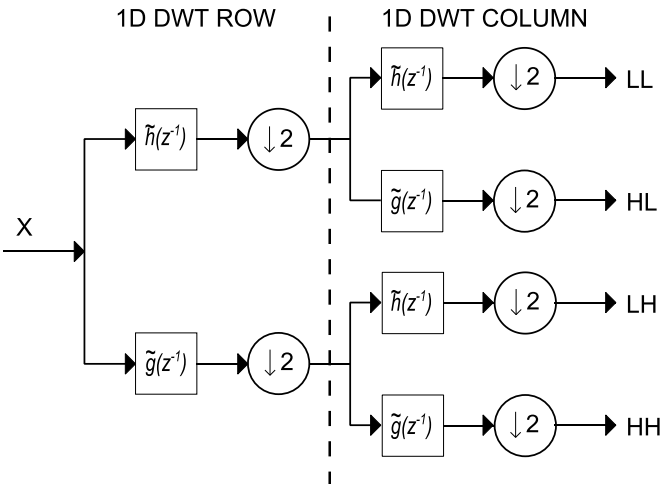
1.1.3 DWT 2D con base en la convolución

Cuando se trabaja con imágenes o matrices es necesaria una transformación 2D. En este caso, la transformación se obtiene al realizar dos veces la DWT 1D, primero por filas y luego por columnas. Esto conduce a 4 diferentes sub-bandas HH, HL, LH y LL, conocidos como coeficientes de aproximación y coeficientes de detalle horizontal, vertical y diagonal respectivamente [3](Figura 4).

1.1.4 DWT 2D basada en el esquema *lifting*

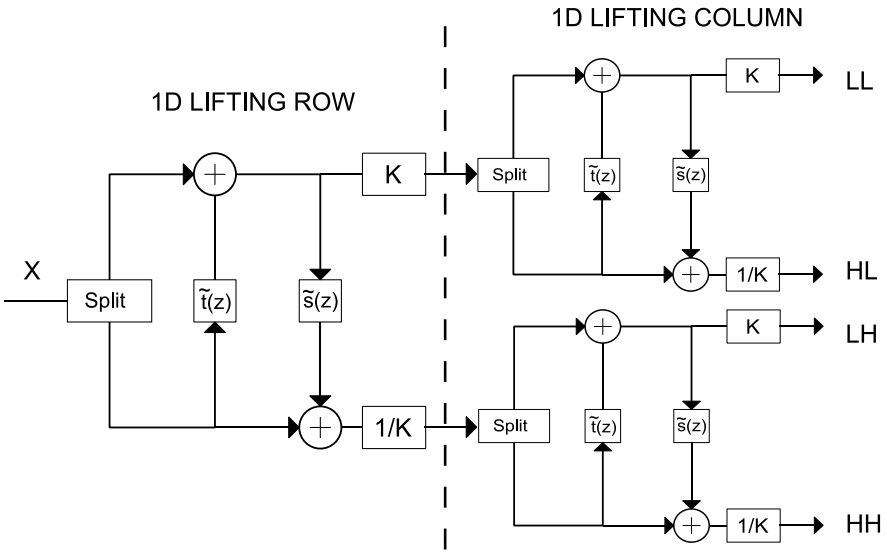
El procedimiento para realizar la transformación utilizando el esquema *lifting* obedece la misma regla que el basado en convolución, es decir, primero por filas y luego por columnas, como se muestra en la Figura 5.

Figura 4. Descomposición *Wavelet* 2D



Adaptado de [4]

Figura 5. Esquema *lifting* para la descomposición *Wavelet* 2D.



Para el filtro *Wavelet* 5/3, los filtros de análisis son:

$$\tilde{h}(z) = -\frac{1}{8}z^{-2} + \frac{1}{4}z^{-1} + \frac{3}{4}z^{0} + \frac{1}{4}z^{1} - \frac{1}{8}z^{2} \quad (5)$$

$$\tilde{g}(z) = -\frac{1}{2}z^{-2} + z^{-1} - \frac{1}{2}z^{0} \quad (6)$$

Usando la ecuación (3) la matriz polifásica es:

$$\tilde{P}(z) = \begin{bmatrix} -\frac{1}{8}z^{-1} + \frac{3}{4} - \frac{1}{8}z & \frac{1}{4} + \frac{1}{4}z \\ -\frac{1}{2}z^{-1} - \frac{1}{2} & 1 \end{bmatrix} \tag{7}$$

La cual se puede factorizar de la siguiente manera:

$$\tilde{P}(z) = \begin{bmatrix} \tilde{h}_e^{new}(z) & \frac{1}{4} + \frac{1}{4}z \\ \tilde{g}_e^{new}(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \tilde{t}(z) & 1 \end{bmatrix} \tag{8}$$

Luego, es necesario encontrar los polinomios de Laurent $\tilde{t}(z)$, $\tilde{h}_e^{new}(z)$ y $\tilde{g}_e^{new}(z)$ tal que:

$$-\frac{1}{8}z^{-1} + \frac{3}{4} - \frac{1}{8}z = \tilde{h}_e^{new}(z) + \tilde{t}(z)\left(\frac{1}{4} + \frac{1}{4}z\right) \tag{9}$$

$$-\frac{1}{2}z^{-1} - \frac{1}{2} = \tilde{g}_e^{new}(z) + \tilde{t}(z)\left(-\frac{1}{2}\right) \tag{10}$$

Los polinomios $\tilde{h}_e^{new}(z)$ y $\tilde{t}(z)$ se hallan dividiendo $\tilde{h}_e(z)$ entre $\tilde{h}_o(z)$, donde el cociente corresponde a $\tilde{t}(z)$ y el residuo a $\tilde{h}_e^{new}(z)$. Una vez conseguido el polinomio $\tilde{t}(z)$, se realiza el mismo procedimiento para hallar $\tilde{g}_e^{new}(z)$dando como resultado:

$$\tilde{t}(z) = -\frac{1}{2}z^{-1} - \frac{1}{2} \tag{11}$$

$$\tilde{h}_e^{new}(z) = 1 \tag{12}$$

$$\tilde{g}_e^{new}(z) = 0 \tag{13}$$

Finalmente la matriz polifásica queda:

$$\tilde{P}(z) = \begin{bmatrix} 1 & \frac{1}{4} + \frac{1}{4}z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z^{-1} - \frac{1}{2} & 1 \end{bmatrix} \quad (14)$$

De aquí que para el filtro 5/3 se tenga que

$$\tilde{t}(z) = -\frac{1}{2}(z^{-1} + 1) \quad (15)$$

$$\tilde{s}(z) = \frac{1}{4}(1 + z) \quad (16)$$

1.2 CODIFICACIÓN HUFFMAN

La codificación *Huffman* es un método que comprime y codifica un conjunto de datos por medio del uso de una tabla de códigos de longitud variable o la construcción de un árbol binario [5]. La tabla y el árbol se derivan con base en la probabilidad de ocurrencia de cada símbolo. La Tabla 1 muestra un ejemplo en el cual se ha determinado la frecuencia de un conjunto de datos.

Tabla 1. Frecuencia de los símbolos

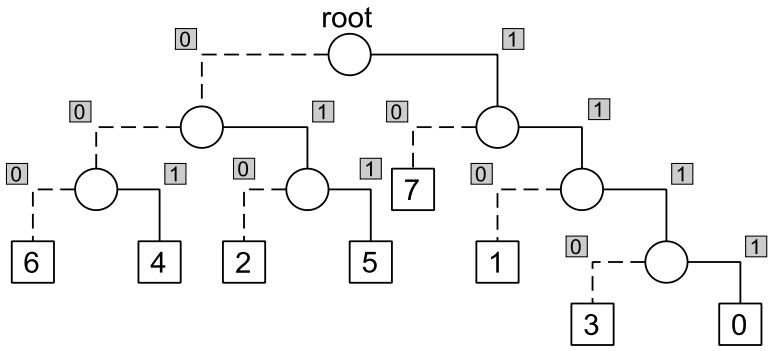
Símbolo	Frecuencia
0	4
1	8
2	5
3	4
4	5
5	5
6	5
7	16

Adaptada de[6]

La Figura 6 muestra el árbol binario, producto de la Tabla 1, en donde se observa que se asigna un valor de ‘0’ y ‘1’ a las ramas izquierda y derecha de cada nodo respectivamente [6]. Para esto se debe tener en cuenta que los símbolos con menor frecuencia están ubicados en los nodos más inferiores y los de mayor frecuencia en los nodos superiores. A partir del árbol se generan caminos desde el nodo *root* (aquel que está encima de todos) hacia los nodos inferiores, creando los códigos

que serán asignados a cada símbolo como se muestra en la Tabla 2.En este documento llamaremos *Diccionario* a la unión de los códigos y los símbolos.

Figura 6. Árbol Binario *Huffman*



Tomada de [6]

Tabla 2. Datos codificados a partir del Algoritmo *Huffman*

Símbolo	Código	Longitud Código
0	1111	4
1	110	3
2	010	3
3	1110	4
4	001	3
5	011	3
6	000	3
7	10	2

Tomada de [12]

1.3 CUANTIFICACIÓN UNIFORME

La Cuantificación es una técnica descompresión con pérdida, pues dado un conjunto de datos que se encuentra dentro de un rango específico, estos son aproximados a un valor único, con lo cual se pierde la calidad de la señal. La forma más sencilla de cuantificar es la cuantificación escalar uniforme (USQ del inglés

Uniform Scalar Quantization) y consiste en que los valores aproximados están igualmente espaciados entre ellos [7].

El proceso de cuantificación tiene como parámetros de entrada el vector de datos (X) y el número de bits con el que se desea cuantificar (n). Este proceso se puede desarrollar mediante los siguientes pasos:

- Se calculan el valor máximo (max) y mínimo (min) del vector de entrada.
- A cada dato del vector X se le resta el valor mínimo y el resultado (X_r) es operado con la siguiente ecuación:

$$X_{cuantificado} = round\left(\frac{X_r * (2^n - 1)}{max}\right) \quad (17)$$

$$X_r = X - min \quad (18)$$

El proceso de cuantificación inversa se puede desarrollar mediante la siguiente fórmula:

$$X' = \frac{X_{cuantificado} * max}{2^n - 1} + min \quad (19)$$

Cabe señalar que, durante el proceso de compresión de datos, la cuantificación agrega ruido a las muestras, ocasionando que la Relación Señal–Ruido (SNR) disminuya.

2. TRABAJO REALIZADO

El desarrollo del método de compresión de datos se dividió en 2 partes: (1) La compresión se realizó en un computador utilizando MATLAB y (2) la descompresión se implementó en una FPGA. Para ambos procesos se tuvo como referencia la Figura 1.

2.1 PROCESO DE COMPRESIÓN

Inicialmente se utilizó el software MATLAB para realizar la compresión y descompresión de trazas sísmicas. Los *scripts* desarrollados para la transformación se realizaron sin el uso de los comandos propios del software, esto con el propósito de familiarizarse con el algoritmo y establecer estrategias para su posterior implementación en la FPGA. Este proceso se realizó en primera instancia para una traza sísmica, a la cual se le aplicó el algoritmo de compresión 1D, es decir, la etapa de transformación fue la DWT 1D. Posteriormente se trabajó con datos sísmicos 2D a los cuales se les aplicó un algoritmo de compresión 2D, es decir, la etapa de transformación fue DWT 2D.

En el proceso de compresión (figura 1), la transformación *Wavelet* tiene como entrada los datos sísmicos y como salida arroja las cuatro sub-bandas HH, HL, LH y LL. La cuantificación uniforme tiene como entradas, las cuatro sub-bandas y la cantidad de bits *(n)* con la que se cuantifica cada dato y se consigue como salida los datos cuantificados. Finalmente, en la codificación *Huffman* ingresan los datos cuantificados y se obtienen los datos codificados y el *Diccionario* necesario para la decodificación.

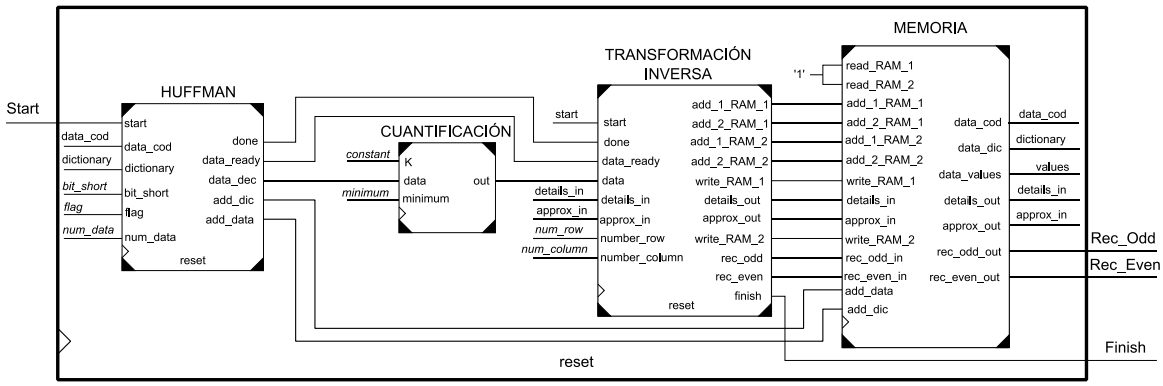
2.2 PROCESO DE DESCOMPRESIÓN EN LA FPGA

Se propuso una arquitectura computacional con el propósito de descomprimir en una FPGA un conjunto de datos correspondientes a M trazas sísmicas con N muestras por traza. La arquitectura propuesta es mostrada en la Figura 7.

2.2.1 Circuito Decodificación *Huffman*

Para la decodificación de los datos se utilizó el trabajo desarrollado en [14]. El circuito *HUFFMAN* tiene como entradas las señales ***start*** (para dar inicio al proceso), ***data_cod*** y ***dictionary*** (correspondientes a los datos comprimidos y el diccionario necesario para su descompresión), y los parámetros de compresión ***bit_short***, ***flag*** y ***num_data*** (Figura 7).

Figura 7. Circuito RECONSTRUCCIÓN



Las salidas del circuito son: ***data_dec*** (datos decodificados), ***add_data*** y ***add_dic*** (correspondientes a las direcciones de memoria donde se encuentran almacenados los datos comprimidos y el diccionario), la señal ***done*** que informa cuándo se termina la decodificación, y la señal ***data_ready*** que indica que en ***data_dec*** se encuentra un dato decodificado válido.

2.2.2 Circuito Cuantificación Inversa Uniforme

A esta etapa ingresa el dato decodificado proveniente de *HUFFMAN* y se realiza el proceso inverso de cuantificación. El circuito *CUANTIFICACIÓN* implementa este proceso mediante la siguiente ecuación:

$$out = (K * data) + minimum \qquad (20)$$

2.2.3 Circuito Transformación *Wavelet* Inversa 2D

La arquitectura propuesta para la transformación inversa es realizada por el circuito *TRANSFORMACIÓN_INVERSA*, mostrada en la figura 8, está dividida en: CONTROL, *GENERADOR_DIRECCIONES*, *DWT_1D_COLUMNAS*, *DWT_1D_FILAS* y cuatro registros.

En la Figura 9, se muestra el circuito de *CONTROL*, el cual está compuesto por cuatro máquinas de estado (FSM). *FSM_TOP* da inicio a la primera etapa (transformación 1D por columnas) activando la señal ***start_column*** (la cual habilita el circuito *LIFT_1D_COLUMNAS*).*FSM_TOP* quedará esperando a que se active la señal ***end_column***, la cual indica que éste proceso finalizó. Una vez finalizada dicha fase, *FSM_TOP* dará inicio a la segunda etapa (transformación 1D por filas). Se habilitará *LIFT_1D_FILAS* (por medio de la señal ***start_row***) y esperará a que se active la señal ***end_row***, señalando la finalización de este último proceso.

El circuito *LIFT_1D_COLUMNAS* controla el proceso de transformación por columnas, mediante la habilitación de los contadores y registros de los circuitos *DWT_1D_COLUMNAS* y *GENERADOR_DIRECCIONES*. Adicionalmente, controla la escritura de datos en *RAM_1*, que es la memoria en la que guardarán los resultados obtenidos en la primera etapa de transformación.

Figura 8. Circuito TRANSFORMACIÓN_INVERSA

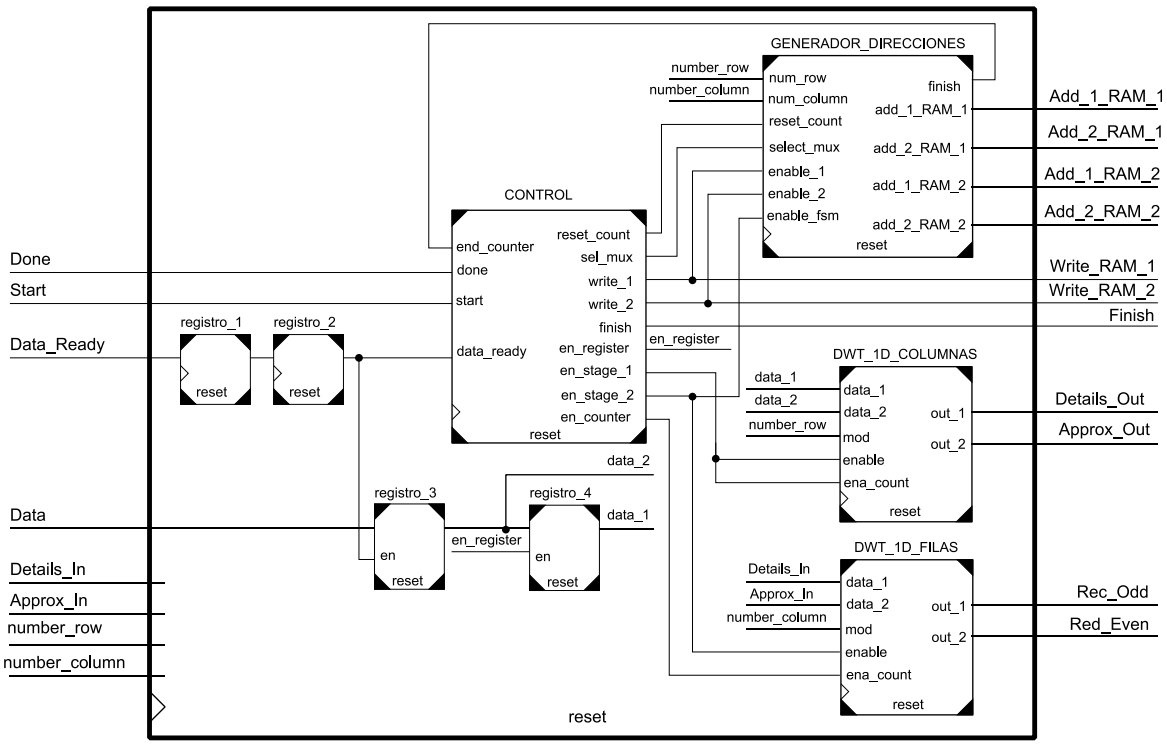
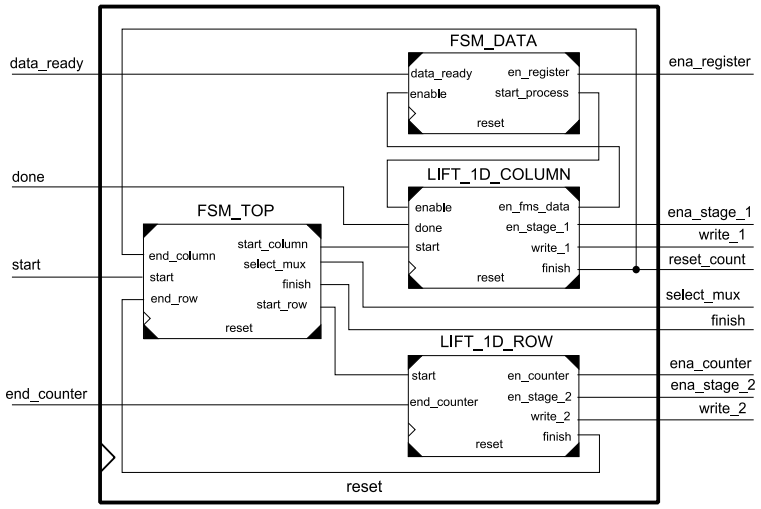


Figura 9. Circuito CONTROL



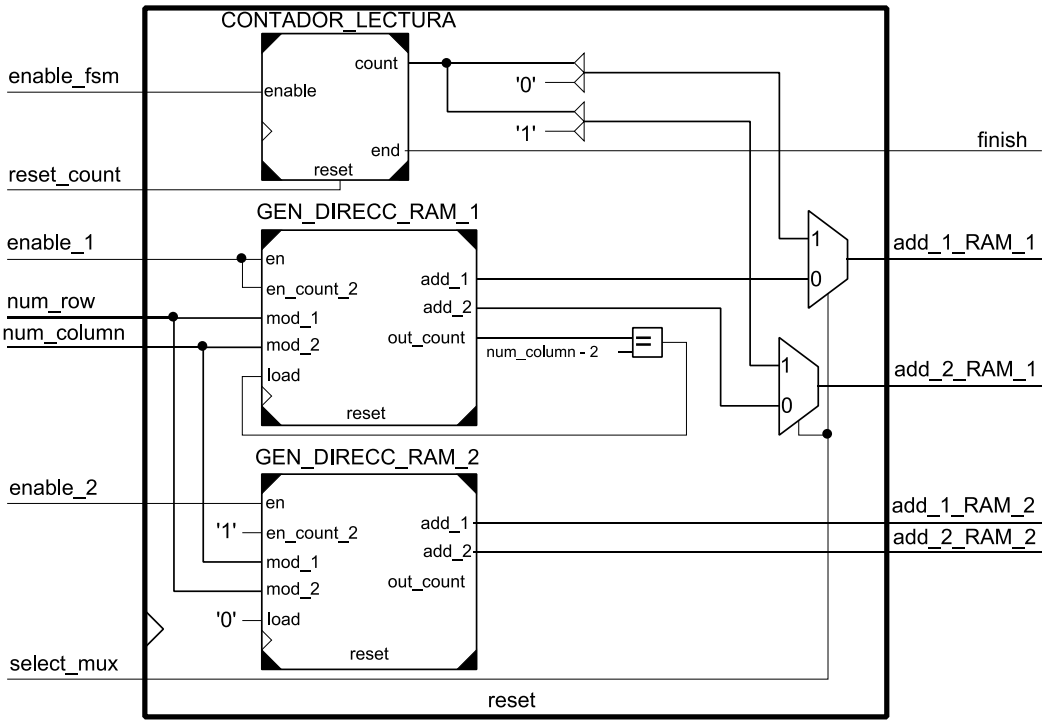
La arquitectura propuesta para la transformación, puede procesar dos datos simultáneamente en cada ciclo de reloj. Sin embargo, debido a que el circuito

HUFFMAN sólo entrega un dato cada *k* ciclos de reloj, fue necesario crear una máquina de estados (*FSM_DATA*) que notificara cuándo se tienen 2 datos para trabajar. Cabe mencionar que *k* depende de la codificación *Huffman*, por lo tanto no es contante.

El circuito *LIFT_1D_FILAS* controla el proceso de transformación por filas, mediante la habilitación de los contadores y registros de los circuitos *DWT_1D_FILAS* y *GENERADOR_DIRECCIONES*, e indica cuando el proceso de la transformación ha finalizado. Este circuito también controla el almacenamiento de los resultados de la segunda etapa de transformación en la memoria *RAM_2*.

El circuito *GENERADOR_DIRECCIONES* (Figura 10) proporciona las direcciones de las memorias *RAM_1* y *RAM_2*. Están conformados principalmente por los circuitos *GEN_DIRECC_RAM_1*, *GEN_DIRECC_RAM_2* y un contador. La señal ***Finish*** se activa cuando el contador alcance el valor $(N \times M)/2$, indicando el final de la etapa de descompresión.

Figura 10. Circuito GENERADOR_DIRECCIONES



Los circuitos *GEN_DIRECC_RAM_1* y *GEN_DIRECC_RAM_2* están compuestos por dos contadores, un multiplexor, dos sumadores y dos registros cada uno (Figura 11). La entrada ***module_1*** corresponde a los valores $N/2$ y $M/2$ respectivamente, mientras que, la entrada ***module_2*** está sujeta a los valores de *M y N* respectivamente.

El análisis realizado con ayuda del *script* desarrollado en MATLAB para llevar a cabo la transformación inversa, nos condujo a establecer reglas para el acceso a las memorias según la etapa que se esté ejecutando (Figura 12). Los números en esta figura representan el orden en que se accede cada par de datos y su posición en la matriz simboliza el patrón de lectura o escritura. Estos patrones obedecen a que los resultados obtenidos en la primera etapa deben ser escritos por columnas en la memoria *RAM_1*, mientras que para la segunda etapa, estos datos deben ser leídos por filas. Por otro lado, los resultados de la segunda etapa son escritos por columnas en la memoria *RAM_2*. Cabe resaltar que las memorias *RAM_1* y *RAM_2* fueron configuradas como memorias *dual port* y los colores en la Figura 12 corresponden a cada puerto de datos.

Teniendo en cuenta lo anterior, en el circuito *GEN_DIRECC_RAM_1*, el circuito *CONTADOR_2* se incrementa de dos en dos, con el propósito de escribir los datos por columna intercalada en *RAM_1*; mientras que en el circuito *GEN_DIRECC_RAM_2*, cuenta de uno en uno con el fin de escribir los datos por columna en *RAM_2*. Adicionalmente, el circuito *CONTADOR_LECTURA* se encarga de generar las direcciones para la lectura de datos de *RAM_1*.

En el circuito *GEN_DIRECC_RAM_1*, la señal ***load*** varía entre ‘0’ y ‘1’ porque, durante el proceso de escritura de la *RAM_1*, es necesario empezar en la primera columna par, cada vez que el contador se reinicie, mientras que esto no es necesario para el proceso de escritura que controla el circuito *GEN_DIRECC_RAM_2*, razón por la cual la señal ***load*** siempre es cero en este último circuito.

Figura 11. Circuito GEN_DIRECC_RAM

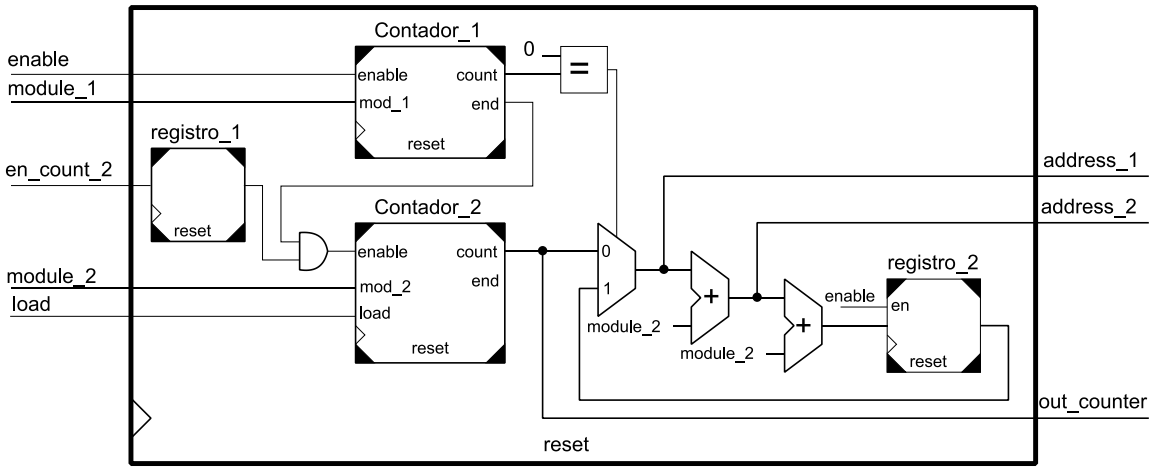
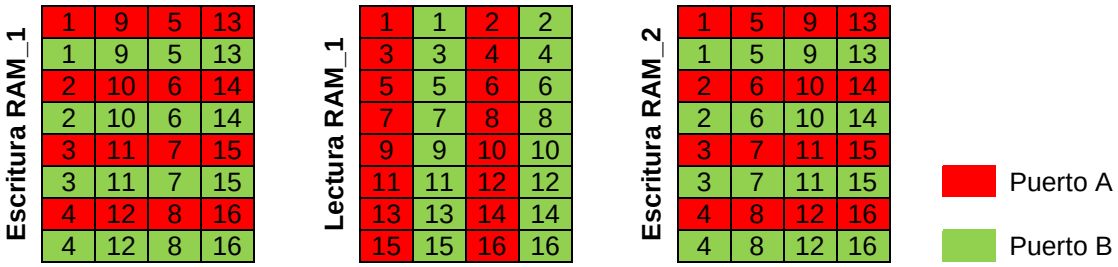


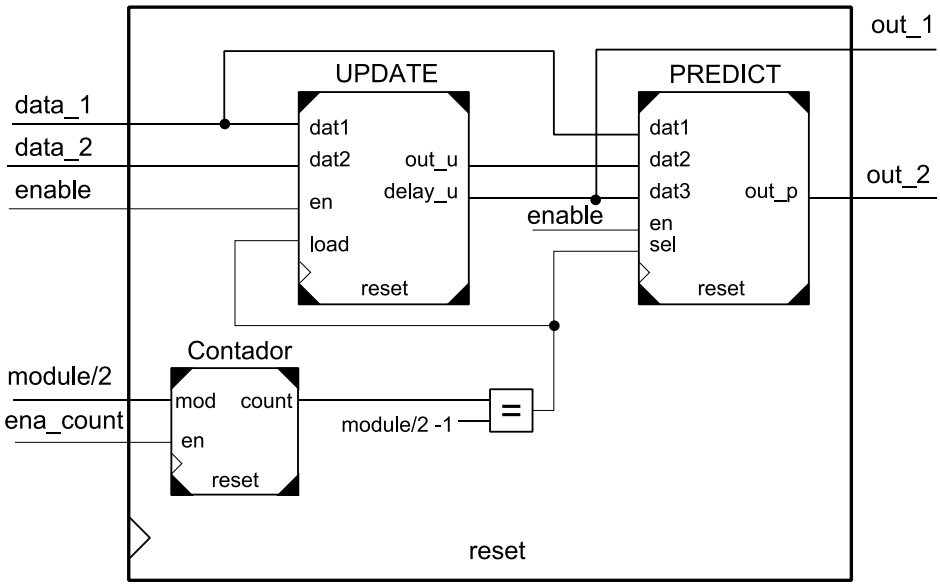
Figura 12. Patrón de lectura/escritura en las memorias



El circuito *DWT_1D* (Figura 13), es empleado dos veces y tiene como finalidad, la reconstrucción de los coeficientes de *Detalle-Aproximación* en la primera etapa y la reconstrucción de los datos *Impares-Pares* en la segunda etapa. El circuito *DWT_1D* está compuesto por los bloques *UPDATE y PREDICT* principalmente.

Para la primera etapa, los datos de entrada a *DWT_1D* provienen del circuito *CUANTIFICACIÓN* y la señal ***module*** corresponde a $N/2$. Las salidas son la recuperación de los *Detalles* y *Aproximaciones*. Para la segunda etapa, los datos de entrada a *DWT_1D* corresponden a la salida de la primera etapa y ***module*** será $M/2$.Las salidas son la reconstrucción de los *datos Impares* y *Pares*.

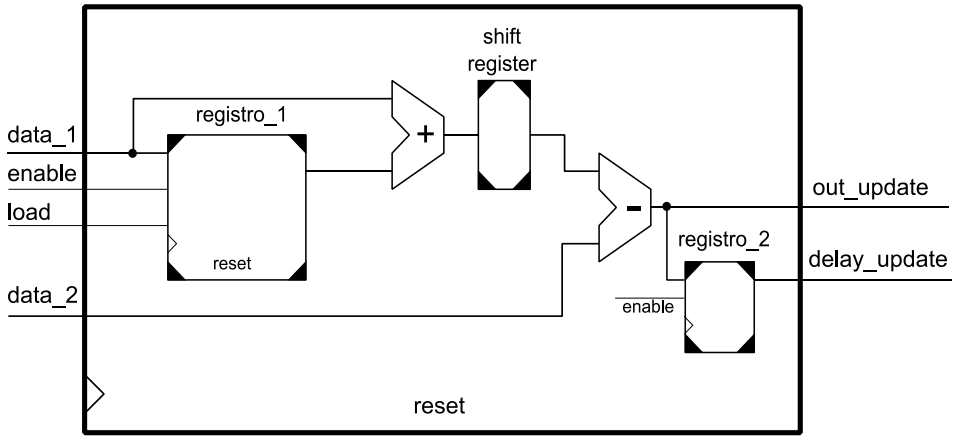
Figura 13. Circuito DWT_1D



Las señales de habilitación ***enable*** y ***ena_count***, son proporcionadas por los circuitos *LIFT_1D_COLUMNAS* y *LIFT_1D_FILAS*, dependiendo de la etapa que se esté ejecutando.

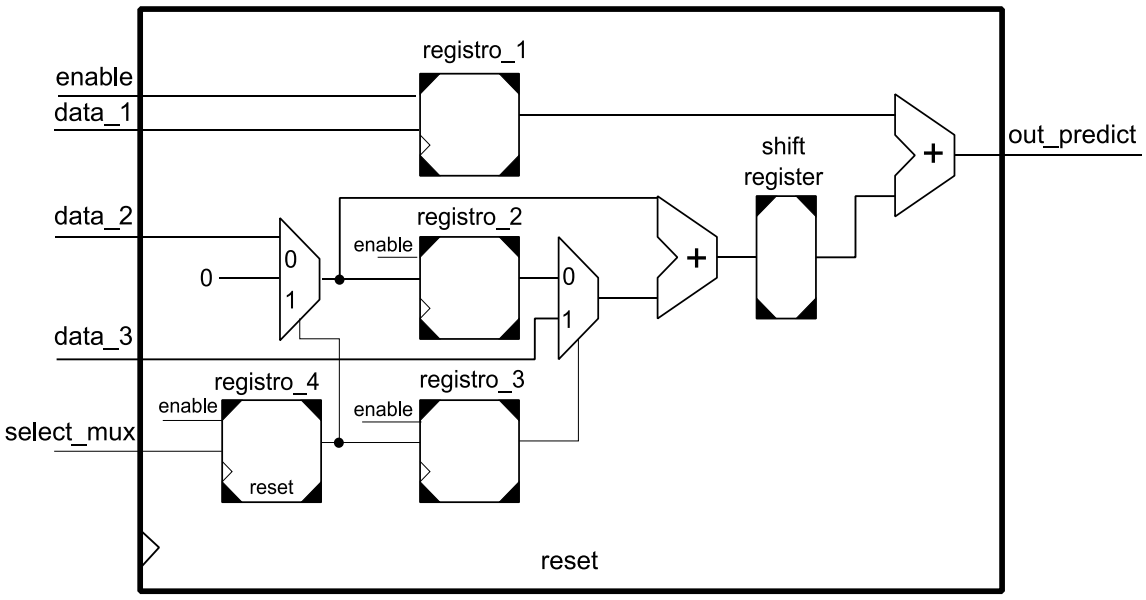
El circuito *UPDATE* (Figura 14) se encarga de la reconstrucción de los *Detalles* en la primera etapa y de los *datos impares* en la segunda etapa. *UPDATE* está compuesto por dos registros, dos sumadores y un registro de desplazamiento encargado de la multiplicación por $1/4$.

Figura 14. Circuito UPDATE



El circuito *PREDICT* (Figura 15) se encarga de la reconstrucción de las *Aproximaciones* en la primera etapa y de los *datos pares* en la segunda etapa. *PREDICT e*stá compuesto de cuatro registros, dos sumadores, dos multiplexores y un registro de desplazamiento encargado de la multiplicación por $1/2$.

Figura 15. Circuito PREDICT

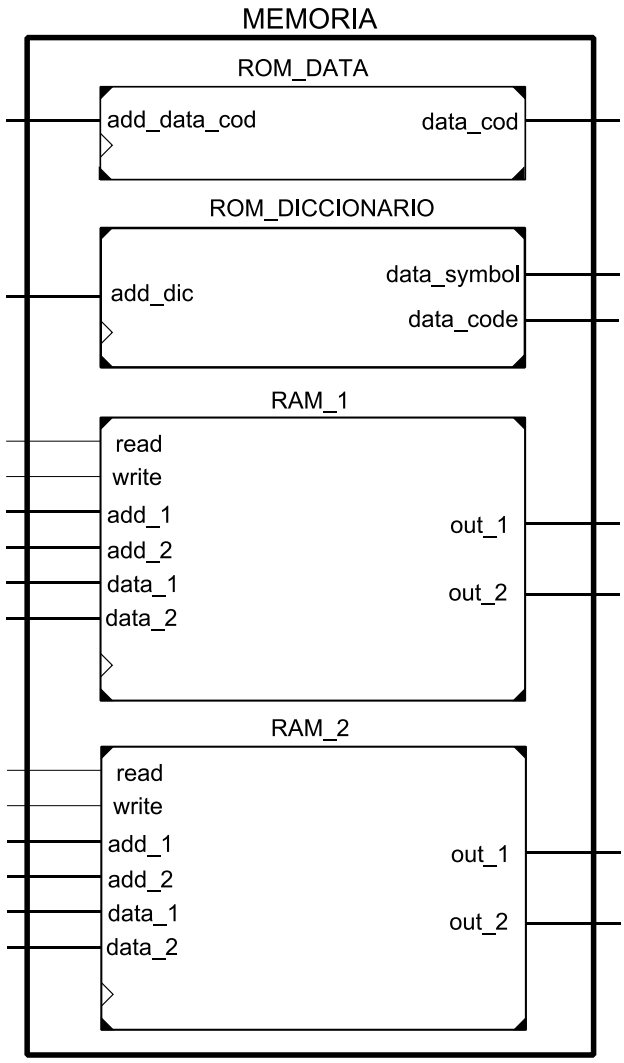


2.2.4 Circuito memorias

La Figura 16 muestra la composición interna del circuito *MEMORY*. Las memorias *ROM_DICCIONARIO* y *ROM_DATA*, contienen el diccionario y los datos codificados generados en la compresión.

En *RAM_1* y *RAM_2* se almacenaron la reconstrucción de los coeficientes de *Detalle-Aproximación* y *Datos Impares* y *Pares* respectivamente. Las señales de escritura (***write***) de estas memorias, son habilitadas por el circuito *CONTROL*, dependiendo de la etapa que se está ejecutando. Las señales de lectura (***read***) se mantuvieron en '1' durante todo el proceso.

Figura 16. Circuito MEMORIA



3. RESULTADOS

El diseño fue desarrollado en el software *ISE Design Suite13.2* de *Xilinx* usando VHDL y luego implementado en una FPGA Virtex 5 - XC5VFX70T. La Tabla 3 muestra algunos de los recursos utilizados para la descompresión de dos trazas sísmicas de 2000 muestras cada una, mientras que en la Tabla 4 se presenta la cantidad de ciclos de reloj necesarios. Cabe señalar que los procesos de transformación (1a etapa) y cuantificación toman pocos ciclos de reloj, debido a que son realizados en paralelo con la decodificación. Adicionalmente, la Tabla 5 exhibe el tiempo empleado para efectuar la descompresión en una CPU y en la FPGA.

Tabla 3. Recursos utilizados en la FPGA

# Datos (N x M)	#Slices Registers	#Slices LUTs	# block RAM/FIFO	Memoryused [KB]
2000 x 2	720 (1%)	1564 (3%)	19 (12%)	612 (11%)

Tabla 4. Ciclos de Reloj

Total Descompresión	C^{-1}	Q^{-1}	T^{-1}(1ª Etapa)	T^{-1} (2ª Etapa)
150578	148568	2	4	2004

Tabla 5. Tiempos Descompresión

CPU (Intel® Core™ i5-2430M @ 2.40 GHz)	101 [ms]
FPGA (Frecuencia operación = 75MHz)	2 [ms]

Para la verificación de los resultados obtenidos en *ISE*, inicialmente se desarrolló un algoritmo de descompresión de trazas sísmicas utilizando el software *MATLAB*, en donde se comparó nuestro algoritmo programado con los comandos propios para asegurar su correcto funcionamiento. En *ISE* se simuló y generó un archivo de texto con los datos reconstruidos, para contrastar y verificar que concordaran con los datos obtenidos en *MATLAB*. Luego se implementó el circuito en la FPGA y

mediante la herramienta *Chipscope*, se hizo un seguimiento a las señales que nos permitieran corroborar que los resultados mostrados en la simulación, correspondieran con los obtenidos en la FPGA.

La arquitectura propuesta para la transformación procesa dos datos a la vez y a su vez genera dos datos reconstruidos, Detalles-Aproximaciones para la primera etapa e Impares-Pares para la segunda etapa. La implementación de la transformación en un conjunto de datos de tamaño *N x M* requiere *2 x N x M* operaciones de desplazamiento (para realizar las multiplicaciones) y *4 x N x M* sumas. Teniendo en cuenta que la FPGA Virtex 5 empleada en este proyecto posee una capacidad máxima de almacenamiento de 5328 *Kibits* en bloques RAM, se estimó que el dispositivo es capaz de descomprimir un conjunto de datos correspondiente a 44 trazas sísmicas con 2000 muestras cada una.

La latencia para la obtención de los datos de la primera etapa está sujeta a la decodificación, debido a que éste entrega sólo un dato cada *k* ciclos de reloj, donde *k* depende de la codificación de cada dato; mientras que la latencia de la segunda etapa es de 4 ciclos de reloj. El diseño emplea *(N x M)/2 + 4* ciclos de reloj para computar la reconstrucción de los datos pares e impares (segunda etapa) para datos de tamaño *N x M.* El software ISE reporta para la arquitectura propuesta una frecuencia de operación máxima de 75.88 MHz.

4. CONCLUSIONES

Se ha diseñado una arquitectura que realiza la descompresión de trazas sísmicas en una FPGA. La implementación ejecuta la Decodificación *Huffman*, Cuantificación Inversa Uniforme y la DWT 2D Inversa, utilizando el esquema *lifting.* El diseño se ha descrito en VHDL e implementado en una FPGA. El software *ISE* reporta para la arquitectura propuesta una frecuencia de operación máxima de 75.879 MHz.

El algoritmo de descompresión, presenta un cuello de botella en la Decodificación *Huffman*, debido a que este sólo entrega un dato cada *k* ciclos de reloj mientras que, la transformación se diseñó para procesar dos datos al mismo tiempo. Por lo tanto, se sugiere que se implemente otro módulo *Huffman*, con el fin de procesar más rápido los datos.

La arquitectura propuesta posee varias memorias, sin embargo, es posible eliminar la memoria *RAM_1* si, una vez obtenidos los primeros resultados de la primera etapa (reconstrucción *Detalles* y *Aproximaciones*) estos se procesan inmediatamente, para obtener los *datos impares* y *pares* reconstruidos, en lugar de esperar a que finalice la primera etapa.

REFERENCIAS BIBLIOGRÁFICAS

[1] M. F. Khene and S. H. Abdul-Jauwad, “Adaptive seismic compression by wavelet shrinkage,” pp. 544–548, 2000.

[2] W. Wu, Z. Yang, Q. Qin, and F. Hu, “Adaptive Seismic Data Compression Using Wavelet Packets,” *2006 IEEE Int. Symp. Geosci. Remote Sens.*, no. 3, pp. 787–789, Jul. 2006.

[3] C. Fajardo and J. Castillo, “Reducción de los tiempos de cómputo de la Migración Sísmica usando FPGAs y GPGPUs: Un artículo de revisión,” vol. 9, no. 17, pp. 261–293, 2013.

[4] M. A. Al-Moohimeed, “Towards an Efficient Compression Algorithm for Seismic Data,” *Radio Sci. Conf. 2004.*, pp. 550–553, 2004.

[5] A. Z. Averbuch, F. Meyer, J. O. Stromberg, R. Coifman, and A. Vassiliou, “Low Bit-Rate Efficient Compression for Seismic Data.,” *IEEE Trans. Image Process.*, vol. 10, no. 12, pp. 1801–14, Dec. 2001.

[6] D. Salomon, *Data Compression The Complete Reference*, 4th ed. Springer, 2007.

[7] M. E. Angelopoulou, K. Masselos, P. Y. K. Cheung, and Y. Andreopoulos, “Implementation and Comparison of the 5/3 Lifting 2D Discrete Wavelet Transform Computation Schedules on FPGAs,” *J. Signal Process. Syst.*, vol. 51, no. 1, pp. 3–21, Oct. 2007.

[8] I. Daubechies and W. I. M. Sweldens, “Factoring wavelet transforms into lifting steps,” no. November, 1997.

[9] K. Andra, C. Chakrabarti, and T. Acharya, “A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform,” vol. 50, no. 4, pp. 966–977, 2002.

[10] K. Ranjeet, A. Kumar, and R. K. Pandey, “An efficient compression System for ECG signal using QRS periods and CAB technique based on 2D DWT and Huffman Coding,” pp. 1–6.

[11] H.-A. Pham, V.-H. Bui, and A.-V. Dinh-Duc, “An Adaptive Huffman Decoding Algorithm for MP3 Decoder,” *2010 Fifth IEEE Int. Symp. Electron. Des. Test Appl.*, pp. 153–157, 2010.

[12] S. Beak, B. Van Hieu, H. Lee, S. Choi, I. Kim, K. Lee, Y. Lee, and T. Jeong, “Novel binary tree Huffman decoding algorithm and field programmable gate array implementation for terrestrial-digital multimedia broadcasting mobile handheld,” *IET Sci. Meas. Technol.*, vol. 6, no. 6, p. 527, 2012.

[13] J. Yu, “Advantages of Uniform Scalar Dead-zone Quantization in Image Coding System,” pp. 20–23, 2004.

[14] C. Angulo, C. Fajardo, O. M. Reyes, and J. Castillo Villar, “FPGA Implementation of a Huffman Decoder for High Speed Seismic Data Decompression,” *2014 Data Compression Conf.*, no. 0266, pp. 396–396, Mar. 2014.

BIBLIOGRAFÍA

ABDUL-JAUWARD, S.H. y KHENE, M.F. Adaptive seismic compression by wavelet shrinkage. 2000. p. 544–548.

AL-MOOHIMEED, Mohammed A. Towards an Efficient Compression Algorithm for Seismic Data. En: Radio Science Conference. 2004. p. 550–553.

ANDRA, Kishore; CHAKRABARTI, Chaitali, Member IEEE y ACHARYA, Tinku, Senior Member IEEE. A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform. Abril 2002. vol. 50, No. 4. p. 966–977.

ANGELOPOULOU, Maria E. Implementation and Comparison of the 5/3 Lifting 2D Discrete Wavelet Transform Computation Schedules on FPGAs. Oct. 2007vol. 51, No. 1. p. 3–21.

ANGULO, Julio, *et al.* FPGA Implementation of a Huffman Decoder for High Speed Seismic Data Decompression. En: Data Compression Conference. Marzo, 2014. No. 0266. p. 396–396.

AVERBUCH, Amir Z, *et al*. Low Bit-Rate Efficient Compression for Seismic Data. Dec. 2001. vol. 10, No. 12. p. 1801–14.

BEAK, S, *et al*. Novel binary tree Huffman decoding algorithm and field programmable gate array implementation for terrestrial-digital multimedia broadcasting mobile handheld. En: IET Science, Measurement & Technology. 26 Diciembre, 2012. vol. 6, No. 6. p. 527.

DAUBECHIES, Ingrid y SWELDENS, Wim. Factoring wavelet transforms into lifting steps. Nov. 1997.

FAJARDO, Carlos; CASTILLO VILLAR, Javier y PEDRAZA, César. Reducción de los tiempos de cómputo de la Migración Sísmica usando FPGAs y GPGPUs: Un artículo de revisión. 2013. vol. 9, No. 17. p. 261–293.

PHAM, Hoang-Anh; BUI, Van-Hieu y DINH-DUC, Anh-Vu. An Adaptive Huffman Decoding Algorithm for MP3 Decoder. En: Fifth IEEE International Symposium on Electronic Design, Test & Applications. 2010. p. 153–157.

RANJEET, K, Student Member IEEE.; KUMAR, A. y PANDEY, Rajesh K. An efficient compression System for ECG signal using QRS periods and CAB technique based on 2D DWT and Huffman Coding. p. 1-6.

SALOMON, David. Data Compression The Complete Reference. 4 ed. Springer, 2012.

WENBO, Wu, *et al*. Adaptive Seismic Data Compression Using Wavelet Packets. En: IEEE International Symposium on Geoscience and Remote Sensing. Jul. 2006. p. 787–789

YU, Jinhua. Advantages of Uniform Scalar Dead-zone Quantization in Image Coding System. 2004. p. 20–23.