IMPLEMENTACIÓN DE LA TRANSFORMADA WAVELET SOBRE UN SISTEMA EMBEBIDO PARA EL PRE-PROCESAMIENTO DE SEÑALES UNIDIMENSIONALES NO ESTACIONARIAS

JEFERSON RUIZ SALAZAR. 1115280

DAVID ALEXANDER OREJUELA CAICEDO. 1120113

UNIVERSIDAD DE SAN BUENAVENTURA CALI

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA ELECTRÓNICA

CALI, 2016

IMPLEMENTACIÓN DE LA TRANSFORMADA WAVELET SOBRE UN SISTEMA EMBEBIDO PARA EL PRE-PROCESAMIENTO DE SEÑALES UNIDIMENSIONALES NO ESTACIONARIAS

JEFERSON RUIZ SALAZAR. 1115280

DAVID ALEXANDER OREJUELA CAICEDO. 1120113

INFORME DE PROYECTO DE GRADO PARA OPTAR AL TÍTULO DE

INGENIERO ELECTRÓNICO

Director

Ph.D. JOSÉ FERNANDO VALENCIA MURILLO

Co Director

ING. DANIEL FELIPE VALENCIA VARGAS

UNIVERSIDAD DE SAN BUENAVENTURA CALI

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA ELECTRÓNICA

CALI, 2016

Nota de Aceptación:

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Universidad de San Buenaventura Cali para optar al título de ingeniero(a) electrónico(a).

_____

Firma del Jurado

_____

Firma del Jurado

_____

Director

_____

Co director

Santiago de Cali, Noviembre de 2016.

AGRADECIMIENTOS

Agradecemos a todas las personas que nos brindaron ese importante acompañamiento a partir de sus conocimientos y observaciones para lograr llevar a cabo tanto la construcción de este proyecto como la elaboración de nuestra formación como ingenieros. Docentes de numerosas facultades, laboratoristas, ingenieros y a nuestro director Ph.D. José Fernando Valencia Murillo y Co director el Ing. Daniel Felipe Valencia Vargas, por ser verdaderos guías que nos brindaron carisma, conocimiento y disposición en todo momento.

TABLA DE CONTENIDO

pág.

0. INTRODUCCIÓN 10
0.1. Descripción del problema 12
0.2. Justificación 14
0.3. Objetivos 15
0.3.1. General 15
0.3.2. Específicos. 15
1. MARCO DE REFERENCIA 16
1.1. Antecedentes 16
1.2. Marco teórico 18
1.2.1. Matlab 18
1.2.2. HDL coder 19
1.2.3. Field Programmable Gate Array (FPGA) 19
1.2.4. FPGA IN LOOP (FIL) 20
1.2.5. Registro de corrimiento 20
1.2.6. Representaciones tiempo-frecuencia 20
1.3. Marco conceptual 21
1.3.1. Tipos de señales. 21
1.3.2. Pre-procesamiento de señales. 22
1.3.3. Transformada wavelet. 22
1.3.4. Métodos de umbralización 38
1.4. Bitácora de investigación 41
2. PROPUESTA Y MODELO TEÓRICO 44
2.1. Métodos para implementar la TW 44
2.1.1. Método Convolucional 44
2.1.2. Método Lifting 46
2.2. Eliminación de ruido mediante la TW 48
2.2.1. Descomponer 50
2.2.2. Seleccionar el nivel del umbral 53
2.2.3. Reconstruir la señal 54

2.3. TW Haar 59
2.4. FPGA 59
2.4.1. DE0-Nano 60
2.5. Modelo teórico 60
3. DESARROLLO DEL MODELO 64
3.1. HDL coder 64
3.2. Variables persistentes 64
3.3. Algoritmo TW Haar en Matlab 65
3.4. Calculo del umbral 66
3.5. Calculo de la MAD 66
3.6. Umbralización dura 67
3.7. Test bench 67
3.8. Conversión a VHDL 68
4. PRUEBAS Y ANÁLISIS DE LOS RESULTADOS 70
4.1. Verificación mediante FPGA IN LOOP 71
5. CONCLUSIONES 78
6. REFERENCIAS. 80
7. APENDICE 84
7.1. Código en Matlab para HDL CODER 84
7.2. Código LWT Haar 85
7.3. Código ILWT Haar 85
7.4. Código etapa de thresholding 85
7.5. Testbench del código en Matlab 86
7.6. Código en VHDL generado por HDL coder 87

LISTA DE TABLAS

pág.

Tabla 1. Resumen de las Características y propiedades asociadas a las Wavelets más comunes..... 24
Tabla 2. Características de las Wavelets Daubechies..... 30
Tabla 3. Características de las Wavelets Symlets..... 32
Tabla 4. Características de la Wavelet Morlet. 33
Tabla 5. Características de la Wavelet Mexican Hat. 34
Tabla 6. Características de la Wavelet Meyer. 35
Tabla 7. Características de la Wavelet Gaussiana. 37
Tabla 8. Tipo de wavelet y dispositivo en el que distintos autores han implementado la TW 42
Tabla 9. Calculo de los coeficientes detallados por el método convolucional. 45
Tabla 10. Calculo de los coeficientes aproximados por el método convolucional. 46
Tabla 11. Calculo de la reconstrucción de la parte impar. 46
Tabla 12. Calculo de la reconstrucción de la parte par..... 46
Tabla 13. Funcionamiento de la función merge..... 48
Tabla 14. Descomposición en niveles de la señal del EEG..... 52
Tabla 15. Comparación entre la señal original y la señal con ruido gaussiano. 57
Tabla 16. Comparación entre la señal original y la señal recuperada mediante DWT 58
Tabla 17.Trabajos que emplean la eliminación de ruido con la TW sobre una FPGA. 77

LISTA DE FIGURAS

pág.

Figura 1. Funcionamiento de HDL Coder. 19
Figura 2. (a) Señal unidimensional proveniente de un electrocardiograma (b) señal bidimensional proveniente de una fotografía. 22
Figura 3. Función wavelet Haar..... 27
Figura 4. Función de escala asociada a la función wavelet Haar. 27
Figura 5. Familia wavelet Daubechies. 29
Figura 6. Función de escala (db4) asociada a la función wavelet (db4). 29
Figura 7. Familia wavelet Symlets. 31
Figura 8. Función de escala (sym4) asociada a la función wavelet (sym4). 31
Figura 9. Función wavelet Morlet. 32
Figura 10. Función wavelet Mexican Hat. 33
Figura 11. Función wavelet Meyer. 35
Figura 12. Función de escala asociada a la función wavelet Meyer. 35
Figura 13. Familia wavelet Gaussian. 36
Figura 14. Estructura método de convolución. 44
Figura 15. Estructura de reconstrucción método convolucional. 45
Figura 16. Estructura método lifting. 47
Figura 17. Estructura de reconstrucción método lifting 47
Figura 18. Señal del EEG muestreada a 100Hz 49
Figura 19. Señal del EEG muestreada a 100Hz con ruido gaussiano de -10dB ... 50
Figura 20. Esquema de descomposición multi resolución. 51
Figura 21. Coeficientes de la señal con ruido gaussiano. 52
Figura 22. Coeficientes después del umbralizado suave 54
Figura 23. Esquema de reconstrucción multi resolución. 54
Figura 24. Reconstrucción de la señal del EEG 55
Figura 25.Comparación entre la señal original y la señal reconstruida. 56
Figura 26. Comparación entre la señal reconstruida y la señal original con ruido gaussiano. 57
Figura 27. RMSE promedio al eliminar el ruido de una señal con el método SOFT 58
Figura 28. RMSE promedio al eliminar el ruido de una señal con el método HARD 59
Figura 29. FPGA DE0-nano 60
Figura 30. Diagrama de flujo de la eliminación de ruido por medio de la TW. 61
Figura 31. Modelo teórico implementado en hardware. 62
Figura 32. Declaración de las variables persistentes s y z. 65
Figura 33. Descomposición mediante la TW Haar en el esquema lifting en Matlab. 65
Figura 34. Reconstrucción mediante la TW Haar en el esquema lifting en Matlab. 66

Figura 35. Calculo de la media aritmética de los coeficientes resultantes en la descomposición. 66
Figura 36. Calculo de la desviación estándar absoluta y la MAD. 67
Figura 37. Umbralización dura códificada. 67
Figura 38. Código de prueba para la TW con HDL coder. 68
Figura 39. Selección del tipo datos. 69
Figura 40. Definición de las señales de entrada del bloque de VHDL 69
Figura 41. Conexiones aceptadas por FIL. 70
Figura 42. Código para añadir la ruta de la versión 15.0 de Quartus II a Matlab utiliza en este proyecto. 70
Figura 43. Selección de la FPGA y la comunicación a utilizar con Simulink. 70
Figura 44. Selección de las señales de entrada y salida del módulo en VHDL. ... 71
Figura 45. Selección del tipo de dato de salida para la verificación del módulo en VHDL. 71
Figura 46. Bloque de VHDL verificado en Simulink. 72
Figura 47. Señal del sujeto ay con ruido gaussiano con SNR de 20db. 73
Figura 48. Señal original y señal recuperada del sujeto ay 74
Figura 49. Señal del sujeto aa con ruido Gaussiano a 20db..... 75
Figura 50. Señal del sujeto aa original y reconstruida. 75
Figura 51. Disminución de RMSE con la TW Haar. 76
Figura 52. Aumento de PSNR con la TW Haar. 76

0. INTRODUCCIÓN

La Transformada Wavelet (TW) es una técnica de procesamiento de señales no estacionarias, encargada de mapear una señal transitoria en una representación de tiempo-frecuencia. Este método permite el filtrado y extracción de características fundamentales de una señal (como son tiempo, frecuencia, amplitud, etc.) por medio de un análisis multi-resolución, en cual permite analizar con detalle las características de interés.

Este trabajo corresponde a la implementación en hardware de una investigación previa presentada por los autores de este documento en el simposio internacional de tratamiento de señales, imágenes y visión artificial (STSIVA) 2016 [1], la cual consiste en la definición de un método óptimo de umbralización para eliminar ruido realizado a través de la plataforma Matlab, en síntesis, mediante el presente informe se busca implementar el algoritmo óptimo en un sistema embebido, y de esta manera darle otro repertorio de aplicaciones a la TW.

Tanto la TW como la Transformada de Fourier (TF), son herramientas utilizadas para procesar una señal con el fin de modificar su naturaleza, observar patrones fácilmente o analizar comportamientos, sin embargo, la TW destaca por conservar los detalles importantes de cualquier tipo de la señal con algoritmos que pueden llegar a ser menos exigentes en términos de recursos computacionales en comparación con la TF.

Para responder a la necesidad de un diseño eficiente que permita conservar información importante de una señal no estacionaria mediante la TW, se ha estructurado este documento en 6 capítulos, que se describen de la siguiente manera:

El capítulo 0, contextualiza al lector en cuanto al conjunto de problemas de la implementación de la TW, ilustrando el origen y los alcances del proyecto, además se plantean los objetivos que conducen a implementar la TW para el pre-procesamiento de señales.

En el capítulo 1, se definen los términos pertinentes en este proyecto, como son la definición de la TW, descripción de los tipos de señales, los métodos de umbralización para la eliminación de ruido, el dispositivo en que se ha embebido la TW y las herramientas que se utilizaron para el desarrollo del proyecto.

En el capítulo 2, se sustenta la selección de la clase de wavelet, el método para implementar la clase de wavelet, el tipo de umbralización, el método para aplicar el umbral y el dispositivo seleccionado. Por otro lado, en el capítulo 3 se encuentra el desarrollo con HDL coder de la TW con la estructura de eliminación de ruido y su verificación con FIL.

Más adelante, en el capítulo 4, se encuentran las pruebas realizadas al sistema desarrollado con las métricas RMSE y PSNR. Finalmente, en el capítulo 5 se indican las conclusiones obtenidas con la realización de este proyecto.

0.1. Descripción del problema

Es importante reconocer que toda señal del mundo físico no existe sin ruido, en ocasiones este puede ser imperceptible, ahora bien, existen muchos casos en que el ruido se manifiesta de manera evidente en una señal y debe ser eliminado para realizar un correcto análisis de la información, como, por ejemplo: en el análisis de señales o imágenes médicas, minería de datos, radioastronomía, entre otro gran abanico de aplicaciones.

Cada aplicación tiene sus características específicas que se deben tomar con extrema precaución dando como ejemplo el análisis de señales o imágenes médicas, si el proceso de eliminación del ruido no cuida los detalles que caracterizan la señal como lo haría un simple filtro pasa-bajas, se perderá información de alta relevancia, convirtiendo la etapa de eliminación de ruido en un proceso que está modificando la naturaleza de la señal a analizar, algo no deseado en el manejo de información clínica.

La TW es eficiente para el análisis de señales biomédicas, mapeando la señal en una representación de tiempo-escala; se diferencia de la TF por proveer un análisis con resolución adaptable, en donde, las ventanas angostas son utilizadas para realizar el análisis de datos con altas frecuencias y las ventanas anchas se utilizan para analizar información de baja frecuencia [2], este cálculo requiere de un algoritmo que permita la entrega de resultados con cortos tiempos de espera.

La implementación de la TW puede realizarse sobre distintas bases: software en un computador multitarea, hardware programable o ejecutarse mediante instrucciones en un sistema con capacidad de procesamiento, esto permite tener diferentes opciones para realizar la TW en su análisis multi-resolución. La complejidad de implementar la TW nace debido a la elevada cantidad de operaciones que deben ser realizadas [3], por consiguiente, la elección de un formato de datos adecuado para realizar las operaciones, la base en la que se va a implementar la TW y el tipo de TW, son factores importantes para implementar el algoritmo.

El principal campo de acción de este trabajo de grado se encuentra enfocado en el área de la medicina, donde señales fisiológicas no estacionarias como las del EEG, ECG y EOG son de interés para mejorar el análisis e interpretación de eventos clínicos, no obstante en el momento de analizar la señal se presentan artefactos que dificultan obtener la información clínica [4]. Por esto, se propone una etapa de filtrado o pre-procesamiento a través de la TW, que permita suprimir la información indeseada de tal forma que el ruido de las señales del EEG sea reducido para mejorar su visualización [5].

En este proyecto de grado se plantea el algoritmo wavelet aplicado al pre-procesamiento de señales unidimensionales no estacionarias, las cuales están definidas por ser variantes en el tiempo y presentar una única variable

independiente, como puede ser, una señal sonora o señal biológica (BS). El algoritmo se ha implementado sobre un sistema embebido, el cual recibe la señal y posteriormente ejecuta la TW para realizar la respectiva eliminación de ruido de la misma y finalmente entrega la señal con una disminución de ruido.

Considerando los enunciados anteriores acerca de la TW en el ámbito del procesamiento digital de señales, es oportuno plantear la pregunta, ¿Cómo se puede realizar el pre-procesamiento de señales unidimensionales no estacionarias usando la TW mediante un sistema embebido?

0.2. Justificación

La falta de efectividad de la TF en el análisis de señales no estacionarias [6], en conjunto con la complejidad de observar señales debido al ruido que contienen, ha causado en aplicaciones que requieren alta fidelidad en la información, la necesidad de establecer bloques dedicados a mejorar la calidad de la señal transitoria de interés, por lo que en este proyecto se propone una etapa basada en la TW que permita la eliminación de ruido de una señal unidimensional.

Si se utiliza la TF para simbolizar un evento en el eje del tiempo no se puede representar el instante exacto en el que, cierta frecuencia cambia, por añadidura cabe destacar que se puede utilizar la transformada de Fourier de tiempo corto (STFT) la cual utiliza ventanas para analizar el espectro de una señal y brinda información tanto del eje del tiempo como del eje de la frecuencia. Sin embargo, el tamaño uniforme de las ventanas limita la resolución de la frecuencia, esto conlleva a la posible pérdida de información importante, por otro lado la TW utiliza ventanas con resolución adaptable (multi-resolución), volviendo ideal su implementación para aplicaciones en las que se requieren conservar/analizar características muy importantes de la señal, como por ejemplo, en el análisis del EEG [6].

El uso de la STFT no es óptimo para describir señales variantes el tiempo, debido a que causa un gran número de coeficientes aritméticos para representar un evento, esto conlleva a un alto número de operaciones que debe realizar la CPU y por ende a un consumo de potencia elevado; en su lugar la TW revela el comportamiento de la señal debido a la amplitud de pocos coeficientes, lo cual permite el diseño de un algoritmo computacional eficiente [7].

Con la realización de este proyecto se busca facilitar el pre-procesamiento de las señales unidimensionales no estacionarias mediante la implementación de la TW, en efecto, la idea de implementar la transformada sobre en un sistema embebido se realiza para aportar funciones de portabilidad, reducción de costos y consumo de potencia en comparación con los sistemas que implementan la TW mediante un software existente en un sistema operativo de uso general.

0.3. Objetivos

0.3.1. General

Implementar la Transformada Wavelet sobre un sistema embebido que permita el pre-procesamiento de señales unidimensionales no estacionarias.

0.3.2. Específicos.

Establecer la clase de Transformada Wavelet que mejor rendimiento logre representar para la etapa de pre-procesamiento.

Analizar algoritmos que permitan la representación efectiva de la clase de Transformada Wavelet seleccionada en el sistema embebido elegido.

Establecer el dispositivo electrónico que de manera eficiente logre embeber la Transformada Wavelet.

Desarrollar la codificación correspondiente al algoritmo que represente la Transformada Wavelet seleccionada, en un lenguaje comprensible por el sistema embebido designado.

Implementar el código desarrollado en el sistema embebido.

Validar el funcionamiento del sistema embebido.

Divulgar los resultados mediante un artículo que sintetice la investigación y muestre con claridad el desarrollo durante el trabajo de grado.

1. MARCO DE REFERENCIA

1.1. Antecedentes

La TW es una herramienta matemática utilizada en el procesamiento de señales la cual se ha adaptado a diferentes estándares (JPEG2000, DJVU, Dirac), esta herramienta presenta ventajas frente a otros métodos evidenciando un buen desempeño en el análisis de señales, como se observa en las siguientes investigaciones.

En el año 2008, Chilo y Lindblad, proponen la implementación de la TW para señales unidimensionales mediante una FPGA que permite la clasificación de señales infrasonido [1]. Para procesar el flujo continuo de los datos y extraer información o características óptimas, es importante que la clasificación de señales se realice en tiempo real, para ello se desarrolla y se utiliza la TW. El objetivo del trabajo radica en el diseño e implementación de la TW discreta en una FPGA para el procesamiento de datos infrasonido en tiempo real. Los autores realizan el diseño en VHDL, la implementación en una FPGA y se simula en el software QUARTUS II [8].

En el año 2011, Rein S y Reisslein, presentan un tutorial que introduce la comunicación, en el procesamiento de señales por medio de una serie de técnicas de la TW. Los autores explican calcular la TW con la aritmética de punto fijo. En esta aplicación la TW está basada en la wavelet Haar y Daubechies, y son implementadas en un microprocesador dsPIC30F4013, el cual es un controlador digital de señales de 16 bit con 2 kByte de memoria RAM [9].

En el año 2013, autores como K. Iyer (*et. al.*) se enfocan en mejorar la producción de energía eólica distribuida (DWPG), mediante generadores de inducción auto-excitados (SEIG). Un SEIG autónomo accionado por turbina eólica es capaz de suministrar energía, en particular en las zonas remotas, a las cargas domésticas, industriales, y agrícolas. El problema que muestran los autores se debe a los principales desafíos que presenta un SIEG; como son su pobre regulación de voltaje (VR) y frecuencia. Además, un reto adicional que se presenta en los sistemas DWPG es la detección de fallos (FD), ya que estos en el funcionamiento conducen a pérdidas económicas y cortes de energía. Los autores para solucionar los problemas mencionados anteriormente proponen un algoritmo basado en la Transformada Wavelet Discreta (TWD), aplicado en un sistema integrado de bajo costo. La TWD está basada en una daubechies implementada en un sistema que consta de tres controladores dsPIC33F16b a una velocidad de 80 MHz, con capacidad de 40-MIPS [10].

[1] El infrasonido es un fenómeno acústico de baja frecuencia que normalmente oscila entre 0.01 a 20 Hz.

En el año 2014, Y. Qassim (*et. al.*), implementan la función wavelet Morlet, en una FPGA Spartan 3AN. Los cálculos de la CWT se realizaron en el espacio de Fourier y se ejecutaron sobre la FPGA después de varios esquemas de optimización. La memoria usada en el sistema propuesto se optimizo almacenando únicamente los valores distintos de cero que retorna la función wavelet. Según los autores, esto reduce 89% de la memoria de almacenamiento y permite implementar todo el diseño en la FPGA. El diseño propuesto se probó usando datos de EEG y demostró ser adecuado para la extracción de características de los potenciales relacionados con eventos. Como se mencionó anteriormente, el diseño utiliza las técnicas del espacio de Fourier y emplea varios métodos de optimización para mejorar la velocidad y reducir significativamente los requerimientos de recursos. Los autores además enuncian que el diseño propuesto, no se limita a una función wavelet fijo, sino que también se puede adaptar fácilmente a diferentes funciones wavelet sin la necesidad de resintetizar o rediseñar el circuito [3].

En el 2015, F. Canbay (*et. al.*), estudian y aplican el algoritmo transformada wavelet complejo de doble árbol (DTTWC) en una FPGA, con el fin de realizar la extracción de características y eliminación de ruido para señales biomédicas (SB) en tiempo real. Las SB son observaciones que llevan información acerca de los sistemas biológicos y fisiológicos tales como el sistema cardiovascular, sistema respiratorio y el sistema nervioso. En la arquitectura propuesta por los autores, DTTWC se implementa con un único sumador y un multiplicador, en donde además la arquitectura está diseñada para N canales en paralelo [11]. Los coeficientes resultantes del algoritmo propuesto implementado en FPGA también son comparados con los coeficientes obtenidos a partir de la aplicación de MATLAB DTTWC para la verificación. Los resultados de mejor exactitud se lograron con la arquitectura propuesta por los autores [11].

En el año 2015, M. Schimmack (*et. al.*), trabajan en la eliminación de ruido en señales obtenidas a través de una serie de sensores de temperatura ubicados en la superficie de la piel. Ellos presentan un sistema embebido termosensible con un bus de sensores, los cuales miden los cambios de temperatura de la piel, y utiliza una DWT para los fines de localización y eliminación de ruido de la señal. El sistema integrado está basado en el sistema de Broadcom BCM2836, que cuenta con una unidad de cuatro núcleos ARM Cortex-A7 central de procesamiento (CPU) con una unidad de procesamiento de doble núcleo de gráficos VideoCore IV (GPU). El método propuesto se construyó sobre la base de algoritmos de wavelet de la biblioteca WaveLab 850 de la Universidad de Stanford (EE.UU.), utilizando la familia wavelet Haar para este proceso [12].

Por otra parte, S. Chen y Y. Chen, en el año 2015, introducen la TWD en una FPGA, para la eliminación de ruido en tiempo real de señales médicas. Los autores diseñaron una arquitectura de procesamiento digital de señales asociada con el esquema de la TWD. Esta arquitectura está basada en un circuito que se compone principalmente de tres módulos en cascada: una etapa de descomposición TWD,

un novedoso esquema umbral (etapa intermedia) y una reconstrucción mediante la inversa de la TWD (ITWD). Los autores indican, que, en un circuito de este tipo, la aplicación en tiempo real de la TWD, requiere un almacenamiento de memoria sustancialmente menor, en comparación con el requerido por el cálculo directo de TWD. Para el experimento y simulaciones los autores utilizaron un conjunto de señales del electrocardiograma (ECG). Los resultados obtenidos indicaron que el esquema propuesto basado en la TWD utilizando la wavelet Daubechies, no solamente cumplía el requisito de procesamiento en tiempo real, sino también permite la reducción del nivel de ruido [13].

1.2. Marco teórico

1.2.1. Matlab

La plataforma MATLAB es un entorno de desarrollo o lenguaje de alto nivel utilizado para analizar, diseñar y resolver problemas científicos y de ingeniería. Esta es una herramienta capaz de realizar diferentes funciones y operar en diversos campos, por ejemplo, procesamiento de señales, procesamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica, cálculos numéricos, modelado, simulación, entre otros, debido a su gran cantidad de librerías o de familias de comandos de áreas específicas llamadas toolbox que posee. Los toolbox son comandos (archivos M) que extienden el ambiente de MATLAB para resolver problemas de áreas específicas de la ciencia e ingeniería. Por ejemplo, existen toolbox para las áreas de procesamiento digital de señales, sistemas de control, redes neuronales, Wavelets, etc.

MATLAB es un sistema interactivo cuyo elemento básico de datos es el arreglo, es decir el lenguaje está basado en matrices, permitiendo expresar y resolver problemas de matemáticas computacionales. MATLAB se puede integrar con otros lenguajes, lo que le permite desplegar algoritmos y aplicaciones con otros entornos de desarrollo.

1.2.2. HDL coder

HDL Coder es un toolbox de Matlab con la característica de generar código Verilog y VHDL sintetizable para FPGAs y diseños ASIC, directamente desde la plataforma Matlab o Simulink como se ilustra en la Figura 1 [14].

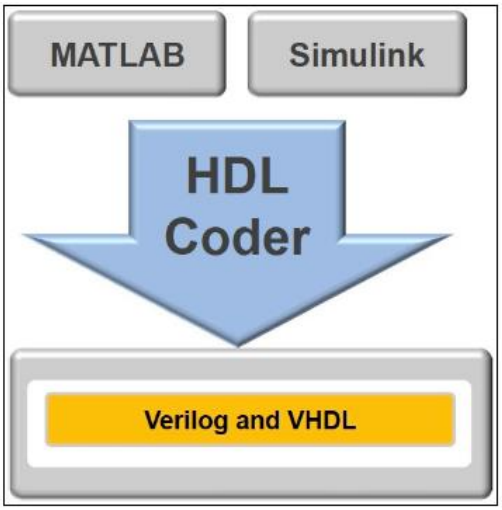


Figura 1. Funcionamiento de HDL Coder.

1.2.3. Field Programmable Gate Array (FPGA)

Es un dispositivo que ha logrado satisfacer necesidades que han surgido en las aplicaciones de procesamiento digital de señales, como por ejemplo el requerimiento de una gran cantidad de recursos. La FPGA integra grandes cantidades de bloques lógicos programables conectados matricialmente con canales de interconexión. El bloque lógico es la unidad básica del dispositivo. La matriz configurable, otorga una gran cantidad de flexibilidad, lo que permite obtener recursos como una RAM o un registro de corrimiento. Posee características como:

- **Alta Velocidad.** El tiempo de ejecución de un diseño es mucho menor en comparación a otras tecnologías como ASIC o microcontroladores, puesto que la tecnología FPGA está basada en bloques lógicos programables, los que permiten realizar operaciones en paralelo, logrando de esta manera la reducción en tiempos de procesamiento y ejecución.

- **Desempeño.** Tomando ventaja del hardware paralelo, las FPGA superan la potencia de cálculo de los procesadores de señales digitales (DSP) rompiendo el paradigma de ejecución secuencial y logrando más por ciclo de reloj.

- **Mantenimiento a largo plazo.** Las FPGA son arreglos de compuertas actualizables y reconfigurables en campo, las cuales pueden seguir el ritmo de modificaciones en una aplicación específica.

1.2.4. FPGA IN LOOP (FIL)

FIL incorpora y proporciona la capacidad de utilizar los entornos de desarrollo software Matlab o Simulink, para permitir la verificación rápida y eficiente de los diseños de HDL. FIL realiza los siguientes procesos:

- Genera un bloque FIL u objeto del sistema FIL que representa el código HDL
- Proporciona síntesis, mapeo lógico, generación de archivos de programación y el canal de comunicación
- Carga el diseño sobre una FPGA

Las funciones que cumple FIL en una simulación son, la transmisión/recepción de datos entre Simulink o MATLAB y la FPGA en un entorno en tiempo real, el enfoque FIL facilita la exploración de manera eficiente de un algoritmo y la simulación de sistemas.

1.2.5. Registro de corrimiento

Los registros de desplazamiento son circuitos secuenciales digitales, formados por una serie de biestables o flip-flops, generalmente de tipo D conectados en serie o cascada. Según la configuración de las conexiones de los biestables, el desplazamiento se realizará a la izquierda o a la derecha de la información almacenada. La capacidad de almacenamiento de un registro es el número total de bits que puede contener. El funcionamiento se realiza de manera síncrona con la señal del reloj.

Las funciones que cumple dentro de un sistema digital son, servir de almacenamiento temporal de un conjunto de bits sobre los que se está realizando una labor de procesamiento y realizar desplazamiento de datos a lo largo de los flip-flops.

1.2.6. Representaciones tiempo-frecuencia

En el ámbito de las ciencias aplicadas usualmente representamos una señal física mediante una función del tiempo $s(t)$ o, alternativamente, en el dominio de la frecuencia por su TF $\hat{s}(\omega)$. Ambas representaciones son en ciertos sentidos naturales, resultantes de la habitual modalidad de enfocar el universo real. Las mismas contienen exactamente la misma información sobre la señal, respondiendo a enfoques distintos y complementarios. Asumiendo que la señal es aperiódica y de

energía finita, estas representaciones se relacionan mediante el par de fórmulas o par de Fourier ilustrados en (1) y (2) [15].

$$\hat{s} = \int_{-\infty}^{\infty} s(t) e^{-i\,\omega t}\, dt \tag{1}$$

$$\hat{s} = \frac{1}{2\pi} \int_{-\infty}^{\infty} s(\omega) e^{i\,\omega t}\, d\omega \tag{2}$$

Donde t representa el tiempo y ω la frecuencia angular. Por consiguiente, la información en uno de los dominios puede recuperarse a partir de la información desplegada en el otro. Un proceso temporal es la superposición integral de una colección de ondas monocromáticas que oscilan con amplitud constante. Estas ondas, con frecuencias definidas, están representadas por las funciones $\hat{s}\omega(t) = \hat{s}(\omega)e^{\iota}\,\omega t$. Al interferirse entre sí conforman los distintos fenómenos y estructuras, localizadas en el tiempo [15].

Al procesar señales es común encontrar fenómenos oscilantes, casi monocromáticos, localizados en el tiempo. También ondas perdurables, que, con amplitudes casi estacionarias, exhiben patrones de frecuencia variables en el tiempo. Surge entonces, en forma natural, la noción de los fenómenos localizados en tiempo y frecuencia, es decir de objetos que para su descripción requieren información conjunta de ambos dominios [15].

El par de Fourier no es la herramienta adecuada para expresar explícitamente este tipo de información conjunta, dado que las funciones elementales de representación son las ondas estacionarias y monocromáticas. En contraposición, para representar tales fenómenos requerimos de patrones elementales capaces de localizar conjuntamente la información de ambos dominios. Esto plantea el problema de las representaciones tiempo-frecuencia [15].

1.3. Marco conceptual

1.3.1. Tipos de señales.

Una señal es cualquier magnitud que podamos medir de alguna forma y que contiene información sobre el comportamiento o la naturaleza de algún fenómeno. Desde el punto de vista matemático una señal es una función de una o varias variables cuyo dominio de definición puede ser un continuo o una cantidad finita de puntos. La forma de una señal sonora o de una producida por un electrocardiograma cambia con el paso del tiempo, de manera que su amplitud es una función de una sola variable, es decir una señal unidimensional. Una imagen fija es, por el contrario,

una señal bidimensional porque está definida sobre un entramado de dos dimensiones, Figura 2 [16].

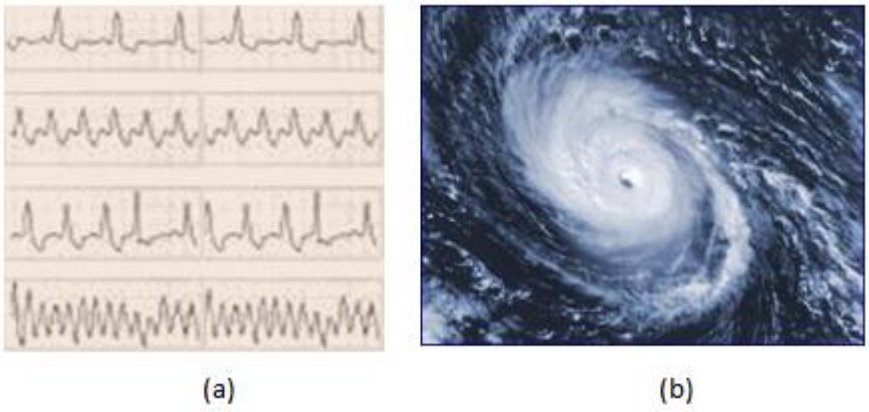


Figura 2. (a) Señal unidimensional proveniente de un electrocardiograma (b) señal bidimensional proveniente de una fotografía.

1.3.2. Pre-procesamiento de señales.

El pre-procesamiento de señales, consiste en descartar la información presente en la señal que no resulta relevante para el reconocimiento de su contenido, como por ejemplo, eliminación del ruido (mediante técnicas clásicas de filtrado, aproximación de funciones, o la TW), eliminación de la interferencia de la red, eliminación de las variaciones de la línea base (utilizando diferentes tipos de filtros), la detección de los puntos significativos de una onda (mediante algoritmos de tratamiento digital de las señales, detectando la primera y segunda derivadas, filtrado digital, transformaciones no lineales, etc.), todo ello para conseguir una señal limpia de interferencias y perfectamente segmentada [17].

1.3.3. Transformada wavelet.

La TW ha demostrado ser una exitosa herramienta para: manejar señales transitorias, comprensión de información, reducción de ancho de banda, análisis de la frecuencia dependiente del tiempo de las señales cortas transitorias, análisis del sonido, y representación de la retina humana [18].

La TW representa una señal en una secuencia de coeficientes de ondas finitas, y puede estar definida por la función wavelet ψ(t) (también llamada wavelet madre (ψ)), y la función de escala φ(t) también conocida como la wavelet padre.

Cuando se refiere a una “wavelet madre”, se está indicando el hecho de que las funciones usadas, derivan de una función principal, es decir, la wavelet madre es el modelo o prototipo, a partir del cual se generan el resto de funciones.

De manera muy general, la TW de una función $f(t)$ es la descomposición de $f(t)$ en un conjunto de funciones $\psi s,\tau(t)$, que forman una base y son llamadas las “Wavelets”, ilustradas en (3) [19].

$$W_f(a,b) = \int f(t)\,\psi_{a,b}(t)dt \tag{3}$$

Las Wavelets son generadas a partir de la traslación y cambio de escala de una misma función wavelet $\psi(t)$ [19]:

1. ***Cambio de escala:*** *Por la que es comprimida o dilatada mediante el parámetro escala “a”.*
2. ***Traslación:*** *Con la que la señal o Wavelet es trasladada por el parámetro traslación “b” a lo largo del eje de tiempos.*

De esta forma, cada familia wavelet está definida mediante la siguiente expresión, mostrada en (4) [19]:

$$\psi_{\mathrm{a,b}}(\mathrm{t}) = \frac{1}{\sqrt{\mathrm{a}}}\psi\left(\frac{\mathrm{t}-\mathrm{b}}{\mathrm{a}}\right), \qquad \mathrm{a} > 0 \tag{4}$$

La TW se puede clasificar en dos grupos de acuerdo a la forma en que los parámetros de desplazamiento y escala son discretizados. Estos dos grupos son conocidos como la TW continúa (TWC) y la TW discreta (TWD).

1.3.3.1. Transformada wavelet discreta

El diseño de una versión discreta de la TW, esencialmente consiste en definir una apropiada red discreta de parámetros $\{(a_j; b_{jk})\}$ ilustrados en (5). [15], de escalas y traslaciones, respectivamente. De modo que la familia de wavelets $\psi_{j,k}$ mostrada en (6) [15], sea admisible. En general, constituye un problema difícil caracterizar en general, aquellas wavelets que definen una Transformada Discreta [15].

$$a_j = 2^{-j};\ b_{jk} = 2^{-j} \quad j,k\ \in Z \tag{5}$$

$$\psi_{j,k}(t) = 2^{\frac{j}{2}}\psi\left(2^j t - k\right), \quad j,k\ \in Z \tag{6}$$

Se cuenta con varias clases de wavelets admisibles. Entre las más difundidas son las wavelets spline, las wavelets de Daubechies y otras análogas, ampliamente difundidas en la literatura y en el software actualmente disponible. Entre estas,

encontramos diversas variantes, y particularmente las que generan bases ortonormales de wavelets, la TW discreta se describe en(7) [15].

$$DTW_{\psi}s(j,k) = < S, \psi_{jk} > = \int_{-\infty}^{\infty} s(t)\psi_{jk}(t)\, dt \qquad (7)$$

Dentro de la TWC y TWD existen sub grupos de TW, conocidas como las wavelets madres que son descriptas a continuación.

1.3.3.2. Clases de wavelet

Las clases de TW son definidas por las propiedades que cada una de ellas poseen, entre las más importantes, se tiene: la función de escala, el número de momentos de desvanecimiento, el soporte compacto, la regularidad y la simetría. Estas propiedades serán descritas a continuación.

En la Tabla 1, se ilustra un resumen de las propiedades asociadas a las wavelets más utilizadas en el ámbito del procesamiento de señales [20].

Propiedades	Haar	Daubechies	Symlets	Morlet	Mexican Hat	Meyer	Gaussiana
Nombre	haar	dbN	symN	morl	mexh	meyr	gausN
Soporte Compacto Ortogonal	*	*	*				
Soporte Compacto Biortogonal							
Análisis Ortogonal	*	*	*			*	
Análisis Biortogonal	*	*	*			*	
DTW	*	*	*				
CTW	*	*	*	*	*	*	*
Momentos de desvanecimiento		*	*				
Simetría	*			*	*	*	*
Antisimetría		*					
Cerca simetría			*				
Función de escala	*	*	*			*	
Regularidad		*	*				
Infinitamente regular				*	*	*	*
Reconstrucción exacta	*	*	*		*	*	*
Algoritmo Rápido	*	*	*				*
Filtros FIR	*	*	*				

Tabla 1. Resumen de las Características y propiedades asociadas a las Wavelets más comunes

Función de escala o wavelet padre φ(t): Es una función que está asociada a las transformaciones wavelet. Definición basada en (8) [19].

$$\varphi_{j,k}(t) = 2^{-\frac{j}{2}}\psi\left(2^{-j}t - K\right), \quad a, b \in Z \qquad (8)$$

Donde *K* varía desde 1 al número de coeficientes en el componente especifico de una descomposición.

La función de escala (φ) se relaciona con las aproximaciones de la descomposición wavelet, es decir, φ realiza aproximaciones de la señal y a pesar de que esta es parte de la definición de la TW, sin embargo, no es una propiedad intrínseca asociada a cualquier wavelet madre.

Momentos de desvanecimiento: Es una propiedad importante para la TW, esta relaciona con la suavidad de la wavelet. El orden de las wavelets es típicamente dado por el número de momentos de desvanecimiento (indica el orden de la transformada). Estos momentos definen la complejidad de la señal de la wavelet, es decir entre más momentos de desvanecimiento, significa, que un conjunto de coeficientes de ondas pueden representar funciones más complejas. También se conoce esta propiedad, como la capacidad de que una wavelet pueda suprimir un polinomio de grado *n* (cuando el valor promedio de una wavelet es cero, se tiene un momento de desvanecimiento, y los polinomios de grado *n* serán eliminados.)
Los momentos de desvanecimiento pueden estar asociados a las funciones de escala o las funciones wavelets madre. Esta es una propiedad útil para propósitos de compresión de información, y que a la vez está relacionada con la eliminación de ruido.

Soporte Compacto: Esta propiedad significa**,** que las funciones wavelets se desvanecen o se anulan fuera de un intervalo finito, es decir, que la wavelet con soporte compacto tiene una duración finita. Esta propiedad le permite a la TW trasladar una función desde el dominio temporal hasta el dominio de frecuencias, de tal manera que posea la localización de ambos dominios.

Regularidad: La regularidad de la wavelet, representa la característica de obtener una señal o imagen reconstruida con suavidad, es decir, es la propiedad que le permite a la wavelet reconstruir fielmente una señal a partir los coeficientes calculados

Simetría: En muchas aplicaciones de filtrado se necesitan filtros con coeficiente de simetría para lograr fase lineal, debido a que los filtros de característica de fase no lineal originarán una distorsión de fase, puesto que las componentes de distinta frecuencia al ser procesadas por el filtro tendrán un retraso que no será proporcional a la frecuencia y por lo tanto se alterará la relación original entre los distintos armónicos que la forman. Por lo tanto, la simetría en el uso de filtros de reconstrucción, significa que la TW tenga fase lineal (La simetría es útil para evitar desfase en el procesamiento de señales). Por ejemplo:

> "*Si la wavelet es simétrica, al verla como un filtro se puede decir que tiene fase lineal, si no es simétrica se introduce distorsión en la fase. Esto es de*

especial interés en aplicaciones de procesamiento de sonido e imágenes" [21].

En síntesis, la importancia de una propiedad sobre otra depende fundamentalmente de las necesidades de la aplicación.

Continuando con las clases de TW, a continuación, se realiza una descripción de las wavelets más conocidas y sus implementaciones más comunes.

Entre las wavelets más conocidas se tiene: Haar, Daubechies, Symlets, Morlet, Mexican Hat, Meyer, Gaussian y Wavelet Dual-Tree Complex**.**

Wavelet Haar.

La Haar es la primera y más sencilla de las wavelet ortogonales propuesta por Alfred Haar en 1909, tiene una estructura simple de onda cuadrada como se observa en la Figura 3, su característica más distintiva radica en el hecho de que se presta fácilmente a cálculos simples, además posee una función de escala asociada [22]. La función wavelet Haar ψ(t) está definida por la siguiente expresión, ilustrada en(9). En la Figura 4, se observa la función de escala φ asociada a la función wavelet Haar.

$$\psi(t) = \begin{cases} 1 & 0 \leq t \leq \frac{1}{2} \\ -1 & \frac{1}{2} < t \leq 1 \\ 0 & t \notin [0,1] \end{cases} \tag{9}$$

Mientras que la función de escala está dada por la expresión (10):

$$\phi(t) = \begin{cases} 1 & t \in [0,1] \\ 0 & t \notin [0,1] \end{cases} \tag{10}$$

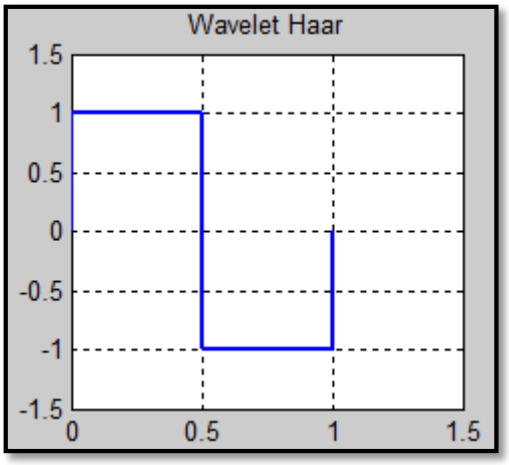


Figura 3. Función wavelet Haar.

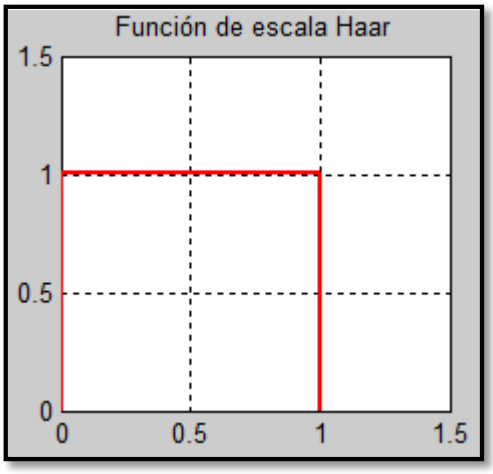


Figura 4. Función de escala asociada a la función wavelet Haar.

La wavelet Haar, que descompone cada señal en dos componentes, uno se llama media (aproximación) o tendencia y el otro es conocido como diferencia (detalle) o fluctuación [22],[23].
La transformada wavelet Haar tiene una serie de ventajas [10], [22]–[24]:

- ✓ Es conceptualmente simple.
- ✓ No requiere a operaciones en punto flotante.
- ✓ Es eficiente en el uso de la memoria, ya que se puede calcular en su lugar sin una matriz temporal.
- ✓ Es la única wavelet ortogonal que es exactamente reversible sin los efectos de borde que son un problema con otras transformadas wavelets.
- ✓ Proporciona alta relación de compresión y alta PSNR (Relación señal a ruido pico).

- ✓ Se mejora el detalle de manera recursiva.
- ✓ Herramienta útil para la compresión de imágenes.
- ✓ Fácilmente representado en software e incluso hardware.

Por otra parte, la Haar tiene sus limitaciones debido a su discontinuidad, por ejemplo, si un gran cambio se lleva a cabo a partir de un valor par a un valor impar, el cambio no se refleja en los coeficientes de alta frecuencia, esto resulta siendo un problema para algunas aplicaciones [24],[25]. Esta limitación o desventaja de la técnica Haar, sin embargo, a la vez es una ventaja para el análisis de imágenes y señales con transiciones repentinas, como la vigilancia de fallas en máquinas eléctricas [26].

La wavelet Haar es buena opción para detectar la información localizada en el tiempo y en bajas frecuencias, posee las propiedades de simetría, admite soporte compacto, aunque cuenta con el soporte más corto entre todas las wavelets ortogonales. No se adapta bien a la aproximación de funciones suaves (no es regular), ya que sólo tiene un momento de fuga o desvanecimiento, puede representar una señal mediante la TWD y TWC [27].

Wavelet Daubechies.

Las Daubechies (Db), propuestas por Ingrid Daubechies, son una familia de wavelets ortogonales ilustradas desde los niveles 2 al 5 en la Figura 5, las cuales definen una transformación discreta y están caracterizados por un número máximo de momentos de desvanecimiento en cierto soporte dado. Con cada tipo de wavelet de esta clase, hay una función de escala asociada que genera un análisis de múltiples resoluciones ortogonales [28]. En la Figura 6, se observa la función de escala db4 asociada a la función wavelet db4.

Las Db se definen de la misma manera que la transformada wavelet Haar, calculando las medias y las diferencias que se ejecutan a través de productos escalares con señales de escala, la única diferencia entre ellas consiste en cómo se definen estas señales de escala y pequeñas ondas. Para las transformadas Db, las señales de escala y ondas tienen soportes ligeramente más largos. Este ligero cambio, sin embargo, ofrece una enorme mejora en las capacidades de estas nuevas transformaciones.

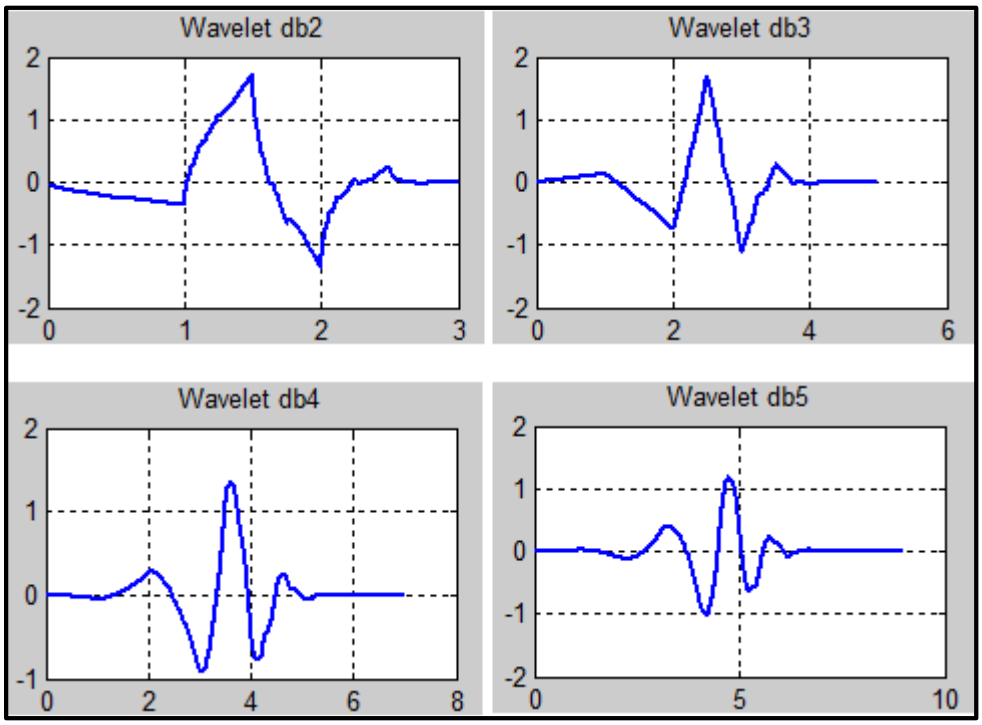


Figura 5. Familia wavelet Daubechies.

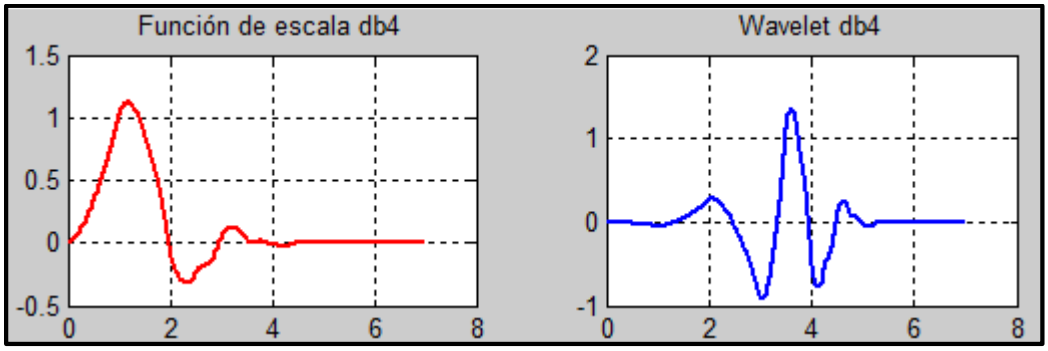


Figura 6. Función de escala (db4) asociada a la función wavelet (db4).

Como ya se había enunciado anteriormente, Db son una amplia familia, conocidos como db1, db2,..., db(n); a pesar de que cada una de ellas son muy similares, la diferencia más obvia entre ellas es la longitud de los soportes de sus señales de escala y de las funciones wavelet. La wavelet Daubechies de orden uno (db1) es la misma que la wavelet Haar. Todas las Db, proporcionan un conjunto de potentes herramientas para la realización de tareas básicas de procesamiento de señal. Estas tareas incluyen la compresión y la eliminación de ruido para señales de audio y de imágenes, e incluyen la mejora de imágenes y reconocimiento de señales.
Las wavelet Db se utilizan en el procesamiento de señales discretas, además, se considera la wavelet madre más utilizada en estudios de sistemas de potencia debido a su propiedad ortogonal, que es potente para la localización y clasificación

de perturbaciones. En comparación con otras ondas ortogonales como Haar y symlet, Db da un mayor rendimiento en términos de complejidad de cálculo y respuesta del filtro [10].

En la Tabla 2 [20], se puntualizan las características de esta familia de wavelets, donde se visualiza por ejemplo si aplica para un análisis ortogonal, biortogonal, si posee soporte compacto, etc.

Wavelet Daubechies	
Creador	Daubechies
Nombre corto	db
Orden N	*N* = 1(Haar), 2, 3, …
Ejemplo	db1(Haar), db4, db15
Ortogonal	✓
Biortogonal	✓
Soporte compacto	✓
TWD	posible
TWC	posible
Ancho de ventana	2N-1
Longitud de filtro	2N
Regularidad	0,2N para N grande
Simetría	Poco simétrica
Momentos de desvanecimiento	N

Tabla 2. Características de las Wavelets Daubechies.

Wavelet Symlets.

Las wavelets Symlets ilustradas en la Figura 7 desde los niveles 2 al 7, son una familia de ondas casi simétricas propuestas por Ingrid Daubechies, como modificación a la familia Db. Las propiedades de estas dos familias de wavelets son muy similares, pero con la única diferencia de que la symlets posee aproximación simétrica. Al igual que en la familia Db, para cada tipo de la familia Symlets, existe una función de escalada asociada, como por ejemplo, para la wavelet sym4 se ilustrada en la Figura 8 la función de escala. Las Symlets, son utilizadas en aplicaciones como la eliminación de ruido, clasificación y extracción de características de señales o de imágenes, entre otras. En la Tabla 3 [20], se detallan las características de esta familia de wavelets.

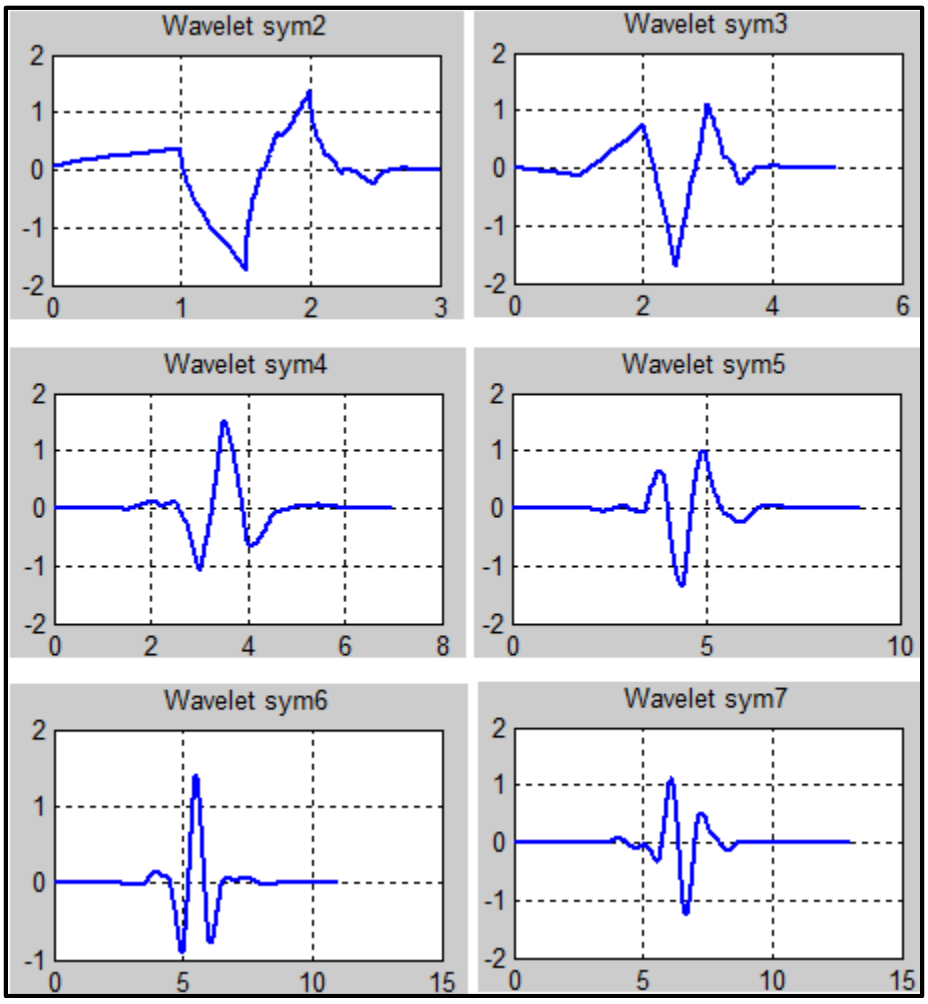


Figura 7. Familia wavelet Symlets.

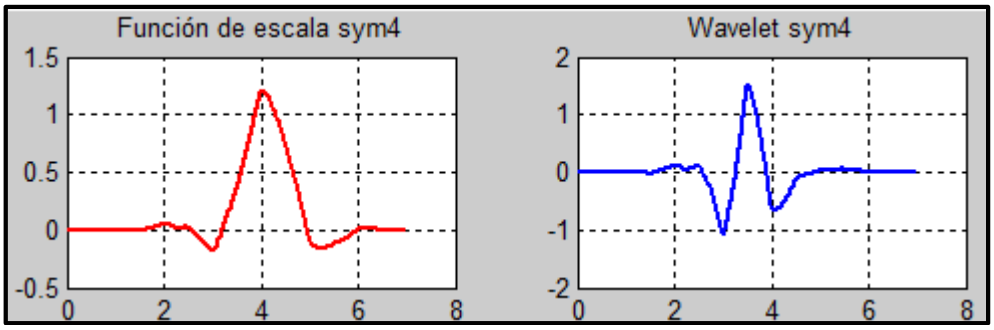


Figura 8. Función de escala (sym4) asociada a la función wavelet (sym4).

Wavelet Symlet	
Creador	Daubechies
Nombre corto	sym
Orden N	*N* = 2, 3, …
Ejemplo	Sym2, sym8
Ortogonal	✓
Biortogonal	✓
Soporte compacto	✓
TWD	Posible
TWC	Posible
Ancho de ventana	2*N* - 1
Longitud de filtro	2*N*
Simetría	Casi simétrica
Momentos de desvanecimiento	*N*

Tabla 3. Características de las Wavelets Symlets

Wavelet Morlet.

La wavelet Morlet está definida como una función exponencial compleja (transformada de Fourier) ilustrada en (11) [20], o como una onda senoidal compleja que contiene una envolvente gaussiana, propuesta por el ingeniero Jean Morlet y el Físico Alex Grossman a principio de la década de los 80. Es una transformada simétrica, utilizada para realizar la TWC (Wavelet Morlet ejemplo clásico de una wavelet de tiempo continuo), esta wavelet no tiene función de escala. La función wavelet se visualiza en la Figura 9, mientras que las características se ilustran en la Tabla 4 [20].

$$\psi(t) = Ce^{\left(-\frac{t^2}{2}\right)}\cos(5t) \tag{11}$$

Donde *C* es una constante usada para normalización en vista de la reconstrucción. La wavelet Morlet es usada en muchas aplicaciones de procesamiento de señales, pero su uso más común, es en el análisis de frecuencia y tiempo de las señales acústicas.

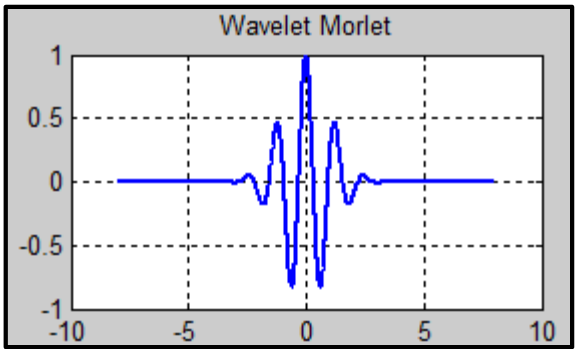


Figura 9. Función wavelet Morlet.

Wavelet Morlet	
Creador	J. Morlet y A. Grossman
Nombre corto	morl
Ortogonal	×
Biortogonal	×
Soporte Compacto	×
TWD	×
TWC	Posible
Ancho de ventana	Infinito
Soporte efectivo	[-4, 4]
Simetría	✓

Tabla 4. Características de la Wavelet Morlet.

Wavelet Mexican Hat

La Wavelet Mexican Hat o Sombrero mexicano, se define como la segunda derivada de la función de distribución Gaussiana, puntualizada en (12) [20]. Al igual que la wavelet Morlet, es simétrica y no tiene función de escala asociada. La función wavelet se ilustra en la Figura 10, y sus características en la Tabla 5 [20].

$$\psi(t) = \frac{2}{\sqrt{3}\,\pi^{\frac{1}{4}}}\, e^{-\frac{t^2}{2}}(1 - t^2) \tag{12}$$

La TW Mexican Hat, se ha utilizado ampliamente en la detección de borde de una imagen.

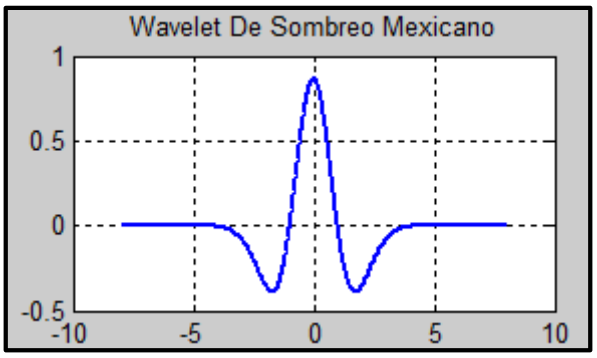


Figura 10. Función wavelet Mexican Hat.

Wavelet Mexican Hat	
Nombre corto	mexh
Ortogonal	×
Biortogonal	×
Soporte Compacto	×
TWD	×
TWC	Posible
Ancho de ventana	Infinito
Soporte efectivo	[-5, 5]
Simetría	✓

Tabla 5. Características de la Wavelet Mexican Hat.

Wavelet Meyer

La wavelet Meyer propuesta por Yves Meyer en 1985, es una wavelet ortogonal infinitamente diferenciable con apoyo finito, contiene las propiedades de simetría, suavidad, función de escala, no tiene soporte compacto. La función de wavelet se ilustra en la Figura 11, mientras que la función de escala asociada a la wavelet se ilustra en la Figura 12, amabas se definen en el dominio de la frecuencia [29]. La función wavelet y escala están definidas por (13) y (14) respectivamente [20].

$$\hat{\psi}(\omega) = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{i\frac{\omega}{2}} sen\left(\frac{\pi}{2} v\left(\frac{3}{2\pi}|\omega| - 1\right)\right) & \frac{2\pi}{3} \leq |\omega| \leq \frac{4\pi}{3} \\ \frac{1}{\sqrt{2\pi}} e^{i\frac{\omega}{2}} cos\left(\frac{\pi}{2} v\left(\frac{3}{4\pi}|\omega| - 1\right)\right) & \frac{4\pi}{3} \leq |\omega| \leq \frac{8\pi}{3} \\ 0 & |\omega| \notin \left[\frac{2\pi}{3}, \frac{8\pi}{3}\right] \end{cases} \quad (13)$$

$$\hat{\phi}(\omega) = \begin{cases} \frac{1}{\sqrt{2\pi}} & |\omega| \leq \frac{2\pi}{3} \\ \frac{1}{\sqrt{2\pi}} cos\left(\frac{\pi}{2} v\left(\frac{3}{2\pi}|\omega| - 1\right)\right) & \frac{2\pi}{3} \leq |\omega| \leq \frac{4\pi}{3} \\ 0 & |\omega| > \frac{4\pi}{3} \end{cases} \quad (14)$$

Donde *v* es una función auxiliar, definida como se muestra en (15).

$$v = a^4(35 - 84a + 70a^2 - 20a^3) \qquad a \in [0,1] \quad (15)$$

A pesar que la wavelet Meyer no tiene soporte compacto, existe una buena aproximación que conlleva a los filtros FIR (Finite Impulse Response), que permite el uso de esta wavelet en la TWD. En la Tabla 6 [20], se detallan las propiedades que posee esta wavelet. Eventualmente la wavelet Meyer, es usada para la eliminación de ruido y diagnóstico de fallas para motores DC.

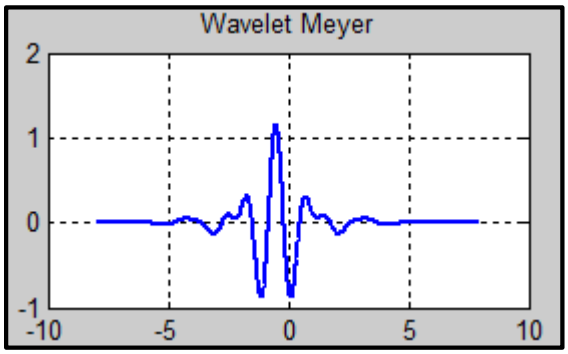


Figura 11. Función wavelet Meyer.

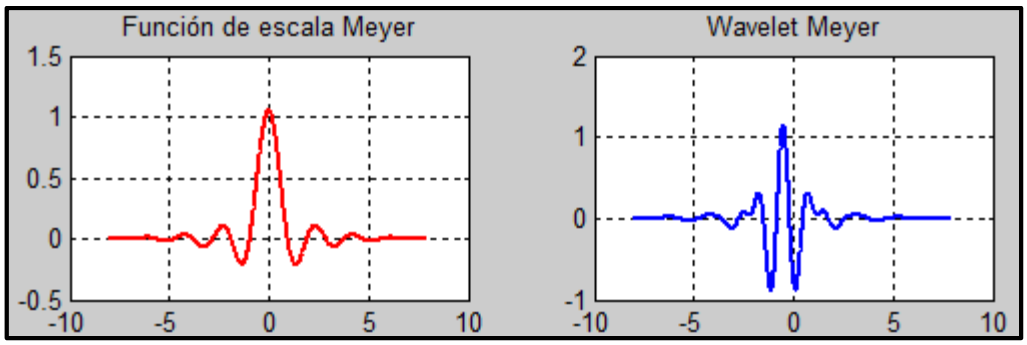


Figura 12. Función de escala asociada a la función wavelet Meyer.

Wavelet Meyer	
Creador	Y. Meyer
Nombre corto	meyr
Ortogonal	✓
Biortogonal	✓
Soporte Compacto	×
TWD	Posible, pero sin FTW basado en Aproximaciones mediante FIR
TWC	Posible
Ancho de ventana	Infinito
Soporte efectivo	[-8, 8]

Tabla 6. Características de la Wavelet Meyer.

Wavelet Gaussiana

La familia wavelet Gaussiana, se define como la primera derivada de la función de densidad de probabilidad gaussiana ilustrada en (16); si se deriva a un orden superior, se tiene la segunda derivada gaussiana, la cual, como se enuncio anteriormente define a la TW Mexican Hat. La familia wavelet gaussiana se ilustra en la Figura 13, desde el nivel 2 al 5. Sus propiedades se detallan en la Tabla 7 [20]. Esta wavelet es utilizada en la comprensión de imágenes y eliminación de ruido.

$$\psi(t)\frac{2}{\sqrt{3}\pi^{1/4}}e^{\left(\frac{-t^2}{2}\right)\left(1-t^2\right)} \tag{16}$$

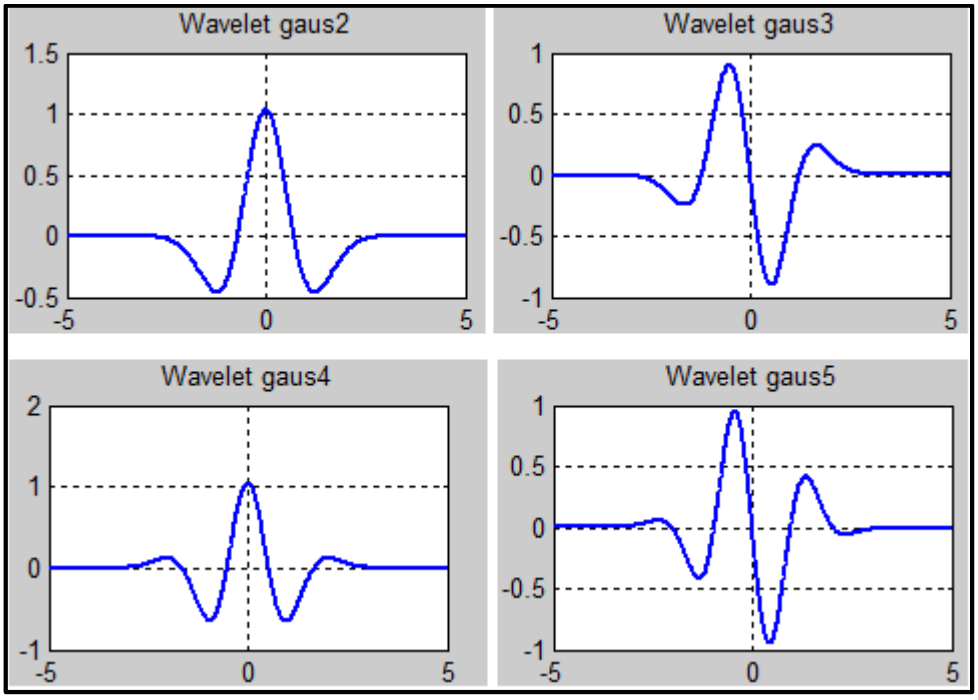


Figura 13. Familia wavelet Gaussian.

Wavelet Gaussiana	
Nombre corto	gaus
Ortogonal	×
Biortogonal	×
Soporte Compacto	×
TWD	×
TWC	Posible
Ancho de ventana	Infinito
Soporte efectivo	[-5, 5]
Simetría	✓ *n* par → Simétrica *n* impar → Antisimétrica

Tabla 7. Características de la Wavelet Gaussiana.

Wavelet Dual-Tree Complex

La TW compleja de doble árbol (TWC-DA) desarrollado por N. G. Kingsbury en 1998, corresponde a una mejora de la TWD, con importantes propiedades adicionales. El TWC-DA ha recibido recientemente un gran interés en el procesamiento de señales, debido principalmente a su dirección selectiva y cerca de desplazamiento de propiedades invariantes.

Definida como una forma de transformada wavelet discreta que genera coeficientes complejos mediante el uso de un árbol doble de filtros de ondas pequeñas para obtener partes reales e imaginarias, es decir, el TWC-DA utiliza dos árboles separados que forman pares de filtros de Hilbert que conducen a coeficientes reales e imaginarios complejos en cada sub-banda (Cada árbol contiene filtros puramente reales, pero los dos árboles producen las partes real e imaginaria, respectivamente, de cada coeficiente wavelet compleja).

Como se enuncio anteriormente esta transformación, surge con el fin de proporcionar una mejor resolución direccional (filtros direccionalmente selectivos) en dos o más dimensiones y la invariancia de cambio aproximada en comparación con la TWD convencional [30]–[32].

El sistema TWC-DA es una potente herramienta para la eliminación de ruido de imagen y vídeo debido a su desplazamiento y rotación alrededor de la invariancia, además es particularmente adecuada para otras señales multidimensionales [30]. La invariancia cerca de desplazamiento y la mejora de la selectividad direccional han facilitado excelentes resultados en la eliminación de ruido, la fusión y otras aplicaciones de procesamiento de imágenes [33].

Características del TWC-DA [32]:

- Invariancia de cambio aproximada.
- Filtrado direccionalmente selectivo en 2 o más dimensiones.
- Factor de redundancia de tan solo $2^m:1$ para señales m-dimensiones.
- Reconstrucción perfecta con filtros de fase lineal cortos.
- Redundancia limitada 2:1 en 1-D, 4:1 en 2-D etc.

La TWC-DA proporciona análisis y reconstrucción de forma óptima para diferentes aplicaciones, como son eliminación de ruido, registro, análisis de textura, clasificación, marca de agua de las imágenes, comprensión etc.

1.3.4. Métodos de umbralización

Los métodos de umbralización, son técnicas que permiten reconocer el ruido en los coeficientes wavelet. El objetivo de la umbralización, consiste en disminuir el ruido de una señal utilizando herramientas estadísticas para determinar los valores que deben ser ruido de una señal para su posterior eliminación.

1.3.4.1. Eliminación de ruido

Para la reducción o eliminación de ruido mediante los métodos de umbralización, es necesario de tres pasos básicos:

- Descomponer: Se elige el tipo de wavelet madre y se selecciona un nivel N de descomposición. En este primer paso se calcula la descomposición wavelet de la señal hasta un nivel N.
- Umbralización: Se realiza la valoración del umbral y el umbral es aplicado a los coeficientes, es decir, aquí se selecciona un umbral y para cada nivel de 1 a N se aplica el umbral a los coeficientes.
- Reconstruir: Reconstrucción wavelet utilizando los coeficientes de aproximación originales de nivel N y los coeficientes de detalle modificadas de los niveles de 1 a N.

1.3.4.2. Función De Umbralización

Existen varios métodos de fijación de umbrales que son utilizados para el propósito de eliminación de ruido de una señal contaminada. Entre estos métodos están la umbralización dura y suave.

El método de umbralización se basa en el descarte de los coeficientes que poseen ruido con respecto a un umbral, debido, que la mayor parte del ruido comúnmente en el proceso de la Transformada Wavelet (TW) se manifiesta o tiende a ser representado por los coeficientes [34],[35].

En el proceso, cada coeficiente es tratado mediante la comparación de umbral contra umbral, es decir, si el coeficiente es menor que el umbral, este coeficiente se establece en cero, de lo contrario, el coeficiente se mantiene o se modifica. Como finalidad del método radica en sustituir los coeficientes con ruido por ceros, y en donde además, en la recuperación de la señal, la TW inversa en el resultado permite conducir a la reconstrucción con las características de la señal esenciales [36].

El método de fijación de umbral es sencillo y eficaz para la reducción de ruido. El Rendimiento de umbral depende del tipo de método de umbral y la técnica (regla) de umbral utilizada para la aplicación requerida. La umbralización dura y suave, se definen a continuación:

1.3.4.2.1. Umbralización Dura.

La umbralización dura consiste en establecer en cero los coeficientes cuyos valores absolutos son menores que el umbral como se representa en(17) [37].

$$f_h = \begin{cases} C(t), & para\ |C(t)| \geq T \\ 0, & En\ otro\ caso \end{cases} \tag{17}$$

Donde $C(t)$ representa los coeficientes de ondas con ruido y T el valor de la umbralización. La señal del umbral es igual a los coeficientes si estos son mayores que el umbral, es decir, los coeficientes de umbral duro menor que el umbral T se ponen en cero, mientras que otros coeficientes se mantienen sin cambios.

1.3.4.2.2. Umbralización Suave

La umbralización suave es una extensión de la umbralización dura, donde se ponen en cero los coeficientes cuyos valores absolutos son menores que el umbral, y además resta el umbral a los coeficientes mayores que el umbral como se representa en(18) [37].

$$f_s = \begin{cases} sgn\big(C(t)\big)(C(t) - T), & para\ |C(t)| \geq T \\ 0, & En\ otro\ caso \end{cases} \tag{18}$$

Donde $sgn()$, es una función signo que devuelve 1 si el coeficiente es mayor que cero, devuelve 0 si es igual a cero y -1 si es menor que cero [37].

La eliminación de ruido tiene una amplia gama de aplicaciones en el procesamiento de señales, como también en otros campos. Estos métodos de umbralizacion permiten la eliminación o reducción de ruido de señales unidimensionales, bidimensionales y tridimensionales, haciendo uso de técnicas como Sqtwolog, Rigrsure, Heursure y Minimaxi.

1.3.4.3. Técnicas de Umbralización

Cuatro reglas muy utilizadas para la selección de umbral se describen a continuación:

Sqtwolog

Es un tipo de método de umbralización propuesto por Donoho y Johnstone. Los valores de umbral se calculan por el método universal de umbral dada por (19) [2],[3].

$$th_j = \sigma_j \sqrt{2\ln(N_j)} \tag{19}$$

Donde σ es la desviación media absoluta (MAD) y N_j es la longitud de la señal ruidosa. La MAD se expresa en (20) [2],[3].

$$\sigma_j = \frac{MAD_j}{0.7980} = \frac{mean(|x - \bar{x}|)}{0.7980} \tag{20}$$

En donde $\bar{x}$ representa la media aritmética, y x el coeficiente wavelet a escala j.

Rigrsure

Es un método umbral suave evaluador de riesgo imparcial. Su fórmula se expresa en(21).

$$th_j = \sigma_j \sqrt{\omega_b} \tag{21}$$

Donde ω_b es el b coeficiente wavelet al cuadrado (coeficiente en riesgo mínimo) elegido a partir del vector W = $[\omega_1, \omega_2, \dots, \omega_N]$. Este vector contiene los valores de los coeficientes wavelet al cuadrado organizados de menor a mayor y σ es la deviación estándar de la señal ruidosa [38], [39].

Heursure

Es una combinación de los métodos Sqtwolog y Rigrsure. Si la relación señal a ruido es muy pequeña, la estimación del método SURE es ruidosa. En tal caso se utiliza un umbral Sqtwolog [38], [39].

Minimax

Utiliza un umbral fijo para producir un rendimiento minimax para el error cuadrado medio contra un procedimiento ideal. El principio de minimax se utiliza en estadística con el propósito de diseñar estimadores. El umbral esta dado en (22) [38], [39].

$$\lambda = \begin{cases} \sigma(0.3936 + 0.1829log_2 N), & N > 32 \\ 0 & N < 32 \end{cases} \quad (22)$$

Donde $\sigma = median\left(\frac{|\omega|}{0.6745}\right)$ y ω es el vector de coeficientes wavelet a escala unitaria y N es la longitud del vector de la señal.

1.4. Bitácora de investigación

10 artículos en conjunto con los antecedentes se han analizado con detenimiento para obtener una lista que permita brindar el mejor tipo de wavelet útil en este proyecto y el tipo de sistema en capacidad de realizar el procesamiento de una señal de una dimensión.

Año de publicación	**Autores**	**Sistema embebido**	**Aplicación**	**Tipo de Transformada Wavelet**
2009	Mukul Shirvaikar, Tariq Bushnaq	TMS320C6416 DSP Starter Kit and ALTERA EP2C35F672 C6	Comparación entre las plataformas DSP y FPGA, para el procesamiento de imágenes en tiempo real.	Transformada Wavelet 5/3 2-D
2011	Qijun Huang, Yajuan Wang and Sheng Chang	DE II, ALTERA EP2C35F672C6	Procesamiento de imágenes	Daubechies 8
2012	Md. Rezwanul Ahsan, Muhammad Ibn Ibrahimy	ALTERA Stratix III EP3SE50F780I4 L	Procesamiento de señales en tiempo real.	Daubechies 4
2013	Haider Ismael Shahadi, Razali Jidin	Xilinx Spartan6-SP601 evaluation board	Procesamiento de señales de audio	Transformada Wavelet Haar entera
2013	Tzu-Hao Yen, Chung-Yu Chang	MSP430FG4618 conectado via bluetooth a un	Sistema de monitorio y reconocimiento	TWD de 5 niveles

		telefono con la plataforma android.	en tiempo real de señales del ECG.	
2014	Leji C Koshy, A Sanjeevi Gandhi	SPARTAN 3E development kit	Eliminación de ruido de alta frecuencial en tiempo real.	Daubechies 5
2015	Saurabh Acharya, Hitesh Kabra	Spartan FPGA family	Descomposición de señales del ECG.	Transformada Wavelet Haar entera
2015	Ferhat Canbay, Vecdi Emre Levent	Spartan-6 board	Procesamiento de señales embolicas	Transformada Wavelet de Doble Árbol Compleja (TWDAC) de 5 niveles.
2015	R.Sindhujaa , K.Kavin	XILINX Virtex-5 FPGA Kit	Remover el ruido de fondo al hablar.	Transformada Wavelet Estacionaria y Paquete de Transformadas Wavelet
2015	Saleha Khatun, Ruhi Mahajan	Windows 7 OS computer system with (processor Intel Core i5, 3.33 GHz), 4 GB of RAM	Eliminar ruido de un canal del EEG.	sym3, haar, coif3, and bior4.4

Tabla 8. Tipo de wavelet y dispositivo en el que distintos autores han implementado la TW

Mediante la Tabla 8, se presenta un resumen de la investigación, donde se destaca el tipo de TW y el sistema embebido. En esta tabla, distintos autores han embebido la TW en diversos dispositivos, se ha visualizado un especial interés en el dispositivo FPGA, ya que este permite opciones de portabilidad, reducción de costos y escalabilidad en términos de hacer posibles cambios a futuro en el filtro. Por otra parte, en la tabla, se observa una TW que puede representar sus operaciones de manera entera en comparación con las demás familias wavelet. De acuerdo y basados en estos artículos, por la información proporcionada, se ha optado por tomar la decisión de utilizar la clase wavelet Haar e implementarla en una FPGA.

La TW Haar es la única que no requiere de operaciones de punto flotante e incluye ventajas de ejecución rápida y eficiente uso de la memoria, también es una transformación que es perfectamente reversible sin el problema del efecto del borde

a diferencia de otras transformaciones, y es conveniente para las señales de baja velocidad como señales biomédicas.

2. PROPUESTA Y MODELO TEÓRICO

Considerando los antecedentes e investigaciones realizadas, como se observa en la sección 1.4, se ha optado por implementar la TW Haar mediante el método de lifting en una FPGA. Las pruebas se realizaron en una FPGA Altera DE0-Nano, sin embargo, para asegurar la fácil portabilidad entre distintos dispositivos, el lenguaje en que se describe el comportamiento de la TW se ha definido como VHDL.

2.1. Métodos para implementar la TW

Existen diversos métodos para la representación de la TW entre ellos se analizan los más reconocidos, el método convolucional y el esquema lifting; cada método posee un procedimiento distinto para poder encontrar los coeficientes detallados y aproximados, sin embargo, el resultado de ambos métodos es el mismo y son ideales para la aplicación de eliminar el ruido de una señal a través de umbrales.

2.1.1. Método Convolucional

El método por convolución consta de filtrar una señal de entrada de forma independiente a través de filtros de descomposición pasa altas y pasa bajas. Las salidas provenientes de los filtros son sub-muestreadas por dos, es decir, se selecciona a partir de cada dos muestras para producir coeficientes de aproximación A(n) y coeficientes de detalles D(n) como se muestra en la Figura 14. Los filtros son conocidos como bancos de filtros de descomposición.

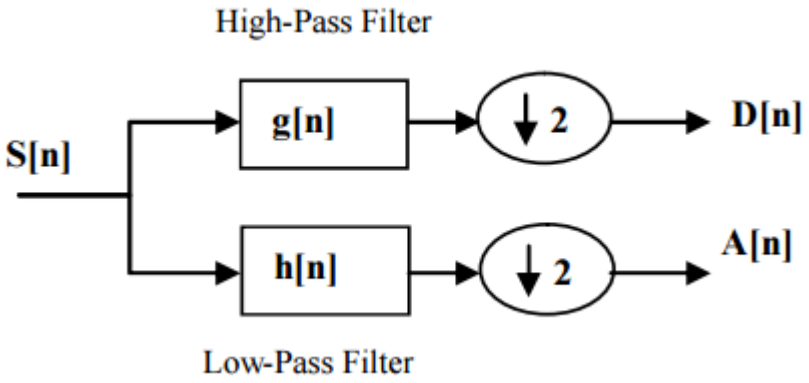


Figura 14. Estructura método de convolución.

La señal original puede ser reconstruida por una interpolación, la cual consiste en insertar ceros entre cada muestra proveniente del sub-muestreo, esta salida pasa a los filtros de reconstrucción pasa altas y pasa bajas respectivamente. Posteriormente, las salidas de los filtros se suman para obtener la señal reconstruida, Figura 15.

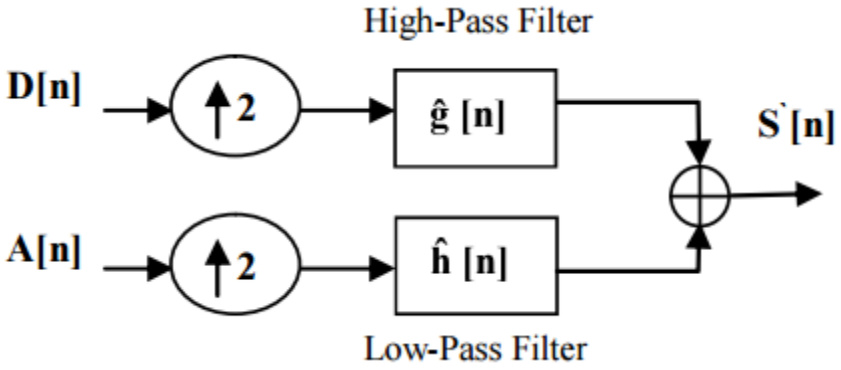


Figura 15. Estructura de reconstrucción método convolucional.

A continuación, se realiza un ejemplo de descomposición mediante este método. Se tiene una señal de entrada de seis valores ilustrada en (23). Los coeficientes de los filtros de descomposición para la TW Haar (24) y los coeficientes de reconstrucción (25)

$$x(n) = [1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6] \tag{23}$$

$$G = \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right], \quad H = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right] \tag{24}$$

$$\hat{G} = \left[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right], \quad \hat{H} = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right] \tag{25}$$

Primer paso, se realiza la convolución entre la señal de entrada y los filtros de descomposición pasa altas y pasa bajas como se observa en la Tabla 9 y Tabla 10. La señal resultante de la convolución tanto para los coeficientes de detalles y aproximación son submuestreados por 2.

Señal $x(n)$	$[1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6]$
Coeficiente del filtro paso alto	$G = \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$
Señal resultante de la convolución	$[-0.7071 \;\; -0.7071 \;\; -0.7071 \;\; -0.7071 \;\; -0.7071 \;\; -0.7071 \;\; 4.2426]$
Coeficientes de detalles	$[-0.7071 \;\; -0.7071 \;\; -0.7071]$

Tabla 9. Calculo de los coeficientes detallados por el método convolucional.

Señal $x(n)$	$[1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6]$
Coeficiente del filtro paso bajo	$H = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$
Señal resultante de la convolución	$[0.7071 \quad 2.1213 \quad 3.5355 \quad 4.9497 \quad 6.3640 \quad 7.7782 \quad 4.2426]$
Coeficientes de aproximación	$[\ 2.1213 \quad 4.9497 \quad 7.7782]$

Tabla 10. Calculo de los coeficientes aproximados por el método convolucional.

Para la reconstrucción se sigue un proceso similar que en la descomposición, este consiste en utilizar la convolución entre los coeficientes detallados, aproximados y los respectivos coeficientes de los filtros de reconstrucción, para regenerar componentes cuya suma represente a la señal original, como se aprecia en la Tabla 11 y Tabla 12.

Coeficientes detallados	$[-0.7071 \quad -0.7071 \quad -0.7071 \quad]$
Coeficientes detallados interpolados	$[-0.7071 \ \ 0 \ \ -0.7071 \ 0 \ \ -0.7071]$
Coeficiente del filtro paso alto de reconstrucción	$\hat{G} = \left[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right]$
Señal resultante de la convolución	$[-0.5 \quad 0.5 \quad -0.5 \quad 0.5 \quad -0.5 \quad 0.5]$

Tabla 11. Calculo de la reconstrucción de la parte impar.

Coeficientes de aproximación	$[\ 2.1213 \quad 4.9497 \quad 7.7782]$
Coeficientes aproximados interpolados	$[\ 2.1213 \quad 0 \ \ 4.9497 \ \ 0 \ \ 7.7782 \ ]$
Coeficiente del filtro paso bajo de reconstrucción	$\hat{H} = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$
Señal resultante de la convolución	$[1.5 \quad 1.5 \quad 3.5 \quad 3.5 \quad 5.5 \quad 5.5]$

Tabla 12. Calculo de la reconstrucción de la parte par.

Finalmente se suman los componentes pares e impares de la señal, es decir se suma la secuencia
$[1.5 \quad 1.5 \quad 3.5 \quad 3.5 \quad 5.5 \quad 5.5]$ con $[-0.5 \quad 0.5 \quad -0.5 \quad 0.5 \quad -0.5 \quad 0.5]$ para recuperar la señal original $[1\ 2\ 3\ 4\ 5\ 6]$.

2.1.2. Método Lifting

El método lifting es un método rápido y eficaz para la representación de coeficientes aproximados y detallados. La estructura de este método se ilustra en la Figura 16 [40], el cual consta de tres pasos: El primer paso consiste en dividir o separar (Split), el segundo en la predicción (P) y la tercera actualización (U).

Split (S): Divide los valores de la señal de entrada, en dos subconjuntos; un subconjunto de muestras pares y otro de muestras impares.

Predict (P): Se utilizan los valores del conjunto de muestras pares para predecir los valores impares.

Update (U): Este paso consiste en actualizar los valores de las muestras pares basados en las muestras impares.

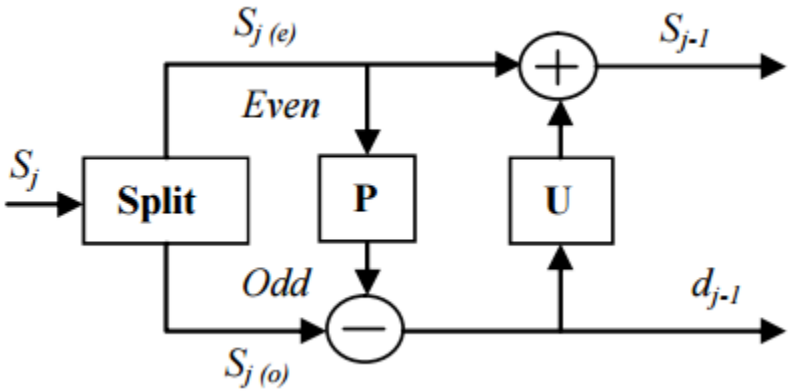


Figura 16. Estructura método lifting.

Para obtener la reconstrucción de la señal se realizan los pasos inversos como se muestra en la Figura 17 [40]. Primero se crean las muestras pares a partir de la diferencia de los coeficientes de detalles y de aproximación. Para obtener el subconjunto de muestras impares se realiza la suma entre las muestras pares y los coeficientes de detalles.

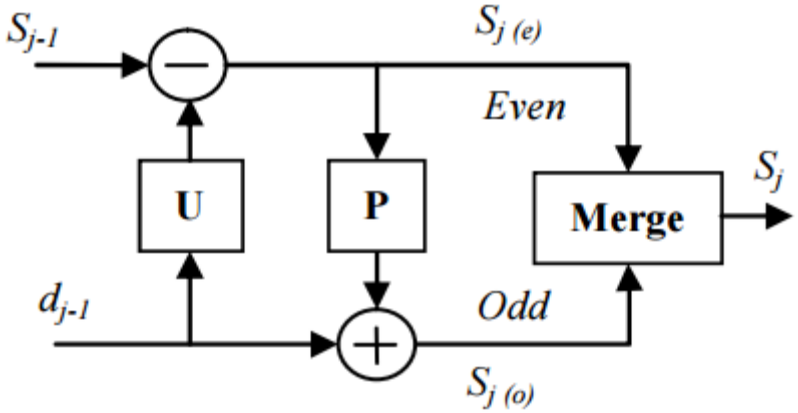


Figura 17. Estructura de reconstrucción método lifting

La función merge o mezcla en la Figura 17, consiste en interpolar el conjunto de señales pares intercalándola con ceros, posteriormente, donde se encuentran los

ceros se colocan en orden los valores de las señales impares como se representa en la Tabla 13.

Secuencia parte par	[1 3 5]
Secuencia parte impar	[2 4 6]
Secuencia par interpolado	[1 0 3 0 5 0]
Resultado de merge	[1 2 3 4 5 6]

Tabla 13. Funcionamiento de la función merge.

El significado de los pasos de predict y update, dependen de la familia wavelet que se esté utilizando. Mediante la transformada wavelet haar en su esquema lifting se pueden obtener, los coeficientes detallados y aproximados mediante (26) y (27) [40] respectivamente

$$Cd = Par - Impar \tag{26}$$

$$Ca = Par + \frac{Par - Impar}{2} \tag{27}$$

La TW Haar inversa consiste en revertir los pasos predict y update para recuperar los componentes pares (28) y los componentes impares (29) para finalmente reemplazar la función split por merge lo cual permite reconstruir la señal original.

$$Par = Cd - Ca \tag{28}$$

$$Impar = Ca + \frac{Cd - Ca}{2} \tag{29}$$

La sencillez matemática del método lifting se aprovecharía, si podemos discernir de ante mano que valores de los coeficientes contienen ruido, sin embargo, debido a que se ha decidido implementar un proceso de eliminación de ruido adaptable, es necesaria una etapa que defina el valor en que los coeficientes tienen ruido para eliminarlo con facilidad, este paso es conocido como la etapa de selección de umbral.

2.2. Eliminación de ruido mediante la TW

El filtrado mediante la transformada wavelet es un proceso de eliminación de ruido a partir de la selección de bandas de frecuencias, existen diversos métodos para llevar a cabo este tipo de procedimiento, cabe señalar, en este documento se hace énfasis en el siguiente:

1. Cargar la señal.

2. Realizar la descomposición de la señal mediante la TW seleccionada hasta un nivel N.
3. Determinar un umbral para referenciar los coeficientes.
4. Regenerar la señal usando los umbrales modificados.

Descomponer: En general descomponer es elegir una wavelet, un nivel N, y procesar la desintegración de la señal a un nivel N.

Nivel de umbral de referencia para los coeficientes: Para cada nivel de 1 a N, o para todos los niveles seleccionar un valor de umbral y aplicar un suavizado para referenciar los coeficientes.

Regenerar: Procesar la reconstrucción de la señal es utilizar los coeficientes modificados del nivel 1 a N para obtener una señal que represente a la original.

Para representar estos tres pasos se utiliza una señal del EEG muestreada a una frecuencia de 100Hz obtenida de la BCI Competition II Figura 18I[41], a la cual se le ha agregado ruido gaussiano de -10dB, Figura 19.

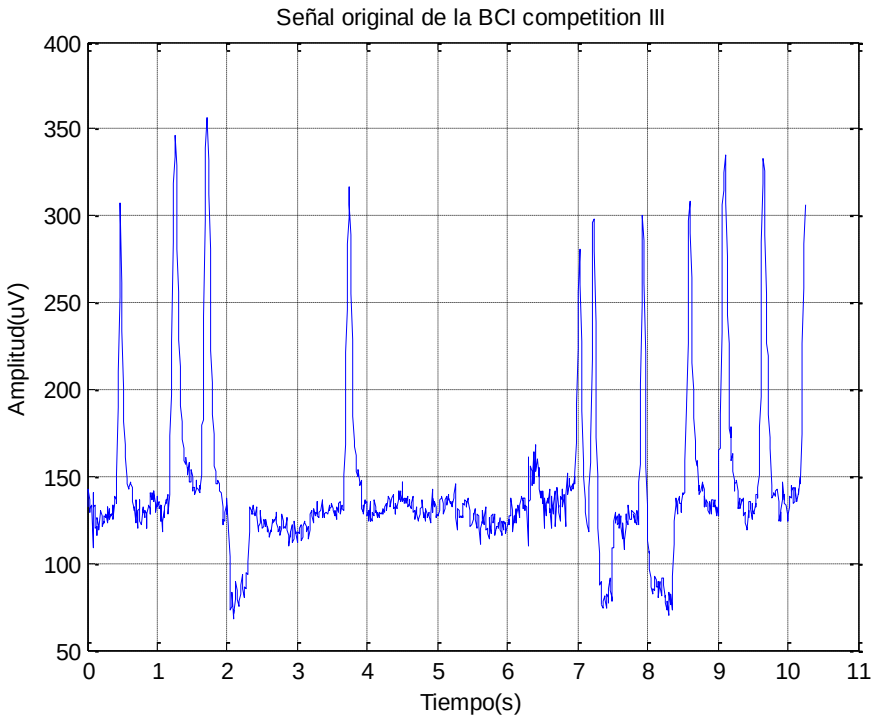


Figura 18. Señal del EEG muestreada a 100Hz

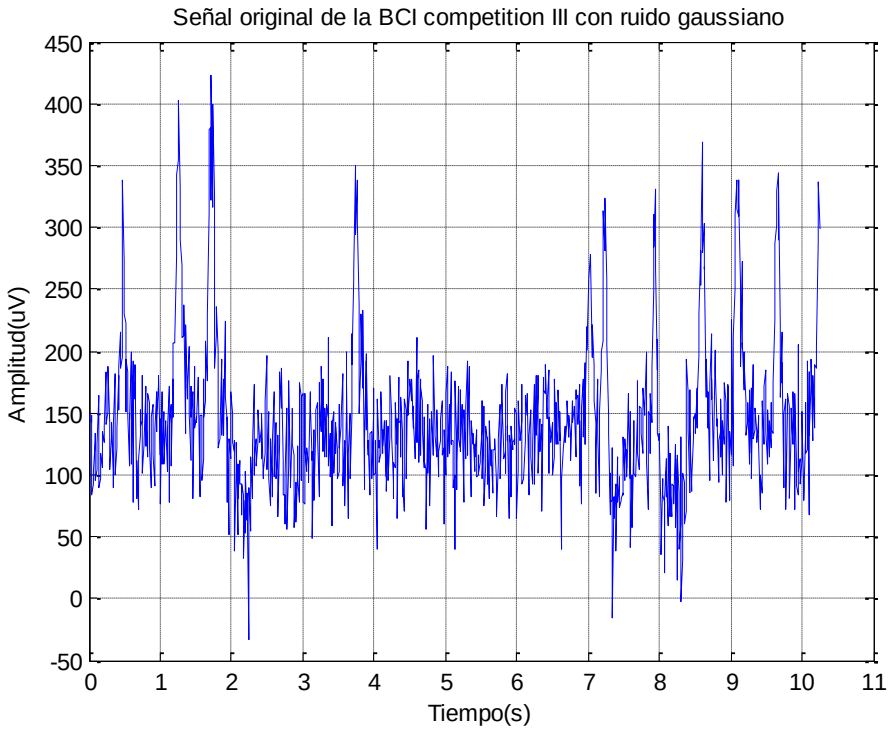


Figura 19. Señal del EEG muestreada a 100Hz con ruido gaussiano de -10dB

2.2.1. Descomponer

Para descomponer una señal mediante la TW es necesario seleccionar una familia wavelet y un nivel de descomposición.

Selección del tipo de wavelet

Cada tipo de wavelet [42] tiene sus peculiaridades que permiten que predominen frente a otras familias para aplicaciones específicas en el mundo del procesamiento digital de señales, en este proyecto se implementa la familia Haar debido a lo expuesto en la sección 1.4, posee ventajas en su implementación, como un rápido y eficiente manejo de la memoria, perfecta reversibilidad y buena referencia en el análisis de SB [43].

Selección del nivel N

La selección del nivel N significa hasta cuantos niveles se realizara la descomposición de la señal mediante los coeficientes de los filtros.

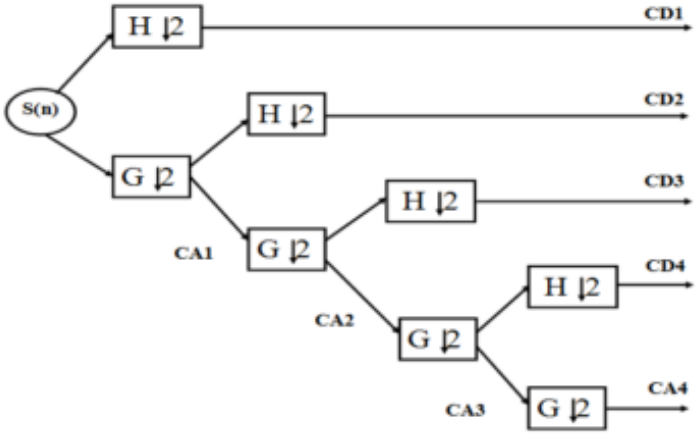


Figura 20. Esquema de descomposición multi resolución.

La separación en niveles consiste en un sub muestreo sucesivo en potencias de base 2 Figura 20 [43], se debe tener en cuenta que la máxima cantidad de niveles está limitada por el número de muestras dada la relación (30) [44]:

$$Maxniveles = log_2 Numerodemuestras \quad (30)$$

Al tener un total de 1024 muestras en la señal, se permite un máximo de 10 niveles, los coeficientes que salen de los filtros pasa-altas son conocidos como detallados, de la misma manera, los coeficientes obtenidos por los filtros pasa-bajos son conocidos como coeficientes aproximados. En la Figura 20, se observa un esquema de descomposición donde s(n) representa la señal original, H representa un filtro pasa alto, G un filtro pasa bajo (los valores de H y G dependen del tipo de wavelet implementada), ↓ 2 es un sub muestro que divide en 2 la frecuencia actual de la señal, CAj son los coeficientes aproximados y CDj son los coeficientes detallados.

La señal tiene una frecuencia de muestreo de 100Hz y por el criterio de Shannon la maxima frecuencia en la que se encuentra información util es de 50Hz, el primer nivel contiene las frecuencias entre 25 – 50 Hz, el nivel 2 contiene las frecuencias entre 12.5-25 Hz, hasta completar la Tabla 14.

Nivel	Rango de frecuencias (Hz) Min	Max	Rango de muestras	Numero de muestras
1	25,00	50,00	1024-512	512
2	12,50	25,00	512-256	256
3	6,25	12,50	128-256	128
4	3,13	6,25	64-128	64
5	1,56	3,13	32-64	32
6	0,78	1,56	16-32	16
7	0,39	0,78	8-16	8
8	0,20	0,39	4-8	4

9	0,10	0,20	2-4	2
10	0,05	0,10	1-2	1

Tabla 14. Descomposición en niveles de la señal del EEG

Como dato de interés la información útil de la señal del EEG se suele encontrar en el rango de los 1 – 25 Hz por lo que no es necesario implementar todos los niveles para regenerar decentemente la señal, lo cual permite reducir enormemente el tiempo de procesamiento del dispositivo.

Mediante la herramienta Matlab se puede realizar la descomposición de la señal hasta un nivel N con un tipo de wavelet especifico, utilizando el comando ***wavedec*** del toolbox wavemenu con la siguiente estructura: [c,l] = wavedec(señal,nivel,tipodewavelet); donde c retorna los coeficientes, y l la cantidad de muestras por nivel tanto del coeficiente aproximado como de los detallados.

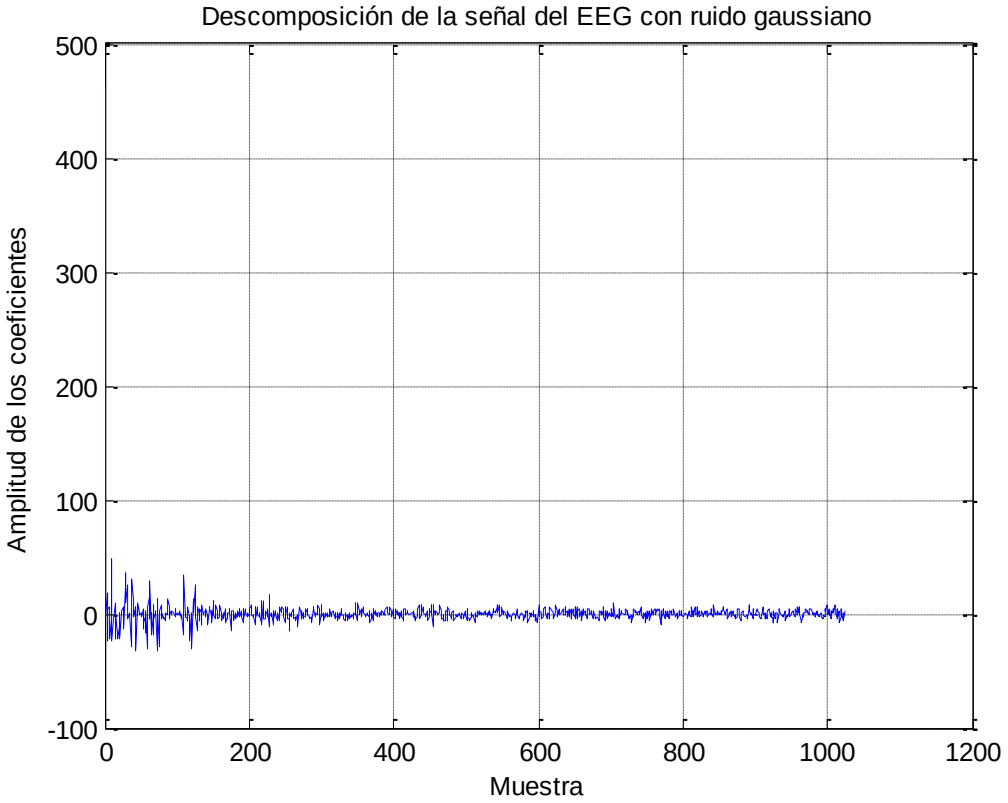


Figura 21. Coeficientes de la señal con ruido gaussiano.

Utilizando la DWT haar se obtienen los coeficientes de la Figura 21, en la cual se aprecia que el rango de muestras en que se manifiesta la mayor magnitud de amplitudes de los coeficientes se encuentra entre las 1 - 200 muestras, es decir entre los niveles 10 y 3 de la Tabla 14, (Los coeficientes de mayor nivel permiten

regenerar con mayor fidelidad la señal). Este tipo de subdivisión en bandas permite representar con una mejor resolución que la FFT los componentes de frecuencia que más actividad tienen de la señal.

2.2.2. Seleccionar el nivel del umbral

Comúnmente para el filtrado de la señal, se utiliza la umbralización dura o la umbralización suave, en la umbralización dura los coeficientes de cada nivel bajo un umbral λ son cero, y los coeficientes por encima de este umbral no son cambiados, mientras que en la umbralización suave los coeficientes que están por encima del umbral son atenuados a cero por un offset λ [45].

En general la umbralización suave da menores artefactos a la salida y conserva de manera efectiva la forma de la señal, la elección de un valor de umbralización es crucial para la eliminación de ruido de la señal [45]. Debido a esta información, se modifican los coeficientes de la sección anterior con una umbralización suave; para esta ocasión se selecciona la umbralización universal ***sqtwolog*** mediante los comandos de Matlab:

```
TPTR= 'sqtwolog'
thr = thselect(c,TPTR);
```

Para aplicar el umbral a todos los coeficientes se utiliza el comando de Matlab wthresh:

```
ytsoft = wthresh(c,'s',thr);
```

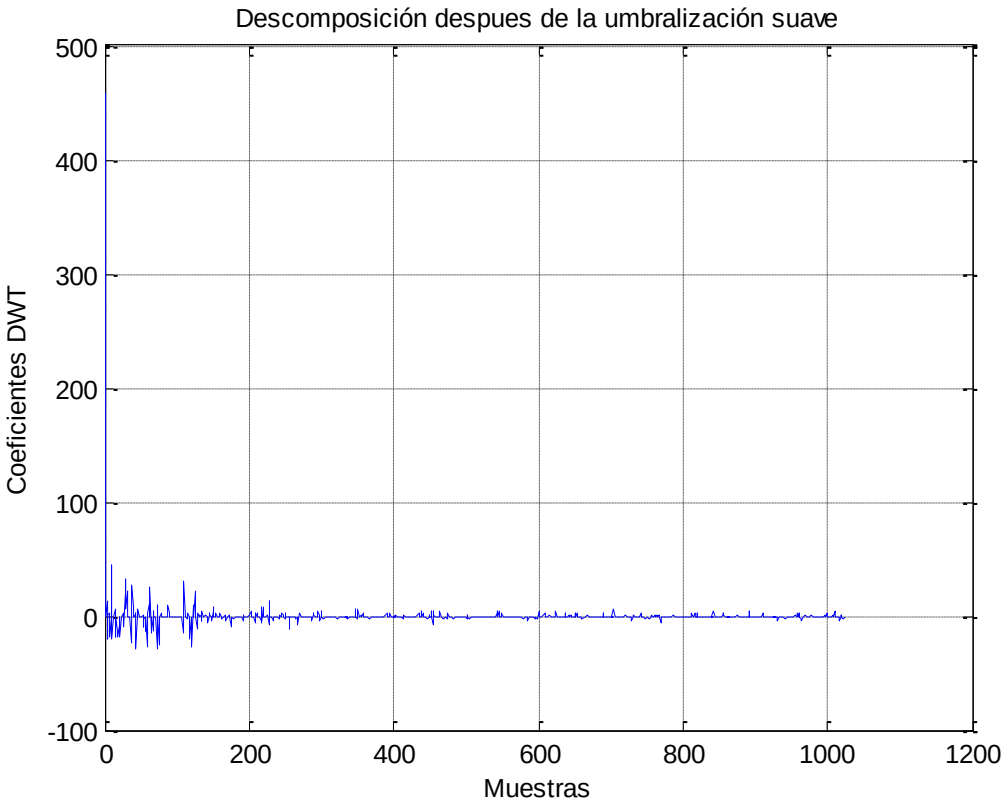


Figura 22. Coeficientes después del umbralizado suave

2.2.3. Reconstruir la señal

Se utilizan los coeficientes detallados y los coeficientes aproximados de nivel N, además se realiza la TW inversa mediante los filtros de reconstrucción.

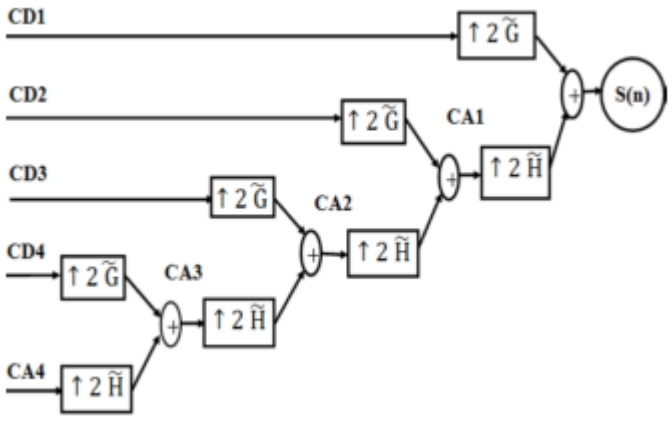
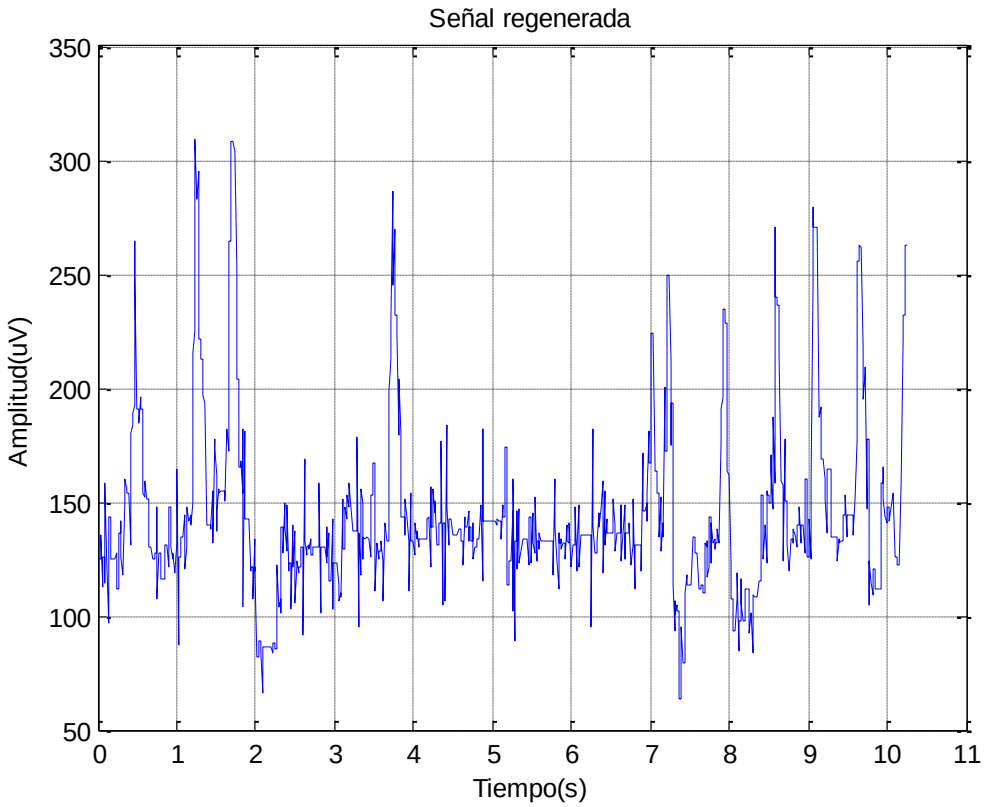


Figura 23. Esquema de reconstrucción multi resolución.

A partir de los coeficientes modificados por el umbral se debe regenerar la señal con la transformada inversa mediante la ilustración de la Figura 23 [43], la cual consiste en realizar un interpolación de los coeficientes y posteriormente realizar la convolución con los coeficientes de los filtros, para esto se utiliza la función ***waverec*** de Matlab mediante el comando.

waverec(ytsoft,l,'haar').



.

Figura 24. Reconstrucción de la señal del EEG

Para este ejemplo ***waverec*** en base de lo acontecido después de la umbralización suave, la cantidad de muestras por nivel y el tipo de wavelet utilizada, está en capacidad de regenerar la señal Figura 24. En la Figura 23, se observa el esquema de reconstrucción en base de los coeficientes para generar la señal, se debe tener en cuenta que el valor de los filtros H y G es diferente para la descomposición y reconstrucción, $\uparrow 2$ es un interpolacion que duplica la frecuencia de los coeficientes de entrada y el valor del coeficiente aproximado y los detallados dependen de la descomposición

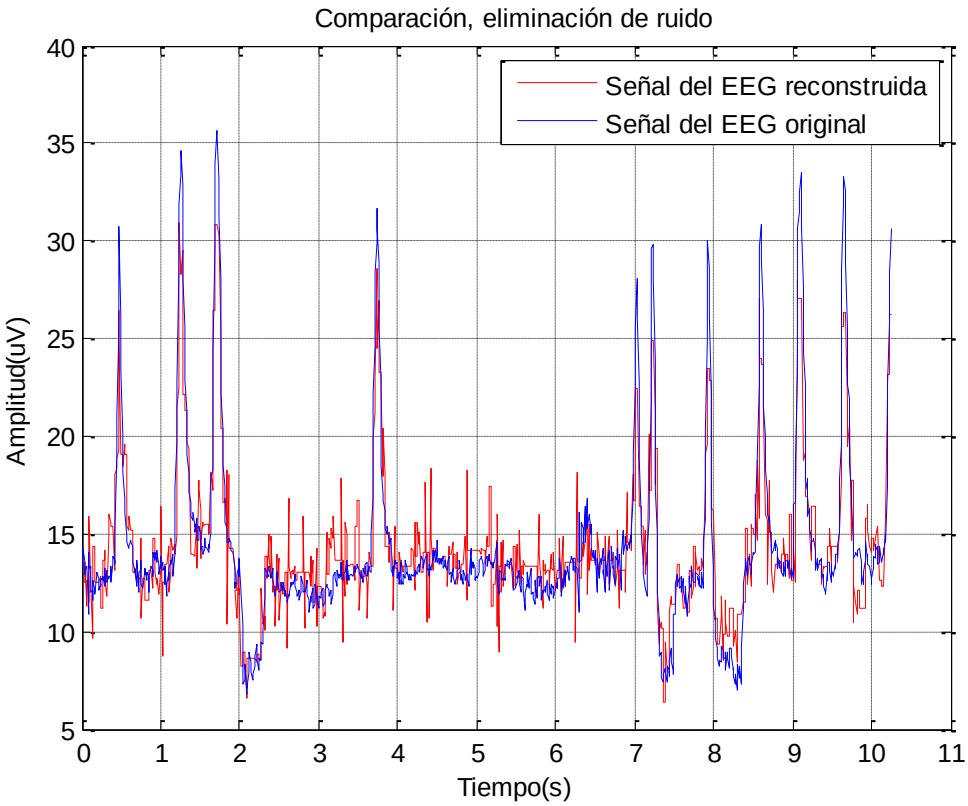


Figura 25.Comparación entre la señal original y la señal reconstruida.

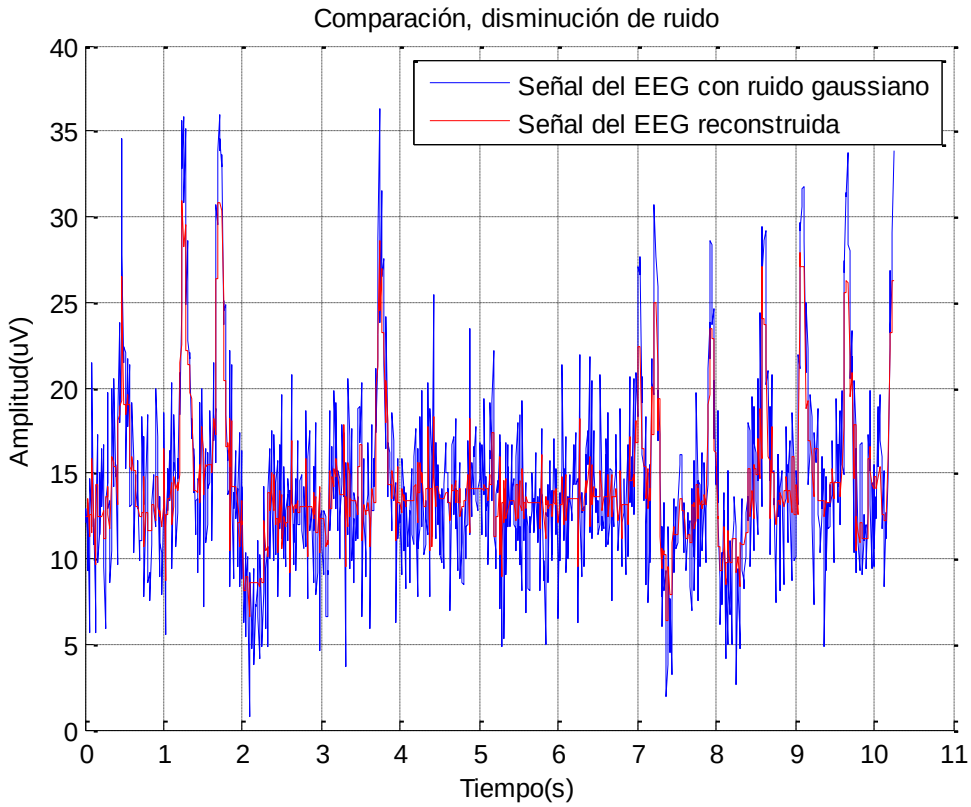


Figura 26. Comparación entre la señal reconstruida y la señal original con ruido gaussiano.

En la Figura 26, se puede apreciar a simple vista que la señal reconstruida tiene una menor cantidad de ruido que la implementada al principio del capítulo, al mismo tiempo, en la Figura 25, se aprecia que el filtrado puede tener mejores características, ya sea variando los niveles de descomposición, ajustando el nivel de umbral mediante otra técnica o de manera manual, o buscando otro tipo de wavelet que se ajuste a las necesidades de la señal.

SNR	C. Correlación	RMSE
-5dB	0.9934	1.7433
-10dB	0.9783	3.2153
-15dB	0.9367	5.6337
-20dB	0.8400	9.9170

Tabla 15. Comparación entre la señal original y la señal con ruido gaussiano.

SNR	C. Correlación	RMSE
-5dB	0.9951	1.5265
-10dB	0.9917	1.9487
-15dB	0.9730	3.5074
-20dB	0.8961	7.4254

Tabla 16. Comparación entre la señal original y la señal recuperada mediante DWT

Para comparar mediante un escalar la similitud entre los dos vectores se ha utilizado el coeficiente de correlación, y el RMSE para diversos niveles ruido. Entre más cercano sea el coeficiente de correlación a 1, y el RMSE a 0 mayores semejanzas tienes las dos señales que se están comparando.

Mediante los resultados de la Tabla 15 y Tabla 16, se observa que, este esquema de eliminación de ruido es útil cuando el ruido gaussiano es de bajo nivel, de lo contrario se vuelve casi imposible reconstruir la señal; de igual manera al existir distintos métodos para definir un umbral vuelve de carácter significativo conocer el rendimiento de cada método de discriminación de ruido para implementar el más adecuado.

Para reconocer el comportamiento de cada método de umbralización las técnicas rigrsure, sqtwolog, heursure y minimaxi han sido aplicadas tanto con umbralización dura como con umbralización suave a un canal de la señal del EEG de 4 sujetos (aw, av, al, ay) usados en la BCI competition III, pertenecientes al data set IVa (motor imagery) facilitados gracias a Fraunhofer FIRST, grupo de análisis de datos [41], estos datos fueron filtrados entre 0.05 – 200 HZ y luego digitalizados a 16 bits, con una precisión de 0.1uV.

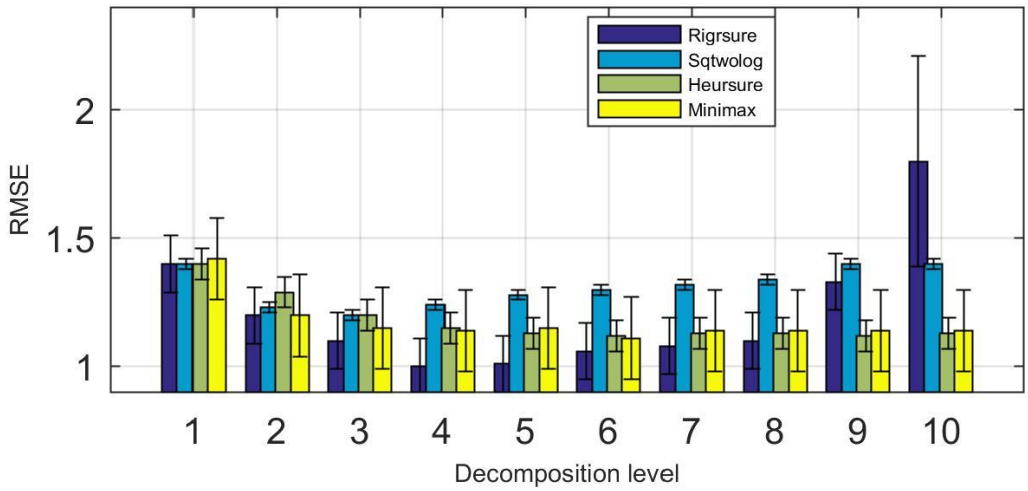


Figura 27. RMSE promedio al eliminar el ruido de una señal con el método SOFT

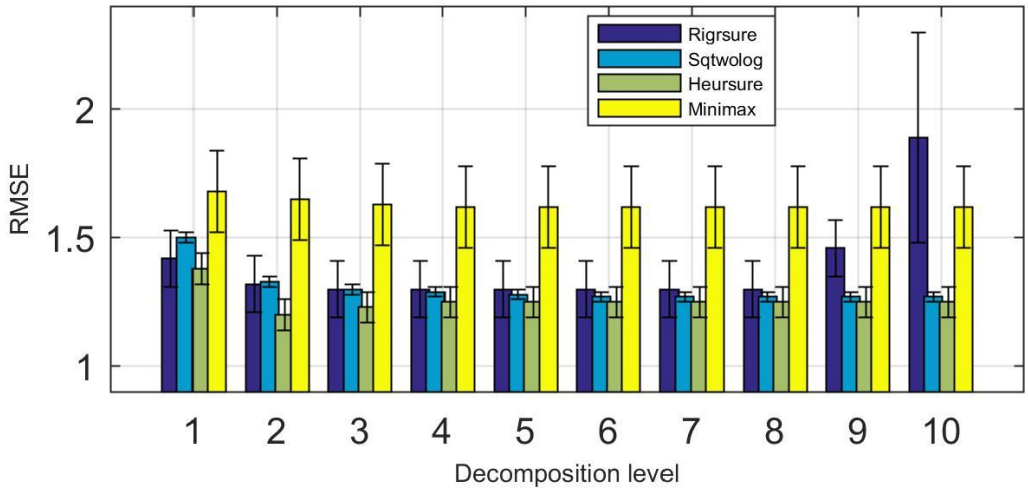


Figura 28. RMSE promedio al eliminar el ruido de una señal con el método HARD

Debido a la naturaleza de los resultados comparativos de la Figura 27 y Figura 28 [1] para eliminar ruido se ha discernido que estas diferencias de RMSE no son tan significativas como para descartar el funcionamiento de cada uno de los tipos de selección de umbral y por tanto se ha mirado en la eficiencia con la que se puede implementar cada uno de estos métodos en hardware, por su implementación sencilla en términos algorítmicos y la poca desviación estándar que presenta entre los varios sujetos el umbral sqtwolog con el método hard, es de interés para este proyecto.

2.3. TW Haar

La TW Haar ha sido elegida para el desarrollo de este proyecto como se describió en la sección 1.4, por ser una técnica que no recurre a operaciones en punto flotante, lo que la hace ventajosa frente a otras técnicas, además de que la Haar se compone de una secuencia de filtros de paso bajo y paso alto, conocidos como los bancos de filtros. Los bancos de filtros de la TW Haar tienen coeficientes de escasa complejidad numérica que permiten un eficiente uso de memoria, ya que, sus cálculos matemáticos son simples en comparación con otras wavelets que conducen a ocupar más espacio en memoria volátil y requerir de mayores recursos para realizar operaciones.

2.4. FPGA

En base a la superioridad que ha demostrado la FPGA ante los DSP en los antecedentes así como también en el resumen planteado en la sección 1.4, en conjunto con la capacidad de realizar y analizar la codificación de manera práctica,

se ha decidido realizar la tarea de eliminación de ruido mediante la TW en una FPGA.

Las implementaciones en las FPGAs son desafiadas por los ASIC, debido a sus habilidades aritméticas, sin embargo la escalabilidad de la FPGA que le permite tener una fácil adaptabilidad para futuros cambios es de interés para este proyecto, debido a que dependiendo del tipo de señal puede ser conveniente modificar la cantidad de niveles de descomposición o la familia wavelet.

2.4.1. DE0-Nano

Para la verificación de este proyecto se ha utilizado la FPGA DE0-nano que cuenta 22.320 elementos lógicos, con el fin de verificar el algoritmo de la TW a través de una sola señal, asimismo, se tiene en cuenta la finalidad de que este proyecto pueda ser utilizado con facilidad entre distintas marcas de FPGAs por lo que el lenguaje en que se describe la codificación es VHDL.

Figura 29. FPGA DE0-nano

2.5. Modelo teórico

El método convolucional consiste en aplicar la convolución entre la señal de entrada y el valor de los filtros pasa altas y pasa bajas, (el valor de los filtros cambia con respecto a la familia wavelet que se esté utilizando) sin embargo eso conlleva a la necesidad de utilizar multiplicadores, y registros que almacenen constantemente los valores de la convolución.

Aunque el método convolucional se puede optimizar mediante la aplicación de LOOKUP TABLES, se ha decidido implementar la descomposición y reconstrucción de la señal mediante el método lifting por su sencillez de cálculo para valores enteros.

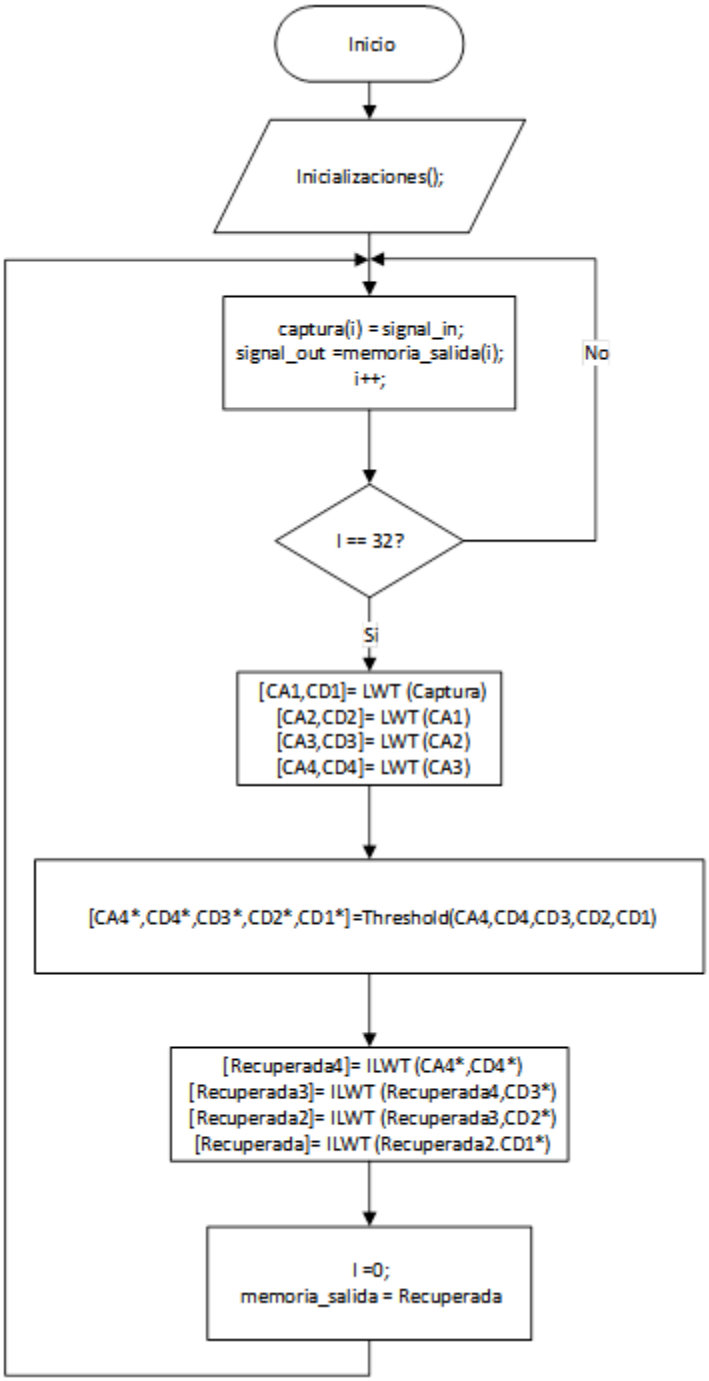


Figura 30. Diagrama de flujo de la eliminación de ruido por medio de la TW.

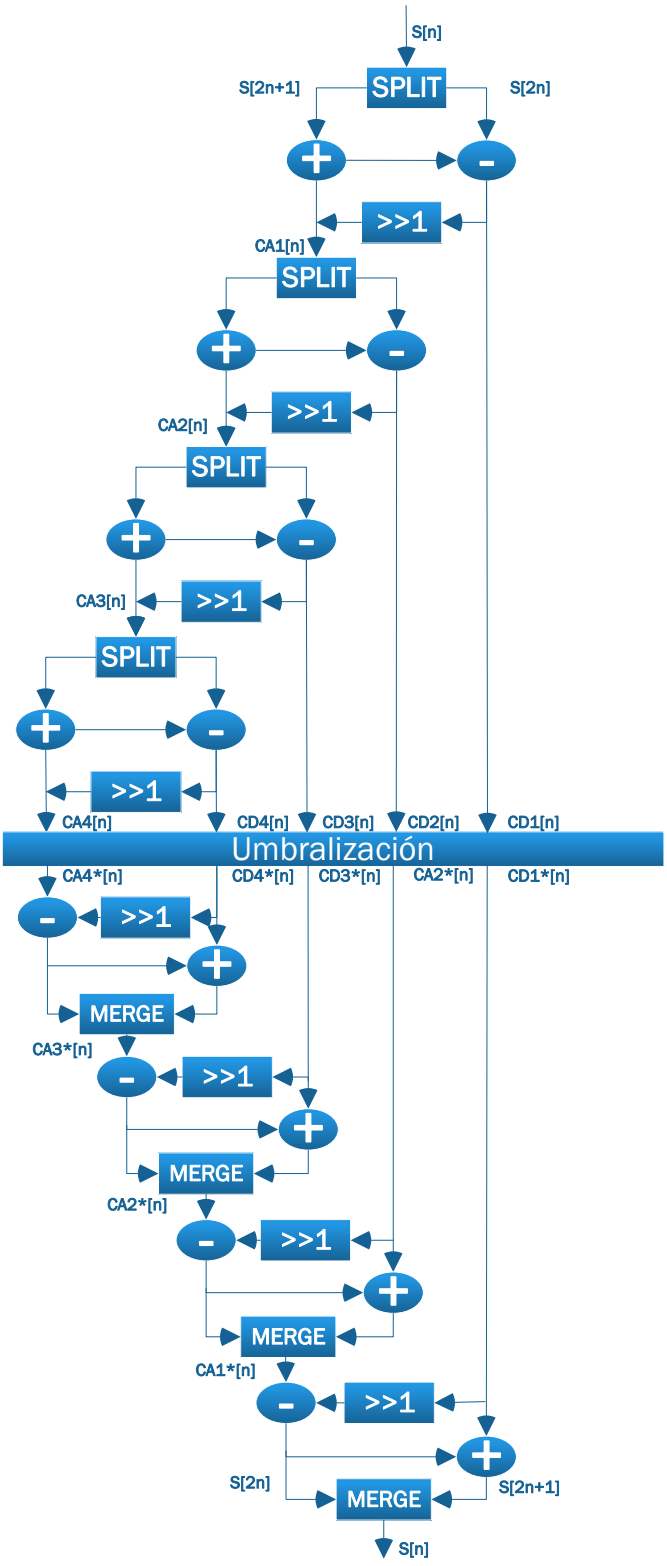


Figura 31. Modelo teórico implementado en hardware.

En la Figura 30 y Figura 31 se observa el modelo que representa la TW Haar; en base de los coeficientes aproximados se crean más niveles de descomposición, hasta llegar a 4 niveles para mejorar la eliminación de ruido con la técnica sqtwolog[1].

Para realizar este modelo se debe tener en cuenta que la etapa de umbralización se encarga de calcular por tramas la MAD y después otorgar un valor de umbral, por lo que es necesario almacenar los datos de entrada, posteriormente ejecutar la umbralización y finalmente otorgar una salida basada en los coeficientes después de la umbralización.

3. DESARROLLO DEL MODELO

Para dar lugar al desarrollo de un modelo en capacidad de representar en hardware la TW Haar, se ha implementado el algoritmo de descomposición lifting, con el técnica sqtwolog y la umbralización hard. Describir el proceso en hardware se ha realizado con base en las siguientes etapas.

3.1. HDL coder

Para la implementación del código en VHDL se han vislumbrado varias opciones que van desde la escritura del código directamente en VHDL parar representar cada etapa, hasta el uso de conversores de lenguaje. Finalmente se ha implementado el conversor de lenguaje HDL coder que permite transformar código escrito en la plataforma de Matlab a código sintetizable en VHDL.

HDL coder se ha elegido sobre la implementación directa en VHDL debido a:

- Facilidad para realizar cambios futuros en el código

La escalabilidad del sistema para pasar de una familia wavelet a otra, modificar los niveles de descomposición o cambiar el método de descomposición es mucho más rápida y comprensible de realizar que un algoritmo escrito plenamente en VHDL.

- Facilidad de simulación

HDL coder exige un test bench antes de sintetizar el código en VHDL para asegurarse del funcionamiento deseado del algoritmo en Matlab, este test bench permite el uso de funciones propias de Matlab, lo cual facilita el análisis de los datos.

A pesar de la versatilidad de HDL coder muchos diseños referentes al procesamiento digital de señales requieren un rediseño que permita la generación de código HDL, esto es causado debido a que hay tipos de datos, estructuras, iteraciones y saltos que no se pueden mapear de manera correcta en hardware.

3.2. Variables persistentes

Las funciones en Matlab, se pueden convertir en bloques de HDL, a estas se les deben indicar las variables de entrada y salida; no obstante, el tipo de dato (booleano, con signo y la cantidad de bits) es detectado por Matlab mediante el test bench.

Si entre cada llamado del bloque se requiere conservar el valor de una variable, estas deben ser declaradas como variables de tipo persistente, las cuales pueden ser mapeadas en la memoria RAM o almacenadas como registros, todo dato es almacenado en un registro de 32 bits si no se indica lo contrario.

```
initval1 = int16(zeros(1,64));
persistent z s%%Declaración de variables persistentes
if isempty(z)%Inicialización de las variables persistente
s = initval1;
z = initval1;
end
```

Figura 32. Declaración de las variables persistentes s y z.

En la Figura 32 se observa la creación de las variables persistentes s y z, como arreglos de 64 datos de16 bits con signo, inicializados en cero

3.3. Algoritmo TW Haar en Matlab

A pesar de que el toolbox wavelet de Matlab posee funciones para descomponer y reconstruir una señal utilizando la TW Haar en el esquema lifting, estas no son soportadas para su uso en conjunto con HDL coder, más aún, las funciones de la TW Haar pueden ser utilizadas por el test bench para verificar la fiabilidad de las funciones diseñadas con la teoría, sin embargo, no para la generación de código en VHDL que pueda funcionar sin Matlab.

Mediante (26) y (27) se obtienen los coeficientes aproximados y detallados de cada nivel, su algoritmo se representa en la Figura 33, el primer paso realizado es dividir la señal de entrada en dos grupos, uno conformado por muestras impares y otro por muestras pares (SPLIT), posteriormente, los coeficientes detallados son encontrados hallando la diferencia entre cada par de muestras adyacentes (PREDICT) y finalmente los coeficientes aproximados son calculados mediante el promedio del par de muestras adyacentes (UPDATE) [46].

```
function [CoefAprox,CoefDetail] = lwthaar(senal,lengths)
Impar = int16(zeros(1,lengths));
Par = int16(zeros(1,lengths));
n = (1:lengths);
Impar(n) = (senal(2*n-1));  % odd signals, separación impar (SPLIT)
Par(n) = (senal(2*n));  % even signals, separación par (SPLIT)
CoefDetail = ((Impar)-(Par)); %Creación de los coeficientes detallados (PREDICT)
CoefAprox = (Par+(bitshift(int16(CoefDetail),-1))); %Coeficientes aproximados (UPDATE)
end
```

Figura 33. Descomposición mediante la TW Haar en el esquema lifting en Matlab.

Usando (28) y (29) se pueden reconstruir los componentes pares e impares de la señal a partir de los coeficientes aproximados y detallados Figura 34, finalmente pasan por la función MERGE para reestablecer la secuencia original.

```
function [recuperada] = ilwthaar(CoefAprox,CoefDetail)
Par = CoefAprox-(bitshift(int16(CoefDetail),-1));%Creación de los componentes pares (UNDO UPDATE)
Impar = (Par)+(CoefDetail);%Creación de los componentes impares (UNDO PREDICT)
recuperada = [(Impar);(Par)]; %(MERGE)
recuperada = (recuperada(:).');%%(MERGE)
end
```

Figura 34. Reconstrucción mediante la TW Haar en el esquema lifting en Matlab.

3.4. Calculo del umbral

Para definir el nivel del umbral se utiliza la técnica sqtwolog, primero se calcula la MAD mediante (20) los coeficientes detallados del primer, segundo, tercer, cuarto nivel y el coeficiente aproximado del cuarto nivel, son usados para calcular la desviación media de los valores absolutos, posteriormente se multiplica por (19) siendo N 32, el cual es el valor mínimo con el que se pueden obtener resultados validos en los umbrales de acuerdo con Kalantari *(et al)* [47]

$$THR = \frac{1}{0.7980}\sqrt{2 * \log(32)} = 2.17 \tag{31}$$

El valor de la constante se ha aproximado a 2 para allegar su cálculo a un corrimiento a la izquierda y evitar el uso de multiplicadores al sintetizar el código en la FPGA, finalmente el umbral se representa mediante (32).

$$THR = 1 \ll MAD \tag{32}$$

3.5. Calculo de la MAD

Tanto la desviación media absoluta como la desviación mediana absoluta son medidas que indican la dispersión de los datos de una señal, la MAD es útil para definir los valores de la señal que son ruido, mediante el escalado que proporciona a la umbralización universal sqtwolog.

```
[CoefAprox,CoefDetail] = lwthaar(signal_out_memory,uint8(16)); %% Primer nivel descom
 media = sum(CoefDetail,'native')+media;%Acumulacion de los datos
[CoefAprox2,CoefDetail2] = lwthaar(CoefAprox,uint8(8));%% Segundo nivel descom
media = sum(CoefDetail2,'native')+media;
[CoefAprox3,CoefDetail3] = lwthaar(CoefAprox2,uint8(4));%% Tercer nivel descom
media = sum(CoefDetail3,'native')+media;
[CoefAprox4,CoefDetail4] = lwthaar(CoefAprox3,uint8(2));%% Cuarto nivel descom
media = sum(CoefDetail4,'native')+media;
media = sum(CoefAprox4,'native')+media;%%Suma de todos los datos
C = [CoefAprox4 CoefDetail4 CoefDetail3 CoefDetail2 CoefDetail];
mediari=bitshift(media,-5);%;Calculo de la media aritmetica de los datos
```

Figura 35. Calculo de la media aritmética de los coeficientes resultantes en la descomposición.

Para el cálculo de la MAD es necesario en primer lugar encontrar la media entre todos los coeficientes que se encuentran en la descomposición, en la Figura 35 se aprecia la implementación en Matlab compatible con HDL coder del cálculo de la media, a partir de la suma de todos los coeficientes y su posterior división para encontrar el promedio aritmético, se realizan diversas sumas sucesivas en lugar de una sola para todo el arreglo que contiene los coeficientes wavelet, en orden de mejorar la manera en que HDL coder interpreta el código .

```
function CoefDetail = threshold(CoefDetail,umbral)
media = int16(zeros(1,32));
acumulador = int16(0);
for a = 1:32
    temporal = umbral-CoefDetail(a);
if (temporal)<0
     media(a) = CoefDetail(a)-int16(umbral);
else
    media(a) = int16(umbral)-CoefDetail(a);
end
acumulador = media(a)+acumulador;%%Desviación estandar absoluta
end
temporal =bitshift(acumulador,-5);% Calculo de la MAD
THR1 = bitshift(int16(temporal),1);%Umbral Sqtwolog
```

Figura 36. Calculo de la desviación estándar absoluta y la MAD.

Mediante la media aritmética de los coeficientes, se calcula la desviación estándar absoluta, posteriormente, se calcula la media aritmética de las desviaciones estándar absolutas para encontrar la MAD.

3.6. Umbralización dura

Para conocer los valores de la umbralización dura se dispone de la Figura 37, solo se modifica el coeficiente que ha sido generado en el proceso de descomposición en caso de que el valor absoluto del coeficiente actual es menor al umbral establecido por la MAD y la técnica sqtwolog.

```
for i = 1:64
if ((CoefDetail(i))>=THR1 ||(-CoefDetail(i))>=THR1 )%%Umbralización dura
else
    CoefDetail(i) = int16(0);
end
end
end
```

Figura 37. Umbralización dura códificada.

3.7. Test bench

HDL coder permite el uso de todas las funciones de Matlab en el test bench, esto se aprovecha para probar la TW con 32 datos. Figura 38. Esto permite reconocer si el comportamiento de la TW es el adecuado antes de crear el código en VHDL y sintetizarlo en una FPGA.

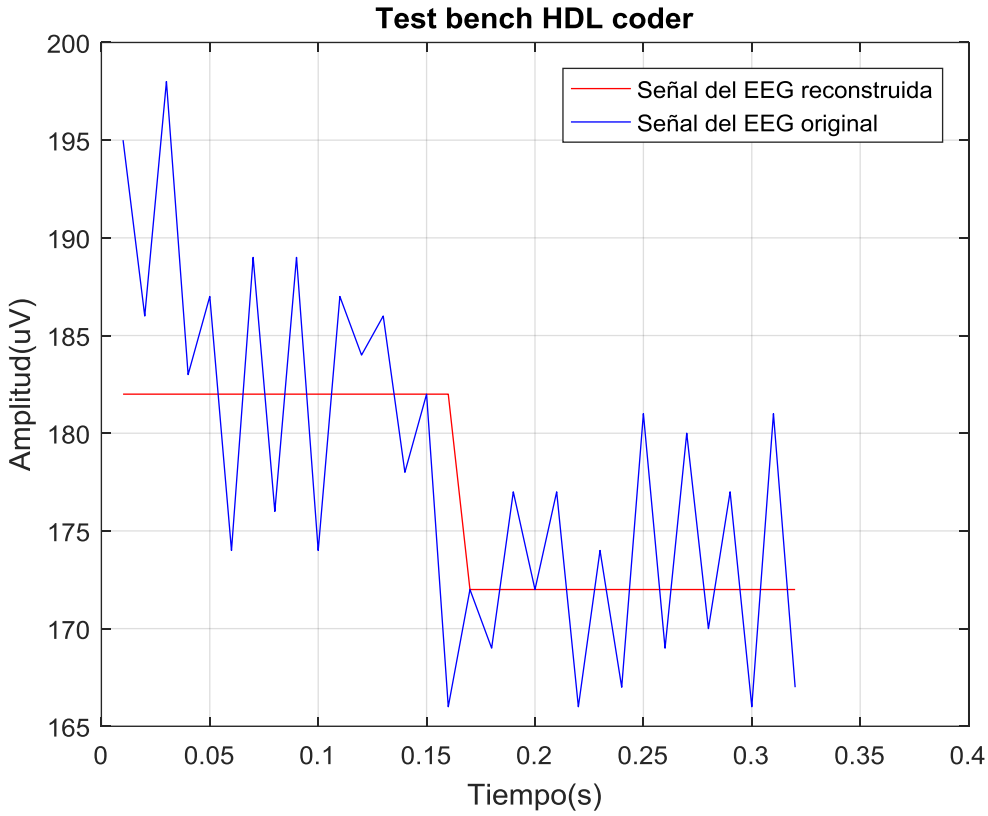


Figura 38. Código de prueba para la TW con HDL coder.

3.8. Conversión a VHDL

HDL coder ofrece la facilidad de crear diseños con lógica de punto fijo a partir de diseños en punto flotante esto permite reducir la complejidad de elaboración del sistema, sin embargo, para esta etapa de eliminación de ruido se han manejado únicamente valores enteros Figura 39, debido a que en trabajos como el de Acharya *(et al)* [43] se demuestra la eficacia del diseño con lógica entera y se establece que no es necesario recurrir a diseños de punto fijo para obtener un buen filtrado de la señal.

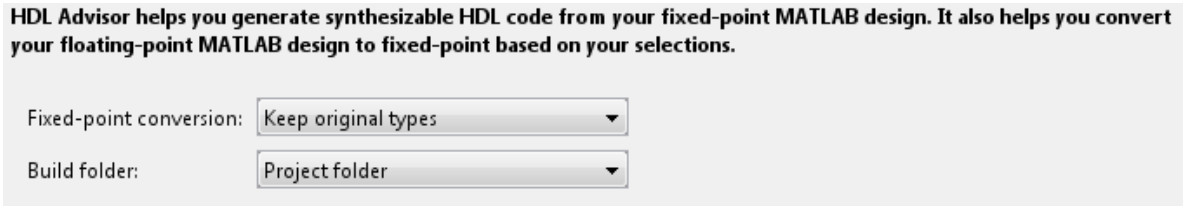


Figura 39. Selección del tipo datos.

El sistema cuenta con una entrada definida de 16 bits con signo, y una señal de start de tipo booleana Figura 40, encargada de definir el momento en que se desea habilitar la conversión.

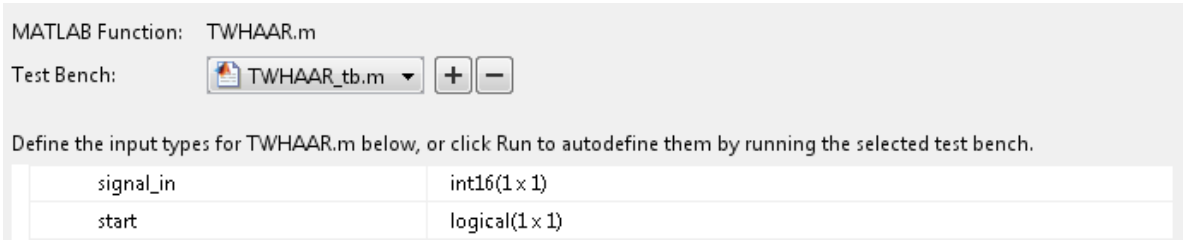


Figura 40. Definición de las señales de entrada del bloque de VHDL

El código generado en VHDL, posteriormente puede ser modificado o sintetizado directamente sobre una FPGA o ASIC que soporte los requerimientos de la codificación.

4. PRUEBAS Y ANÁLISIS DE LOS RESULTADOS

La verificación del bloque de VHDL encargado de representar la TW Haar, fue realizada mediante el uso de la herramienta FIL de Simulink, esta permite comunicar directamente los componentes de Matlab con la FPGA, a través de un bloque de Simulink por una conexión JTAG, Ethernet o PCI Express. En la Figura 41, se ilustra los tipos de conexión que se pueden configurar para la comunicación entre Matlab y la FPGA mediante FIL.

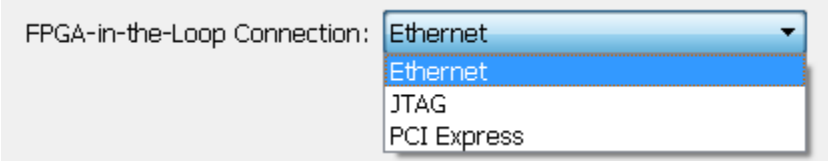


Figura 41. Conexiones aceptadas por FIL.

FIL ofrece soporte para diversas FPGAs fabricadas por Altera y Xilinx; es necesario que el software Quartus II esté instalado y se indique la ruta de instalación en Matlab ilustrado en la Figura 42, para poder realizar la verificación y carga del archivo .sof a la FPGA. A pesar de que una referencia específica de FPGA no se encuentre actualmente en la lista de dispositivos soportados por FIL, el modelo puede ser agregado a FIL, indicando los pines de conexión de la FPGA.

```
hdlsetuptoolpath('ToolName','Altera Quartus II','ToolPath','C:\altera\15.0\quartus\bin64\quartus.exe');
```

Figura 42. Código para añadir la ruta de la versión 15.0 de Quartus II a Matlab utiliza en este proyecto.

En la Figura 43, se indica la configuración realizada en FIL para este proyecto. Mediante la FPGA DE0-Nano conectada por medio de JTAG, se establece la verificación del bloque en VHDL. El método de conexión elegido es JTAG, debido a que el dispositivo DE0-Nano, solo soporta este tipo de conexión, además este método está disponible para un gran abanico de FPGAs.

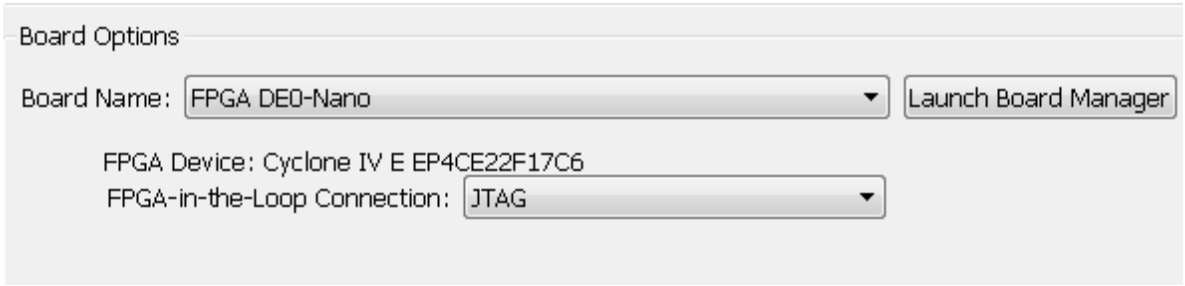


Figura 43. Selección de la FPGA y la comunicación a utilizar con Simulink.

4.1. Verificación mediante FPGA IN LOOP

Todas las señales se pueden modificar desde Simulink, adicionalmente, si el usuario lo desea puede mapear algunas señales con la FPGA, de manera que el bloque de VHDL puede conectarse a los pines asociados de la FPGA; en este proyecto las señales del *clk, reset* y *clk_enable* ilustradas en la Figura 44, son mapeadas a los pines predeterminados de la FPGA, mientras que la señal de entrada, el bit de inicio, el bit que indica si el sistema está ocupado y la salida de la señal, están disponibles para ser conectados a otros bloques de Simulink.

Port Name	Port Direction	Port Width	Port Type
clk	In	1	Clock
reset	In	1	Reset
clk_enable	In	1	Clock enable
signal_in	In	16	Data
start	In	1	Data
ce_out	Out	1	Data
signal_out	Out	16	Data

Figura 44. Selección de las señales de entrada y salida del módulo en VHDL.

En la Figura 45, se ilustra la configuración efectuada a las señales de salida, en esta parte, se indica la representación de las señales, por ejemplo, *booleana, entera, doble, etc.* Así como también, si la señal será representada o no en complemento a dos.

Output Name	Bit Width	Data Type	Sign	Fraction Length
ce_out	1	Boolean		
signal_out	16	Integer	Signed	

Figura 45. Selección del tipo de dato de salida para la verificación del módulo en VHDL.

A pesar de que existe la opción de que FIL asigne automáticamente el tipo de dato, es conveniente seleccionar de manera manual la representación de las señales, para que la verificación siempre otorgue información coherente.

El programa de Simulink consiste en comunicar, un canal de uno de los sujetos (aa, aw, ay, al) de la BCI competition III [41], con la FPGA. La FPGA se encuentra representada por el bloque denominado “TWHAAR” de la Figura 46. La señal proveniente de los sujetos es modificada con ruido gaussiano, agregando una SNR de 20db de magnitud. El ruido gaussiano fue agregado con la función

awgn(señal,20,'measured') de Matlab, posteriormente la señal de salida de "TWHAAR" es transferida de Simulink al workspace de Matlab para el análisis de la TW.

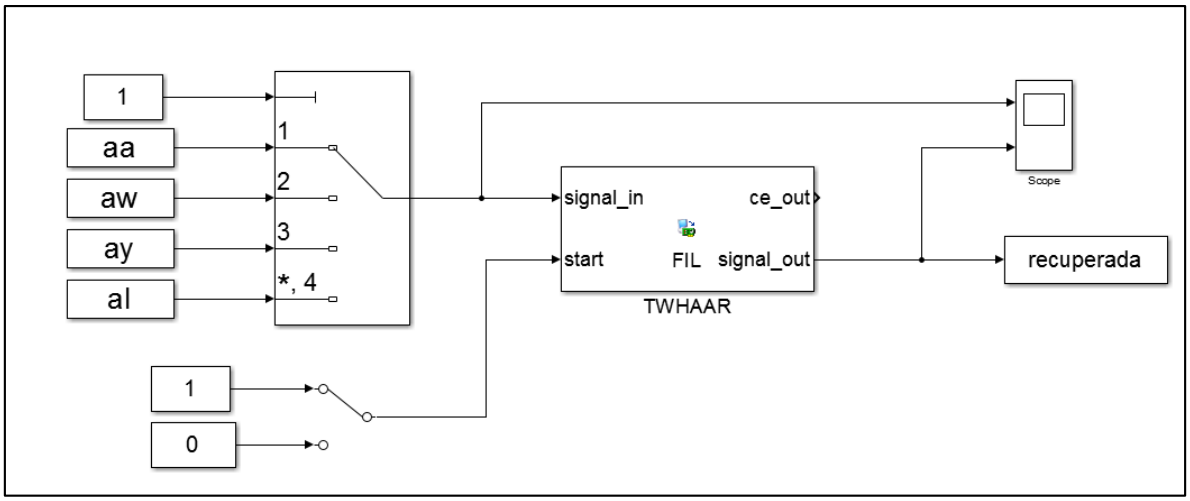


Figura 46. Bloque de VHDL verificado en Simulink.

El bloque "TWHAAR" ilustrado Figura 46, contiene la representación del código implementado en VHDL, donde se usan bancos de memoria, compuertas lógicas, Flips-Flops, registros de corrimiento.

Para el análisis numérico de la señal de salida, se utilizan las métricas PSNR y RMSE. La relación señal a ruido pico (PSNR) se utiliza para definir la analogía entre la máxima energía posible de una señal y el ruido que afecta a esta. La definición del PSNR se ilustra en (33) [48], donde es indispensable la formulación del error cuadrático medio:

$$PSNR = 10log_{10}\left(\frac{{f_{max}}^2}{MSE}\right) \tag{33}$$

Donde f_{max} es el máximo valor de la señal y está dado por (34) y MSE es el error cuadrático medio.

$$f_{max} = ma\,x\big(max\big(f(k)\big), ma\,x\big(f_d(k)\big)\big) \tag{34}$$

Por otra parte, el cálculo del $RMSE$ esta dado en (35) [37]:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(x_i - y_i)^2} \tag{35}$$

Donde x_i es el elemento i de la señal original, y y_i es el elemento i que representa la señal de salida del proceso de eliminación de ruido, N es el número de muestras [37].

En la Figura 47 y Figura 49, se ilustran las señales de los sujetos (ay, aa) contaminadas con ruido y la señal después del proceso de eliminación de ruido mediante la TW. En la Figura 48 y Figura 50, se ilustran las señales originales de los sujetos (ay, aa) y la señal recuperada a partir de las señales contaminadas. Este proceso se realiza con cuatro niveles de descomposición, usando la técnica sqtwolog y el metodo hard.

Debido al retraso de 32 muestras que debe transcurrir para calcular el nivel del umbral, la señal recuperada por la FPGA es adelantada 32 muestras, con el fin de observar las señales en el mismo lapso de tiempo.

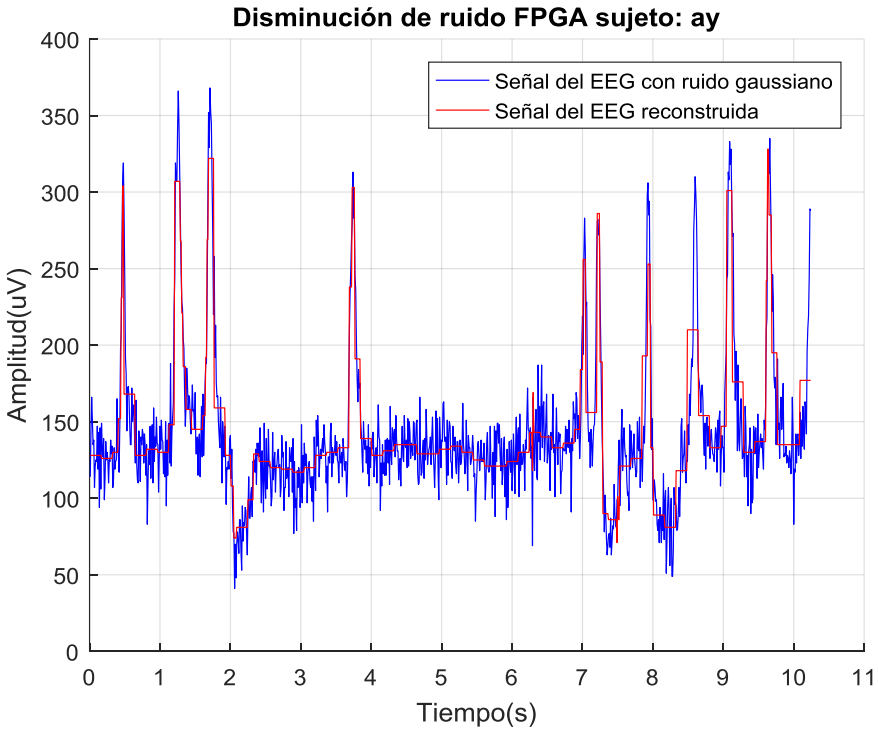


Figura 47. Señal del sujeto ay con ruido gaussiano con SNR de 20db.

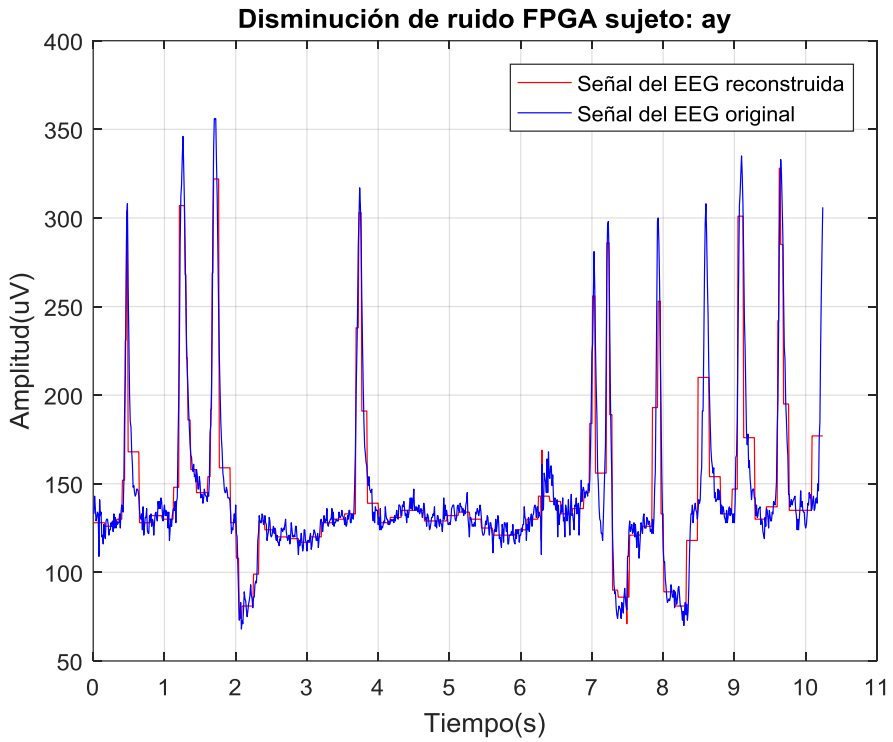


Figura 48. Señal original y señal recuperada del sujeto ay

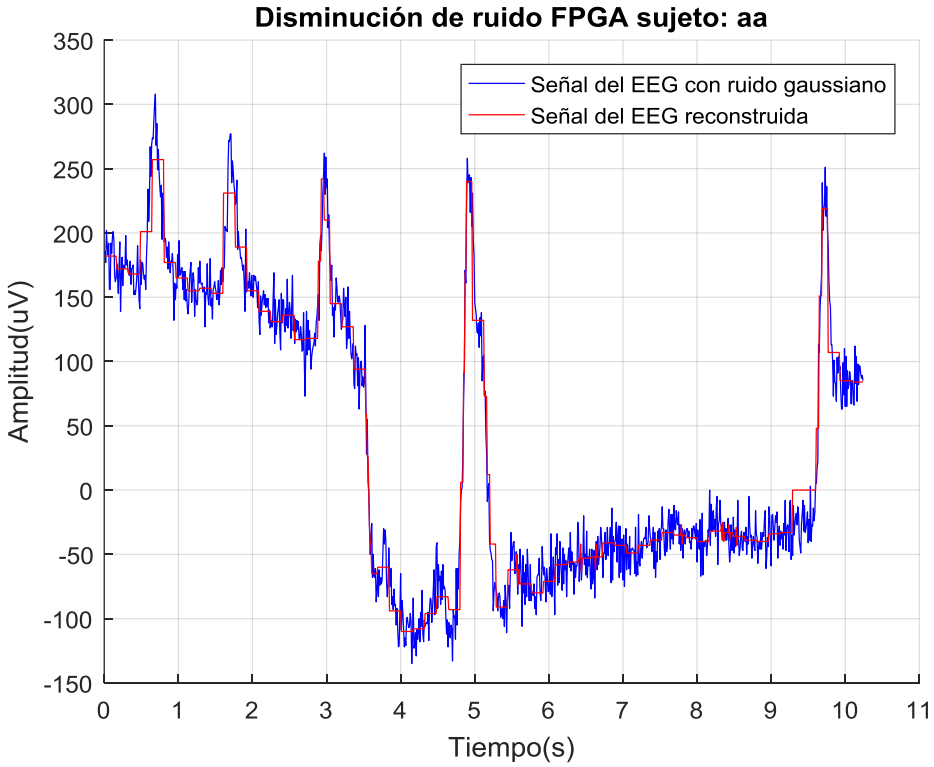


Figura 49. Señal del sujeto aa con ruido Gaussiano a 20db

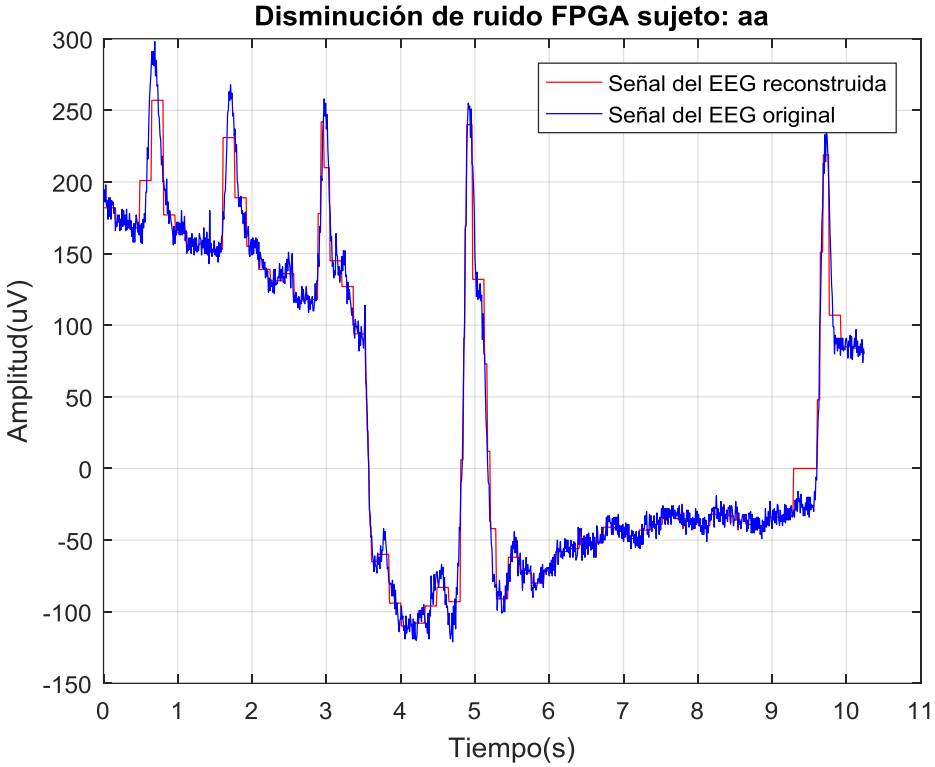


Figura 50. Señal del sujeto aa original y reconstruida.

Para verificar la eliminación de ruido se ha comparado la señal recuperada automáticamente por la función *wden* de Matlab para la wavelet Haar con 4 niveles de descomposición, ante el algoritmo desarrollado en la FPGA. En la Figura 50 y Figura 51 se tienen los resultados en términos de RMSE y PSNR respectivamente, donde se aprecia un mejor filtrado realizado por la TW en Matlab, en comparación del algoritmo implementado en la FPGA.

Las diferencias entre los resultados en Matlab y en la FPGA se atribuyen a las aproximaciones realizadas para lograr calcular el nivel de umbral mediante registros de corrimiento. En Matlab, los umbrales son calculados cada 1024 muestras, en su lugar, la FPGA utiliza 32 muestras que permiten conocer estadísticamente la manera en que el ruido se ha distribuido, de igual manera la TW utilizada en Matlab realiza operaciones en punto fijo, mientras la FPGA aproxima a valores enteros mediante el uso de registros de corrimiento.

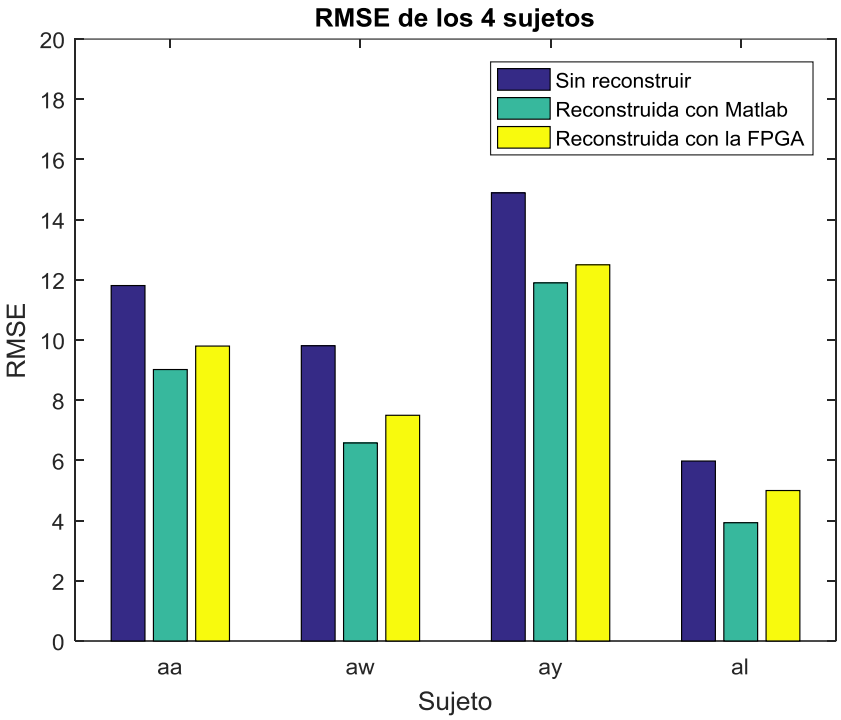


Figura 51. Disminución de RMSE con la TW Haar.

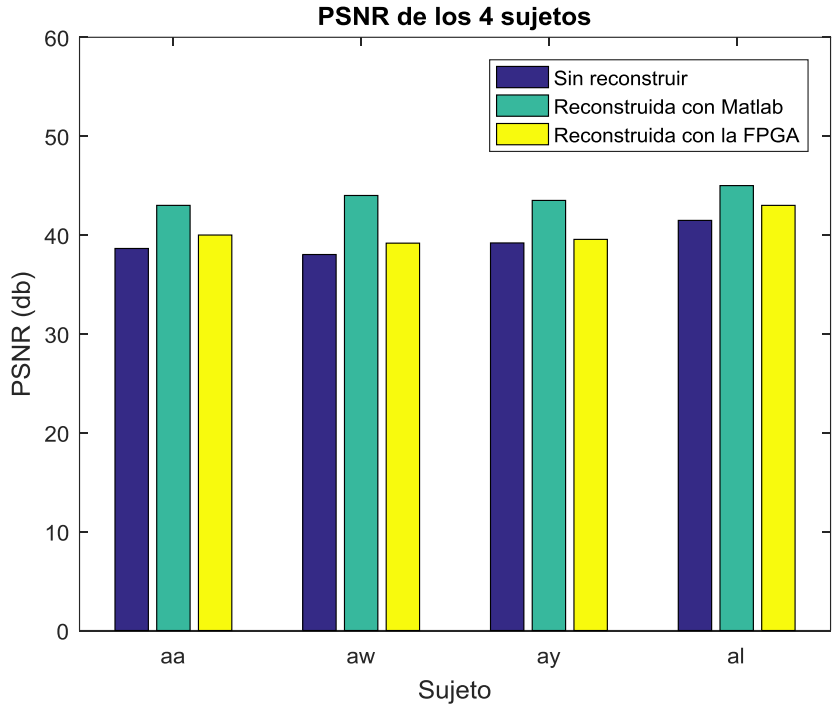


Figura 52. Aumento de PSNR con la TW Haar.

	Nuestro diseño	G. Ciprian [49]	L. Yang [50]
Tipo de wavelet	Haar	DB4	(5/3) wavelet filter
Niveles de descomposición	4	8	2
Método de umbralización	Sqtwolog	Sqtwolog	Sqtwolog
Método para aplicar la umbralización	HARD	Mejora de SOFT	HARD
Tamaño de los datos de entrada	16 bits	8 bits	16 bits
Tamaño de los coeficientes	16 bits	8 bits	16 bits
Traductor de código	HDL coder	---	Xilinx System Generator
FPGA	Altera Cyclone IV	Xilinx XC5VLX110TFF1136	FPGA Spartan 3E starter kit (XC3S500E)
Multiplicadores	0	32	1
Verificado mediante	FPGA IN LOOP	Modelsim	Chipscope
Elementos lógicos consumidos	11.912 LEs (53%)	27451 LEs (39%)	---
Frecuencia máxima (MHz)	34	384	---

Tabla 17.Trabajos que emplean la eliminación de ruido con la TW sobre una FPGA.

Mediante la realización del proyecto se han vislumbrado trabajos similares en los cuales se representa el proceso de filtrado de una señal como se observa en la Tabla 17, la implementación de la TW Haar entera ha permitido el ahorro de multiplicadores de la FPGA; a pesar de que el proyecto se encuentre enfocado en la DE0 Nano, la portabilidad que ofrece el lenguaje VHDL permite transportar fácilmente diversos bloques que apliquen la TW Haar a otro tipo de FPGA que soporte una mayor cantidad de elementos lógicos para un procesamiento multi-canal.

5. CONCLUSIONES

Se implementó la TW Haar sobre la FPGA DE0-Nano, mediante el método Lifting para el pre-procesamiento de señales unidimensionales no estacionarias. Los resultados obtenidos del proceso de filtrado de una señal variante con el tiempo, verifican que el sistema está en capacidad de eliminar ruido. El ruido agregado a las señales de los sujetos, fue un ruido gaussiano de 20 db de magnitud. Para el análisis de los resultados en términos numéricos, se utilizaron las métricas PSNR y RMSE, las cuales mostraron una mejora del 1.9 en términos de RMSE y 1.7 en términos de PSNR con relación a la señal original.

Se determinó la clase de TW a utilizar en el proyecto, mediante las investigaciones realizadas sobre las clases de wavelet, donde se obtuvieron sus propiedades y características, por ejemplo, la complejidad en términos de cálculos, la utilización de memoria, su forma de implementar, etc. Todos estos datos sirvieron para conocer y seleccionar la clase wavelet que más se ajustara a los objetivos del proyecto. Después de analizar y conocer los detalles de cada wavelet, se eligió la TW Haar, por su buen rendimiento, debido a que se compone de filtros paso bajos y paso altos, que no consume demasiada memoria, a causa de su escasa complejidad numérica, que por ende, le permite no requerir mayores recursos para realizar operaciones matemáticas además de su buena referencia en el manejo de señales biológicas.

Para la representación de la TW Haar, se analizaron dos algoritmos, el método convolucional y el método lifting, no obstante, para el desarrollo de la etapa de eliminación de ruido, solo se utilizó el método de lifting, puesto que este, en su versión de entero a entero consiste únicamente de sumadores y registros de corrimiento, asimismo, en comparación con el método convolucional, el método lifting requiere de una menor cantidad de operaciones para obtener resultados y además no demanda unidades de memoria adicionales para almacenar información.

Como dispositivo electrónico para embeber la TW se ha elegido la FPGA Altera DE0-Nano, debido a que en los antecedentes y al igual que en la sección 1.4, se ha evidenciado superioridad en términos de eficiencia computacional y la versatilidad que ofrece para embeber algoritmos, en comparación con otros procesadores de señales digitales vislumbrados en el estado del arte, como son los DSP. Por ejemplo, en un artículo ilustrado en la sección 1.4, se compara la implementación TW en una FPGA utilizando código optimizado en VHDL, contra una implementación en un DSP del mismo algoritmo en lenguaje C. El sistema con el DSP requería de 10,770,432 ciclos de reloj con una velocidad de 600MHz mientras la FPGA requería de 164,354 ciclos de reloj con una velocidad de 50MHz para realizar el mismo algoritmo. Es decir, con 50MHz la FPGA requería de 3.2ms para procesar una imagen de 128x128 pixeles, mientras el DSP requería de 17.9ms [51].

Se ha utilizado el lenguaje VHDL, para desarrollar el algoritmo que representa la TW Haar. El uso de un lenguaje como VHDL, permite la portabilidad simple de un algoritmo entre distintas marcas de FPGA. El diseño del algoritmo se realizó en Matlab, y la transcripción del código a VHDL se ha desarrollado a través de la herramienta de Matlab HDL coder. HDL coder demuestra eficiencia en la depuración de código, además posee características que brindan escalabilidad al sistema como trazabilidad de algoritmos, integración con herramientas de simulación y reportes de utilización de recursos.

El funcionamiento de la TW Haar embebida en la FPGA, se ha validado, a través de la herramienta de verificación FIL de Matlab, debido a que FIL, permite la comunicación de datos entre Simulink, MATLAB y la FPGA con una amplia gama de bloques que permiten el modelamiento del sistema. El enfoque FIL facilita la exploración de un algoritmo de manera eficiente.

El empleo del lenguaje VHDL junto con el conocimiento de la arquitectura wavelet , da la capacidad al programador de controlar con mayor precisión la entidad que desea programar, en comparación con otros tipos de software de diseño como Simulink de Matlab, LabView, etc, que pueden generar líneas de programación adicional, que se traducen en desperdicio de recursos.

6. REFERENCIAS.

[1] D. Valencia, D. Orejuela, J. Salazar, and J. Valencia, “Comparison Analysis Between Rigrsure , Sqtwolog , Heursure and Minimaxi Techniques Using Hard and Soft Thresholding Methods,” pp. 2–6.

[2] C. V. Lopez Torres, “Estudio comparativo entre tipos de transformada wavelet para su uso en reconstrucción tridimensional,” 2012.

[3] Y. T. Qassim, T. R. H. Cutmore, and D. D. Rowlands, “Optimized FPGA based continuous wavelet transform,” *Comput. Electr. Eng.*, vol. 49, pp. 84–94, 2014.

[4] P. S. Kumar, R. Arumuganathan, K. Sivakumar, and C. Vimal, “Removal of artifacts from EEG signals using adaptive filter through wavelet transform,” *2008 9th Int. Conf. Signal Process.*, pp. 2138–2141, 2008.

[5] E. Aguiirre, “Filtros de kalman robustos para el pre procesamiento de senales empleadas en interfaces ˜ cerebro-maquina,” *Eff. Br. mindfulness Interv. acute pain Exp. An Exam. Individ. Differ.*, vol. 1, 2015.

[6] C.-L. Liu, “A Tutorial of the Wavelet Transform,” *History*, pp. 1–72, 2010.

[7] E. J. Mitchell and B. Eng, “A Machine Learning Framework for Automatic Human Activity Classification from Wearable Sensors,” Dublin City University, 2014.

[8] J. Chilo and T. Lindblad, “Hardware Implementation of 1D Wavelet Transform on an FPGA for Infrasound Signal Classification,” *IEEE Trans. Nucl. Sci.*, vol. 55, no. 1, pp. 9–13, 2008.

[9] S. Rein and M. Reisslein, “Low-Memory Wavelet Transforms for Wireless Sensor Networks: A Tutorial,” *IEEE Commun. Surv. Tutorials*, vol. 13, no. 2, pp. 291–307, 2011.

[10] K. L. V. Iyer, X. Lu, Y. Usama, V. Ramakrishnan, and N. C. Kar, “A twofold Daubechies-wavelet-based module for fault detection and voltage regulation in SEIGs for distributed wind power generation,” *IEEE Trans. Ind. Electron.*, vol. 60, no. 4, pp. 1638–1651, 2013.

[11] F. Canbay, V. E. Levent, G. Serbes, S. Goren, and N. Aydin, “Field Programmable Gate Arrays Implementation of Dual Tree Complex Wavelet Transform,” pp. 6026–6029, 2015.

[12] M. Schimmack, S. Nguyen, and P. Mercorelli, “Implemented Wavelet Packet Tree based Denoising Algorithm in Bus Signals of a Wearable Sensorarray,” *J. Phys. Conf. Ser.*, vol. 659, p. 12021, 2015.

[13] S. W. Chen and Y. H. Chen, “Hardware design and implementation of a wavelet de-noising procedure for medical signal preprocessing,” *Sensors (Switzerland)*, vol. 15, no. 10, pp. 26396–26414, 2015.

[14] P. Kumar, “Generating, optimizing and verifying HDL Code with Matlab and Simulink,” *Mathworks*, 2012. .

[15] E. P. Serrano, “Introducción a la transformada wavelet y sus aplicaciones al procesamiento de señales de emisión acústica.” [Online]. Available: http://www.cnea.gov.ar/cac/glea/trabajos/serrano.pdf.

[16] E. Hernández, “Matemáticas de las señales.” [Online]. Available: https://www.uam.es/personal_pdi/ciencias/ehernan/Otros/Senyales-v3.pdf.

[17] P. Márquez, “Procesamiento digital de señales mediante la teoría de wavelets,” Universidad Pontificia Comillas, 2013.

[18] D. Mendlovic, Z. Zalevsky, D. Mas, J. Garcı, and C. Ferreira, “Fractional wavelet transform,” vol. 36, no. 20, pp. 4801–4806, 1997.

[19] E. Yamunaqué, “Aplicación de la transformada wavelet a señal de baja potencia en entorno de Matlab,” Universidad de Piura, 2016.

[20] A. Rodríguez, “Análisis de la señal EGG (Electrocardiograma), reconociendo las ondas P y T en el complejo QRS usando la transformada wavelet,” Universidad Pontificia Comillas.

[21] E. Gómez, G. Aponte, and D. Silva, “Selección de una wavelet madre para el análisis frecuencial de señales eléctricas transitorias.,” *Ingeniare. Rev. Chil. Ing.*, vol. 21, no. 2, pp. 262–270, 2013.

[22] A. Bhardwaj and R. Ali, “Image compression using modified fast haar wavelet transform,” *World Appl. Sci. J.*, vol. 7, no. 5, pp. 647–653, 2009.

[23] S. . Tamboli and V. . Udupi, “Image compression using haar wavelet transform,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 2, no. 8, pp. 3166–3170, 2013.

[24] T. Varma, V. Chitre, and D. Patil, “The haar wavelet and the biorthogonal wavelet trransforms of an image,” *Int. J. Eng. Res. Appl.*, pp. 268–291, 2012.

[25] A. Safari and Y. Kong, “Performance comparison of orthogonal and biorthogonal wavelets using technology libraries,” *2013 13th Int. Symp. Commun. Inf. Technol.*, no. March, pp. 325–329, 2013.

[26] D. Yadav and V. Dubey, “Performance evaluation of haar wavelet for image inpainting,” *Int. J. Adv. Eng. Res. Sci.*, vol. 1, no. 4, pp. 1–5, 2014.

[27] C. Stolojescu, I. Railean, S. Moga, and A. Isar, “Comparison of wavelet families with application to WiMAX traffic forecasting,” *Proc. Int. Conf. Optim. Electr. Electron. Equipment, OPTIM*, pp. 932–937, 2010.

[28] Sai Lakshmi Bhamidipati, Sai Sudha Mindagudla, Harsha Vardhan Devalla, Hima Sagar Goodi, and Hemanth Nag, “Analysis of different discrete wavelet transform basis functions in speech signal compression,” *IOSR J. VLSI Signal Process.*, vol. 4, no. 1, pp. 34–38, 2014.

[29] V. V. Vermehren and H. M. de Oliveira, “Close expressions for meyer wavelet and scale function,” vol. 4, no. 1, pp. 31–35, 2015.

[30] A. Abbas and T. D. Tran, “Rational coefficient dual-tree complex wavelet

transform: Design and implementation," *IEEE Trans. Signal Process.*, vol. 56, no. 8 I, pp. 3523–3534, 2008.

[31] I. W. Selesnick, R. G. Baraniuk, and N. C. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Process. Mag.*, vol. 22, no. 6, pp. 123–151, 2005.

[32] K. J. Priya and R. S. Rajesh, "Local Statistical Features of Dual Tree Complex Wavelet Transform on Parallelogram Image Structure for Face Recognition with Single Sample," in *2010 International Conference on Recent Trends in Information, Telecommunication and Computing*, 2010, pp. 50–54.

[33] P. Hill, A. Achim, M. E. Al-mualla, and D. Bull, "Contrast Sensitivity of the Wavelet , Dual Tree Pyramid Transforms," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2739–2751, 2016.

[34] J. Joy, S. Peter, and N. John, "Denoising Using Soft Thresholding," *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.*, vol. 2, no. 3, pp. 2320–3765, 2013.

[35] R. Ferozepur, "Image Denoising Techniques-An Overview," *Int. J. Comput. Appl.*, vol. 86, no. 16, pp. 975–8887, 2014.

[36] S. Sudha, G. R. Suresh, and R. Sukanesh, "Wavelet Based Image Denoising Using Adaptive Thresholding," in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, 2007, pp. 296–300.

[37] A. Dixit and P. Sharma, "A Comparative Study of Wavelet Thresholding for Image Denoising," *I.J. Image, Graph. Signal Process.*, vol. 12, no. 12, pp. 39–46, 2014.

[38] A. K. Verma and N. Verma, "Performance analysis of wavelet thresholding methods in denoising of audio signals of some indian musical instruments," *Int. J. Eng. Sci. Technol.*, vol. 4, no. 5, pp. 0975–5462, 2012.

[39] N. K. Al-Qazzaz, S. Ali, S. A. Ahmad, M. S. Islam, and M. I. Ariff, "Selection of Mother Wavelets Thresholding Methods in Denoising Multi-channel EEG Signals during Working Memory Task," 2014.

[40] I. Review, "High Performance FPGA Architecture for Dual Mode Processor of Integer Haar Lifting-Based Wavelet Transform," no. September 2013, 2016.

[41] "Berlin Brain-Computer Interface (BBCI)." [Online]. Available: http://www.bbci.de/. [Accessed: 04-Apr-2016].

[42] "Wavelet Families and Properties - MATLAB & Simulink." [Online]. Available: http://www.mathworks.com/help/wavelet/gs/wavelet-families-and-properties.html. [Accessed: 04-Apr-2016].

[43] S. Acharya, H. Kabra, P. V Kasambe, and S. S. Rathod, "Performance Evaluation of an Integer Wavelet Transform for FPGA Implementation," pp. 1–5, 2015.

[44] J. Martínez and R. Castro, *Análisis de la teoría ondículas orientada a las*

aplicaciones en ingeniería eléctrica:Fundamentos. 2002.

[45] M. Üstündag, A. Sengür, M. Gökbulut, and F. Ata, “Performance comparison of wavelet thresholding techniques on weak ECG signal denoising,” *Przegląd Elektrotechniczny*, vol. 89, no. 5, pp. 63–66, 2013.

[46] Q. Huang, Y. Wang, S. Chang, and a M. Algorithm, “High-performance FPGA Implementation of Discrete Wavelet Transform for Image Processing,” pp. 6–9, 2011.

[47] N. K. Kalantari and P. Sen, “Removing the noise in Monte Carlo rendering with general image denoising algorithms,” vol. 32, no. 2, 2013.

[48] A. K. Verma and N. Verma, “Performance analysis of wavelet thresholding methods in denoising of audio signals of some indian musical instruments,” *Int. J. Eng. Sci. Technol.*, vol. 4, no. 5, pp. 0975–5462, 2012.

[49] G. Ciprian, T. Alin, O. Stefan, and B. Attila, “FPGA-based discrete wavelet transforms design using,” pp. 98–101, 2012.

[50] L. Yang, “An Improved Wavelet Filtering Algorithm and Its FPGA Implementation,” no. December, pp. 17–24, 2010.

[51] M. Shirvaikar and T. Bushnaq, “A comparison between DSP and FPGA platforms for real-time imaging applications,” *IS&T/SPIE Electron. Imaging*, vol. 7244, no. February, p. 724406, 2009.

7. APENDICE

7.1. Código en Matlab para HDL CODER

```
function [signal_out] = TWHAAR(signal_in,start)%%Declaración de entradas y
salidas
% WT HAAR
muestras = 32;
initval1 = int16(zeros(1,muestras));
media = int16(0);

persistent signal_out_memory contador signal_in_memory %%Declaración de registros
if isempty(signal_out_memory)%Inicialización de registros y arreglos
    signal_in_memory = initval1;
    signal_out_memory = initval1;
    contador  = uint8(1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%&&&&&&&&&&&&
signal_out_memory(1,contador) = signal_in;%Lectura de los datos
signal_out = int16(signal_in_memory(1,contador));%%Envio de los datos
if (start == true)
    contador = contador+1;%%Contador cantidad de datos
    if contador == muestras+1%%Si llegan 32 datos, realiza la TW con thresholding
        [CoefAprox,CoefDetail] = lwthaar(signal_out_memory,uint8(16)); %% Primer
nivel descom
         media = sum(CoefDetail,'native')+media;%Acumulacion de los datos
        [CoefAprox2,CoefDetail2] = lwthaar(CoefAprox,uint8(8));%% Segundo nivel
descom
        media = sum(CoefDetail2,'native')+media;
        [CoefAprox3,CoefDetail3] = lwthaar(CoefAprox2,uint8(4));%% Tercer nivel
descom
        media = sum(CoefDetail3,'native')+media;
        [CoefAprox4,CoefDetail4] = lwthaar(CoefAprox3,uint8(2));%% Cuarto nivel
descom
        media = sum(CoefDetail4,'native')+media;
        media = sum(CoefAprox4,'native')+media;%%Suma de todos los datos
        C = [CoefAprox4 CoefDetail4 CoefDetail3 CoefDetail2 CoefDetail];
        mediari=bitshift(media,-5);%;Calculo de la media aritmetica de los datos
        C = threshold(C,mediari);%%Modificación de los coeficientes con Sqtwolog,
y Hard thresholding
        CoefAprox4 = C(uint8(1):uint8(2));
        CoefDetail4 = C(uint8(3):uint8(4));
        CoefDetail3 = C(uint8(5):uint8(8));
        CoefDetail2 = C(uint8(9):uint8(16));
        CoefDetail = C(uint8(17):uint8(32));
```

```
        [recuperada4] = ilwthaar(CoefAprox4,CoefDetail4); %% Cuarto nivel recons
        [recuperada3] = ilwthaar(recuperada4,CoefDetail3) ;%% Tercer nivel recons
        [recuperada2] = ilwthaar(recuperada3,CoefDetail2); %% Segundo nivel
recons
        [signal_in_memory] = ilwthaar(recuperada2,CoefDetail); %% Primer nivel
recons
        contador = uint8(1);
    end
end
end
```

7.2. Código LWT Haar

```
function [CoefAprox,CoefDetail] = lwthaar(senal,lengths)
Impar = int16(zeros(1,lengths));
Par = int16(zeros(1,lengths));
n = (1:lengths);
Impar(n) = (senal(2*n-1));  % odd signals, separación impar (SPLIT)
Par(n) = (senal(2*n));  % even signals, separación par (SPLIT)
CoefDetail = ((Impar)-(Par)); %Creación de los coeficientes detallados (PREDICT)
CoefAprox = (Par+(bitshift(int16(CoefDetail),-1))); %Coeficientes aproximados
(UPDATE)
end
```

7.3. Código ILWT Haar

```
function [recuperada] = ilwthaar(CoefAprox,CoefDetail)
Par = CoefAprox-(bitshift(int16(CoefDetail),-1));%Creación de los componentes
pares (UNDO UPDATE)
Impar = (Par)+(CoefDetail);%Creación de los componentes impares (UNDO PREDICT)
recuperada = [(Impar);(Par)]; %(MERGE)
recuperada = (recuperada(:).');%%(MERGE)
end
```

7.4. Código etapa de thresholding

```
function CoefDetail = threshold(CoefDetail,umbral)
media = int16(zeros(1,32));
acumulador = int16(0);
for a = 1:32
```

```
    temporal = umbral-CoefDetail(a);
if (temporal)<0
     media(a) = CoefDetail(a)-int16(umbral);
else
    media(a) = int16(umbral)-CoefDetail(a);
end
acumulador = media(a)+acumulador;%%Desviación estandar absoluta
end
temporal =bitshift(acumulador,-5);% Calculo de la MAD
THR1 = bitshift(int16(temporal),1);%Umbral Sqtwolog
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:32
if ((CoefDetail(i))>=THR1 ||(-CoefDetail(i))>=THR1 )%%Umbralización dura
else
    CoefDetail(i) = int16(0);
end
end
end
```

7.5. Testbench del código en Matlab

```
clc

close all
clear all
load('data_set_IVa_aa.mat')%%Senal de pruebas
muestras = 32;%Cantidad de muestras por trama
senaloriginal = double(0.1*cnt(1:5000,1));%Obtención de un canal
senaloriginal = senaloriginal(1:muestras)';
senal = senaloriginal(1:muestras)
recuperada = zeros(1,muestras);
Fs=100;%Frecuencia de muestreo de la señal
t = 1:1:muestras;
tiempo = t.*1/Fs;
for ii=1:2*muestras
if (ii < muestras+1)
    recuperada(1,ii) = TWHAAR(senal(1,ii),true);%%Envio de datos
else
    recuperada(1, ii-muestras) = TWHAAR(senal(1, ii-muestras),true);%Recepción de
datos
end
end

figure('Name', [mfilename, '_plot']);
plot(tiempo,recuperada,'-r');
ylabel('Amplitud(uV)');
xlabel('Tiempo(s)');
```

```
title('Test bench HDL coder')
hold on
plot(tiempo,senaloriginal,'-b');
legend('Señal del EEG reconstruida','Señal del EEG original')
xlim([0 0.4])
grid on;
hold off
```

7.6. Código en VHDL generado por HDL coder

```
-- -------------------------------------------------------------
--
-- File Name: C:\PROYECTO FINAL\codegen\TWHAAR\hdlsrc\TWHAAR.vhd
-- Created: 2016-10-20 02:15:04
--
-- Generated by MATLAB 9.0, MATLAB Coder 3.1 and HDL Coder 3.8
--
--
--
-- -------------------------------------------------------------
-- Rate and Clocking Details
-- -------------------------------------------------------------
-- Design base rate: 1
--
--
-- Clock Enable  Sample Time
-- -------------------------------------------------------------
-- ce_out        1
-- -------------------------------------------------------------
--
--
-- Output Signal                 Clock Enable  Sample Time
-- -------------------------------------------------------------
-- signal_out                    ce_out        1
-- -------------------------------------------------------------
--
-- -------------------------------------------------------------



-- -------------------------------------------------------------
--
-- Module: TWHAAR
-- Source Path: TWHAAR
-- Hierarchy Level: 0
--
-- -------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.TWHAAR_pkg.ALL;
```

```
ENTITY TWHAAR IS
  PORT( clk                               :   IN    std_logic;
        reset                             :   IN    std_logic;
        clk_enable                        :   IN    std_logic;
        signal_in                         :   IN    signed(15 DOWNTO 0);
-- int16
        start                             :   IN    std_logic;
        ce_out                            :   OUT   std_logic;
        signal_out                        :   OUT   signed(15 DOWNTO 0)
-- int16
        );
END TWHAAR;


ARCHITECTURE rtl OF TWHAAR IS

  -- Signals
  SIGNAL enb                              : std_logic;
  SIGNAL TWHAAR_signal_out_memory         : vector_of_signed16(0 TO 31);
-- int16 [32]
  SIGNAL TWHAAR_contador                  : unsigned(7 DOWNTO 0);  --
uint8
  SIGNAL TWHAAR_signal_in_memory          : vector_of_signed16(0 TO 31);
-- int16 [32]
  SIGNAL TWHAAR_signal_out_memory_next    : vector_of_signed16(0 TO 31);
-- int16 [32]
  SIGNAL TWHAAR_contador_next             : unsigned(7 DOWNTO 0);  --
uint8
  SIGNAL TWHAAR_signal_in_memory_next     : vector_of_signed16(0 TO 31);
-- int16 [32]

BEGIN
  enb <= clk_enable;

  TWHAAR_1_process : PROCESS (clk)
    VARIABLE t_1 : INTEGER;
  BEGIN
    IF rising_edge(clk) THEN
      IF reset = '1' THEN

        FOR t_1 IN 0 TO 31 LOOP
          TWHAAR_signal_in_memory(t_1) <= to_signed(16#0000#, 16);
          TWHAAR_signal_out_memory(t_1) <= to_signed(16#0000#, 16);
        END LOOP;

        TWHAAR_contador <= to_unsigned(16#01#, 8);
      ELSIF enb = '1' THEN
        TWHAAR_contador <= TWHAAR_contador_next;

        FOR t_0 IN 0 TO 31 LOOP
          TWHAAR_signal_out_memory(t_0) <=
TWHAAR_signal_out_memory_next(t_0);
```

```
          TWHAAR_signal_in_memory(t_0) <=
TWHAAR_signal_in_memory_next(t_0);
        END LOOP;

      END IF;
    END IF;
  END PROCESS TWHAAR_1_process;

  TWHAAR_1_output : PROCESS (signal_in, start, TWHAAR_signal_out_memory,
TWHAAR_contador,
       TWHAAR_signal_in_memory)
    VARIABLE media : signed(15 DOWNTO 0);
    VARIABLE CoefDetail : vector_of_signed16(0 TO 15);
    VARIABLE CoefDetail2 : vector_of_signed16(0 TO 7);
    VARIABLE CoefDetail3 : vector_of_signed16(0 TO 3);
    VARIABLE CoefAprox4 : vector_of_signed16(0 TO 1);
    VARIABLE CoefDetail4 : vector_of_signed16(0 TO 1);
    VARIABLE C : vector_of_signed16(0 TO 31);
    VARIABLE mediari : signed(15 DOWNTO 0);
    VARIABLE recuperada4 : vector_of_signed16(0 TO 3);
    VARIABLE recuperada3 : vector_of_signed16(0 TO 7);
    VARIABLE recuperada2 : vector_of_signed16(0 TO 15);
    VARIABLE CoefAprox : vector_of_signed16(0 TO 15);
    VARIABLE coefdetail_0 : vector_of_signed16(0 TO 15);
    VARIABLE CoefAprox2 : vector_of_signed16(0 TO 7);
    VARIABLE coefdetail2_0 : vector_of_signed16(0 TO 7);
    VARIABLE CoefAprox3 : vector_of_signed16(0 TO 3);
    VARIABLE coefdetail3_0 : vector_of_signed16(0 TO 3);
    VARIABLE coefaprox4_0 : vector_of_signed16(0 TO 1);
    VARIABLE coefdetail4_0 : vector_of_signed16(0 TO 1);
    VARIABLE Impar : vector_of_signed16(0 TO 15);
    VARIABLE Par : vector_of_signed16(0 TO 15);
    VARIABLE c_0 : vector_of_signed16(0 TO 15);
    VARIABLE a1 : signed(15 DOWNTO 0);
    VARIABLE impar_0 : vector_of_signed16(0 TO 7);
    VARIABLE par_0 : vector_of_signed16(0 TO 7);
    VARIABLE c_1 : vector_of_signed16(0 TO 7);
    VARIABLE a1_0 : signed(15 DOWNTO 0);
    VARIABLE y : signed(15 DOWNTO 0);
    VARIABLE impar_1 : vector_of_signed16(0 TO 3);
    VARIABLE par_1 : vector_of_signed16(0 TO 3);
    VARIABLE c_2 : vector_of_signed16(0 TO 3);
    VARIABLE a1_1 : signed(15 DOWNTO 0);
    VARIABLE y_0 : signed(15 DOWNTO 0);
    VARIABLE impar_2 : vector_of_signed16(0 TO 1);
    VARIABLE par_2 : vector_of_signed16(0 TO 1);
    VARIABLE c_3 : vector_of_signed16(0 TO 1);
    VARIABLE a1_2 : signed(15 DOWNTO 0);
    VARIABLE promedio : vector_of_signed16(0 TO 31);
    VARIABLE acumulador : signed(15 DOWNTO 0);
    VARIABLE temporal : signed(15 DOWNTO 0);
    VARIABLE temporal2 : signed(15 DOWNTO 0);
    VARIABLE THR1 : signed(15 DOWNTO 0);
    VARIABLE par_3 : vector_of_signed16(0 TO 1);
```

```
VARIABLE impar_3 : vector_of_signed16(0 TO 1);
VARIABLE recuperada : matrix_of_signed16(0 TO 1, 0 TO 1);
VARIABLE c_4 : vector_of_signed16(0 TO 1);
VARIABLE a1_3 : signed(15 DOWNTO 0);
VARIABLE par_4 : vector_of_signed16(0 TO 3);
VARIABLE impar_4 : vector_of_signed16(0 TO 3);
VARIABLE recuperada_0 : matrix_of_signed16(0 TO 1, 0 TO 3);
VARIABLE c_5 : vector_of_signed16(0 TO 3);
VARIABLE a1_4 : signed(15 DOWNTO 0);
VARIABLE par_5 : vector_of_signed16(0 TO 7);
VARIABLE impar_5 : vector_of_signed16(0 TO 7);
VARIABLE recuperada_1 : matrix_of_signed16(0 TO 1, 0 TO 7);
VARIABLE c_6 : vector_of_signed16(0 TO 7);
VARIABLE a1_5 : signed(15 DOWNTO 0);
VARIABLE par_6 : vector_of_signed16(0 TO 15);
VARIABLE impar_6 : vector_of_signed16(0 TO 15);
VARIABLE recuperada_2 : matrix_of_signed16(0 TO 1, 0 TO 15);
VARIABLE c_7 : vector_of_signed16(0 TO 15);
VARIABLE a1_6 : signed(15 DOWNTO 0);
VARIABLE signal_out_memory_temp : vector_of_signed16(0 TO 31);
VARIABLE contador_temp : unsigned(7 DOWNTO 0);
VARIABLE sub_cast : signed(31 DOWNTO 0);
VARIABLE sub_cast_0 : signed(31 DOWNTO 0);
VARIABLE add_temp : unsigned(8 DOWNTO 0);
VARIABLE cast : vector_of_signed64(0 TO 15);
VARIABLE add_cast : vector_of_signed64(0 TO 15);
VARIABLE add_temp_0 : vector_of_signed17(0 TO 15);
VARIABLE sub_temp : vector_of_signed17(0 TO 15);
VARIABLE a1_7 : vector_of_signed32(0 TO 15);
VARIABLE add_temp_1 : vector_of_signed17(0 TO 14);
VARIABLE cast_0 : vector_of_signed64(0 TO 7);
VARIABLE add_cast_0 : vector_of_signed64(0 TO 7);
VARIABLE add_temp_2 : vector_of_signed17(0 TO 7);
VARIABLE sub_temp_0 : vector_of_signed17(0 TO 7);
VARIABLE a1_8 : vector_of_signed32(0 TO 7);
VARIABLE add_temp_3 : signed(16 DOWNTO 0);
VARIABLE add_temp_4 : vector_of_signed17(0 TO 6);
VARIABLE cast_1 : vector_of_signed64(0 TO 3);
VARIABLE add_cast_1 : vector_of_signed64(0 TO 3);
VARIABLE add_temp_5 : vector_of_signed17(0 TO 3);
VARIABLE sub_temp_1 : vector_of_signed17(0 TO 3);
VARIABLE a1_9 : vector_of_signed32(0 TO 3);
VARIABLE add_temp_6 : signed(16 DOWNTO 0);
VARIABLE add_temp_7 : vector_of_signed17(0 TO 2);
VARIABLE cast_2 : vector_of_signed64(0 TO 1);
VARIABLE add_cast_2 : vector_of_signed64(0 TO 1);
VARIABLE sub_temp_2 : vector_of_signed17(0 TO 1);
VARIABLE add_temp_8 : signed(16 DOWNTO 0);
VARIABLE add_cast_3 : signed(15 DOWNTO 0);
VARIABLE add_cast_4 : signed(16 DOWNTO 0);
VARIABLE add_temp_9 : signed(16 DOWNTO 0);
VARIABLE a1_10 : vector_of_signed32(0 TO 1);
VARIABLE add_temp_10 : signed(16 DOWNTO 0);
VARIABLE add_cast_5 : signed(15 DOWNTO 0);
```

```
  VARIABLE add_cast_6 : signed(16 DOWNTO 0);
  VARIABLE add_temp_11 : signed(16 DOWNTO 0);
  VARIABLE add_temp_12 : vector_of_signed17(0 TO 1);
  VARIABLE media_0 : signed(31 DOWNTO 0);
  VARIABLE acumulador_0 : signed(31 DOWNTO 0);
  VARIABLE sub_temp_3 : vector_of_signed17(0 TO 31);
  VARIABLE temporal2_0 : signed(31 DOWNTO 0);
  VARIABLE sub_temp_4 : vector_of_signed17(0 TO 31);
  VARIABLE sub_temp_5 : vector_of_signed17(0 TO 31);
  VARIABLE add_temp_13 : vector_of_signed17(0 TO 31);
  VARIABLE c_8 : vector_of_signed17(0 TO 31);
  VARIABLE cast_3 : vector_of_signed16(0 TO 31);
  VARIABLE a1_11 : vector_of_signed32(0 TO 1);
  VARIABLE sub_temp_6 : vector_of_signed17(0 TO 1);
  VARIABLE add_temp_14 : vector_of_signed17(0 TO 1);
  VARIABLE a1_12 : vector_of_signed32(0 TO 3);
  VARIABLE sub_temp_7 : vector_of_signed17(0 TO 3);
  VARIABLE add_temp_15 : vector_of_signed17(0 TO 3);
  VARIABLE a1_13 : vector_of_signed32(0 TO 7);
  VARIABLE sub_temp_8 : vector_of_signed17(0 TO 7);
  VARIABLE add_temp_16 : vector_of_signed17(0 TO 7);
  VARIABLE a1_14 : vector_of_signed32(0 TO 15);
  VARIABLE sub_temp_9 : vector_of_signed17(0 TO 15);
  VARIABLE add_temp_17 : vector_of_signed17(0 TO 15);
  VARIABLE cast_temp : unsigned(7 DOWNTO 0);
  VARIABLE cast_temp_0 : unsigned(15 DOWNTO 0);
  VARIABLE mul_temp : unsigned(15 DOWNTO 0);
  VARIABLE oldIdx : INTEGER;
  VARIABLE cast_temp_1 : unsigned(7 DOWNTO 0);
  VARIABLE cast_temp_2 : unsigned(15 DOWNTO 0);
  VARIABLE mul_temp_0 : unsigned(15 DOWNTO 0);
  VARIABLE oldidx_0 : INTEGER;
  VARIABLE cast_temp_3 : unsigned(7 DOWNTO 0);
  VARIABLE cast_temp_4 : unsigned(15 DOWNTO 0);
  VARIABLE mul_temp_1 : unsigned(15 DOWNTO 0);
  VARIABLE oldidx_1 : INTEGER;
  VARIABLE cast_temp_5 : unsigned(7 DOWNTO 0);
  VARIABLE cast_temp_6 : unsigned(15 DOWNTO 0);
  VARIABLE mul_temp_2 : unsigned(15 DOWNTO 0);
  VARIABLE oldidx_2 : INTEGER;
BEGIN
  contador_temp := TWHAAR_contador;

  FOR t_0 IN 0 TO 31 LOOP
    signal_out_memory_temp(t_0) := TWHAAR_signal_out_memory(t_0);
    TWHAAR_signal_in_memory_next(t_0) <= TWHAAR_signal_in_memory(t_0);
  END LOOP;

  --%Declaración de entradas y salidas
  -- WT HAAR
  --'TWHAAR:3' muestras = 32;
  --'TWHAAR:4' initval1 = int16(zeros(1,muestras));
  --'TWHAAR:5' media = int16(0);
  --%Declaración de registros
```

```
    --'TWHAAR:8' if isempty(signal_out_memory)
    --%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%&&&&&&&&&&&&
    --'TWHAAR:14' signal_out_memory(1,contador) = signal_in;
    sub_cast := signed(resize(TWHAAR_contador, 32));
    signal_out_memory_temp(to_integer(sub_cast - 1)) := signal_in;
    --Lectura de los datos
    --'TWHAAR:15' signal_out = int16(signal_in_memory(1,contador));
    sub_cast_0 := signed(resize(TWHAAR_contador, 32));
    signal_out <= TWHAAR_signal_in_memory(to_integer(sub_cast_0 - 1));
    --%Envio de los datos
    --'TWHAAR:16' if (start == true)
    IF start = '1' THEN
      --'TWHAAR:17' contador = contador+1;
      add_temp := resize(TWHAAR_contador, 9) + to_unsigned(16#001#, 9);
      IF add_temp(8) /= '0' THEN
        contador_temp := "11111111";
      ELSE
        contador_temp := add_temp(7 DOWNTO 0);
      END IF;
      --%Contador cantidad de datos
      --'TWHAAR:18' if contador == muestras+1
      IF contador_temp = to_unsigned(16#00000021#, 8) THEN
        --%Si llegan 32 datos, realiza la TW con thresholding
        --'TWHAAR:19' [CoefAprox,CoefDetail] =
lwthaar(signal_out_memory,uint8(16));
        --'lwthaar:2' Impar = int16(zeros(1,lengths));
        --'lwthaar:3' Par = int16(zeros(1,lengths));
        --'lwthaar:4' n = (1:lengths);
        --'lwthaar:5' Impar(n) = (senal(2*n-1));
        -- odd signals, separación impar (SPLIT)
        --'lwthaar:6' Par(n) = (senal(2*n));
        -- even signals, separación par (SPLIT)
        --'lwthaar:7' CoefDetail = ((Impar)-(Par));
        --Creación de los coeficientes detallados (PREDICT)
        --'lwthaar:8' CoefAprox = (Par+(bitshift(int16(CoefDetail),-1)));

        FOR m_0 IN 0 TO 15 LOOP
          cast(m_0) := resize(to_signed(m_0, 32) & '0', 64);
          Impar(m_0) :=
signal_out_memory_temp(to_integer(resize(cast(m_0), 31)));
          add_cast(m_0) := resize(to_signed(m_0, 32) & '0', 64);
          Par(m_0) := signal_out_memory_temp(to_integer(1 +
resize(add_cast(m_0), 32)));
          sub_temp(m_0) := resize(Impar(m_0), 17) - resize(Par(m_0), 17);
          IF (sub_temp(m_0)(16) = '0') AND (sub_temp(m_0)(15) /= '0')
THEN
            coefdetail_0(m_0) := X"7FFF";
          ELSIF (sub_temp(m_0)(16) = '1') AND (sub_temp(m_0)(15) /= '1')
THEN
            coefdetail_0(m_0) := X"8000";
          ELSE
            coefdetail_0(m_0) := sub_temp(m_0)(15 DOWNTO 0);
          END IF;
          a1 := coefdetail_0(m_0);
```

```
          a1_7(m_0) := SHIFT_RIGHT(resize(a1, 32) , 1);
          c_0(m_0) := a1_7(m_0)(15 DOWNTO 0);
        END LOOP;


        FOR t_1 IN 0 TO 15 LOOP
          add_temp_0(t_1) := resize(Par(t_1), 17) + resize(c_0(t_1), 17);
          IF (add_temp_0(t_1)(16) = '0') AND (add_temp_0(t_1)(15) /= '0')
THEN
            CoefAprox(t_1) := X"7FFF";
          ELSIF (add_temp_0(t_1)(16) = '1') AND (add_temp_0(t_1)(15) /=
'1') THEN
            CoefAprox(t_1) := X"8000";
          ELSE
            CoefAprox(t_1) := add_temp_0(t_1)(15 DOWNTO 0);
          END IF;
        END LOOP;

        --Coeficientes aproximados (UPDATE)
        --% Primer nivel descom
        --'TWHAAR:20' media = sum(CoefDetail,'native')+media;
        media := coefdetail_0(0);

        FOR k IN 0 TO 14 LOOP
          add_temp_1(k) := resize(media, 17) + resize(coefdetail_0(k +
1), 17);
          IF (add_temp_1(k)(16) = '0') AND (add_temp_1(k)(15) /= '0')
THEN
            media := X"7FFF";
          ELSIF (add_temp_1(k)(16) = '1') AND (add_temp_1(k)(15) /= '1')
THEN
            media := X"8000";
          ELSE
            media := add_temp_1(k)(15 DOWNTO 0);
          END IF;
        END LOOP;

        --Acumulacion de los datos
        --'TWHAAR:21' [CoefAprox2,CoefDetail2] =
lwthaar(CoefAprox,uint8(8));
        --'lwthaar:2' Impar = int16(zeros(1,lengths));
        --'lwthaar:3' Par = int16(zeros(1,lengths));
        --'lwthaar:4' n = (1:lengths);
        --'lwthaar:5' Impar(n) = (senal(2*n-1));
        -- odd signals, separación impar (SPLIT)
        --'lwthaar:6' Par(n) = (senal(2*n));
        -- even signals, separación par (SPLIT)
        --'lwthaar:7' CoefDetail = ((Impar)-(Par));
        --Creación de los coeficientes detallados (PREDICT)
        --'lwthaar:8' CoefAprox = (Par+(bitshift(int16(CoefDetail),-1)));

        FOR m_1 IN 0 TO 7 LOOP
          cast_0(m_1) := resize(to_signed(m_1, 32) & '0', 64);
```

```
          impar_0(m_1) := CoefAprox(to_integer(resize(cast_0(m_1), 31)));
          add_cast_0(m_1) := resize(to_signed(m_1, 32) & '0', 64);
          par_0(m_1) := CoefAprox(to_integer(1 + resize(add_cast_0(m_1),
32)));
          sub_temp_0(m_1) := resize(impar_0(m_1), 17) -
resize(par_0(m_1), 17);
          IF (sub_temp_0(m_1)(16) = '0') AND (sub_temp_0(m_1)(15) /= '0')
THEN
            coefdetail2_0(m_1) := X"7FFF";
          ELSIF (sub_temp_0(m_1)(16) = '1') AND (sub_temp_0(m_1)(15) /=
'1') THEN
            coefdetail2_0(m_1) := X"8000";
          ELSE
            coefdetail2_0(m_1) := sub_temp_0(m_1)(15 DOWNTO 0);
          END IF;
          a1_0 := coefdetail2_0(m_1);
          a1_8(m_1) := SHIFT_RIGHT(resize(a1_0, 32) , 1);
          c_1(m_1) := a1_8(m_1)(15 DOWNTO 0);
        END LOOP;


        FOR t_2 IN 0 TO 7 LOOP
          add_temp_2(t_2) := resize(par_0(t_2), 17) + resize(c_1(t_2),
17);
          IF (add_temp_2(t_2)(16) = '0') AND (add_temp_2(t_2)(15) /= '0')
THEN
            CoefAprox2(t_2) := X"7FFF";
          ELSIF (add_temp_2(t_2)(16) = '1') AND (add_temp_2(t_2)(15) /=
'1') THEN
            CoefAprox2(t_2) := X"8000";
          ELSE
            CoefAprox2(t_2) := add_temp_2(t_2)(15 DOWNTO 0);
          END IF;
        END LOOP;

        --Coeficientes aproximados (UPDATE)
        --% Segundo nivel descom
        --'TWHAAR:22' media = sum(CoefDetail2,'native')+media;
        y := coefdetail2_0(0);

        FOR k_0 IN 0 TO 6 LOOP
          add_temp_4(k_0) := resize(y, 17) + resize(coefdetail2_0(k_0 +
1), 17);
          IF (add_temp_4(k_0)(16) = '0') AND (add_temp_4(k_0)(15) /= '0')
THEN
            y := X"7FFF";
          ELSIF (add_temp_4(k_0)(16) = '1') AND (add_temp_4(k_0)(15) /=
'1') THEN
            y := X"8000";
          ELSE
            y := add_temp_4(k_0)(15 DOWNTO 0);
          END IF;
        END LOOP;
```

```
        add_temp_3 := resize(y, 17) + resize(media, 17);
        IF (add_temp_3(16) = '0') AND (add_temp_3(15) /= '0') THEN
          media := X"7FFF";
        ELSIF (add_temp_3(16) = '1') AND (add_temp_3(15) /= '1') THEN
          media := X"8000";
        ELSE
          media := add_temp_3(15 DOWNTO 0);
        END IF;
        --'TWHAAR:23' [CoefAprox3,CoefDetail3] =
lwthaar(CoefAprox2,uint8(4));
        --'lwthaar:2' Impar = int16(zeros(1,lengths));
        --'lwthaar:3' Par = int16(zeros(1,lengths));
        --'lwthaar:4' n = (1:lengths);
        --'lwthaar:5' Impar(n) = (senal(2*n-1));
        -- odd signals, separación impar (SPLIT)
        --'lwthaar:6' Par(n) = (senal(2*n));
        -- even signals, separación par (SPLIT)
        --'lwthaar:7' CoefDetail = ((Impar)-(Par));
        --Creación de los coeficientes detallados (PREDICT)
        --'lwthaar:8' CoefAprox = (Par+(bitshift(int16(CoefDetail),-1)));

        FOR m_2 IN 0 TO 3 LOOP
          cast_1(m_2) := resize(to_signed(m_2, 32) & '0', 64);
          impar_1(m_2) := CoefAprox2(to_integer(resize(cast_1(m_2),
31)));
          add_cast_1(m_2) := resize(to_signed(m_2, 32) & '0', 64);
          par_1(m_2) := CoefAprox2(to_integer(1 + resize(add_cast_1(m_2),
32)));
          sub_temp_1(m_2) := resize(impar_1(m_2), 17) -
resize(par_1(m_2), 17);
          IF (sub_temp_1(m_2)(16) = '0') AND (sub_temp_1(m_2)(15) /= '0')
THEN
            coefdetail3_0(m_2) := X"7FFF";
          ELSIF (sub_temp_1(m_2)(16) = '1') AND (sub_temp_1(m_2)(15) /=
'1') THEN
            coefdetail3_0(m_2) := X"8000";
          ELSE
            coefdetail3_0(m_2) := sub_temp_1(m_2)(15 DOWNTO 0);
          END IF;
          a1_1 := coefdetail3_0(m_2);
          a1_9(m_2) := SHIFT_RIGHT(resize(a1_1, 32) , 1);
          c_2(m_2) := a1_9(m_2)(15 DOWNTO 0);
        END LOOP;


        FOR t_3 IN 0 TO 3 LOOP
          add_temp_5(t_3) := resize(par_1(t_3), 17) + resize(c_2(t_3),
17);
          IF (add_temp_5(t_3)(16) = '0') AND (add_temp_5(t_3)(15) /= '0')
THEN
            CoefAprox3(t_3) := X"7FFF";
          ELSIF (add_temp_5(t_3)(16) = '1') AND (add_temp_5(t_3)(15) /=
'1') THEN
```

```
            CoefAprox3(t_3) := X"8000";
          ELSE
            CoefAprox3(t_3) := add_temp_5(t_3)(15 DOWNTO 0);
          END IF;
        END LOOP;

        --Coeficientes aproximados (UPDATE)
        --% Tercer nivel descom
        --'TWHAAR:24' media = sum(CoefDetail3,'native')+media;
        y_0 := coefdetail3_0(0);

        FOR k_1 IN 0 TO 2 LOOP
          add_temp_7(k_1) := resize(y_0, 17) + resize(coefdetail3_0(k_1 +
1), 17);
          IF (add_temp_7(k_1)(16) = '0') AND (add_temp_7(k_1)(15) /= '0')
THEN
            y_0 := X"7FFF";
          ELSIF (add_temp_7(k_1)(16) = '1') AND (add_temp_7(k_1)(15) /=
'1') THEN
            y_0 := X"8000";
          ELSE
            y_0 := add_temp_7(k_1)(15 DOWNTO 0);
          END IF;
        END LOOP;

        add_temp_6 := resize(y_0, 17) + resize(media, 17);
        IF (add_temp_6(16) = '0') AND (add_temp_6(15) /= '0') THEN
          media := X"7FFF";
        ELSIF (add_temp_6(16) = '1') AND (add_temp_6(15) /= '1') THEN
          media := X"8000";
        ELSE
          media := add_temp_6(15 DOWNTO 0);
        END IF;
        --'TWHAAR:25' [CoefAprox4,CoefDetail4] =
lwthaar(CoefAprox3,uint8(2));
        --'lwthaar:2' Impar = int16(zeros(1,lengths));
        --'lwthaar:3' Par = int16(zeros(1,lengths));
        --'lwthaar:4' n = (1:lengths);
        --'lwthaar:5' Impar(n) = (senal(2*n-1));
        -- odd signals, separación impar (SPLIT)
        --'lwthaar:6' Par(n) = (senal(2*n));
        -- even signals, separación par (SPLIT)
        --'lwthaar:7' CoefDetail = ((Impar)-(Par));
        --Creación de los coeficientes detallados (PREDICT)
        --'lwthaar:8' CoefAprox = (Par+(bitshift(int16(CoefDetail),-1)));

        FOR m_3 IN 0 TO 1 LOOP
          cast_2(m_3) := resize(to_signed(m_3, 32) & '0', 64);
          impar_2(m_3) := CoefAprox3(to_integer(resize(cast_2(m_3),
31)));
          add_cast_2(m_3) := resize(to_signed(m_3, 32) & '0', 64);
          par_2(m_3) := CoefAprox3(to_integer(1 + resize(add_cast_2(m_3),
32)));
```

```
          sub_temp_2(m_3) := resize(impar_2(m_3), 17) -
resize(par_2(m_3), 17);
          IF (sub_temp_2(m_3)(16) = '0') AND (sub_temp_2(m_3)(15) /= '0')
THEN
            coefdetail4_0(m_3) := X"7FFF";
          ELSIF (sub_temp_2(m_3)(16) = '1') AND (sub_temp_2(m_3)(15) /=
'1') THEN
            coefdetail4_0(m_3) := X"8000";
          ELSE
            coefdetail4_0(m_3) := sub_temp_2(m_3)(15 DOWNTO 0);
          END IF;
          a1_2 := coefdetail4_0(m_3);
          a1_10(m_3) := SHIFT_RIGHT(resize(a1_2, 32) , 1);
          c_3(m_3) := a1_10(m_3)(15 DOWNTO 0);
        END LOOP;

        --Coeficientes aproximados (UPDATE)
        --% Cuarto nivel descom
        --'TWHAAR:26' media = sum(CoefDetail4,'native')+media;
        add_temp_8 := resize(coefdetail4_0(0), 17) +
resize(coefdetail4_0(1), 17);
        IF (add_temp_8(16) = '0') AND (add_temp_8(15) /= '0') THEN
          add_cast_3 := X"7FFF";
        ELSIF (add_temp_8(16) = '1') AND (add_temp_8(15) /= '1') THEN
          add_cast_3 := X"8000";
        ELSE
          add_cast_3 := add_temp_8(15 DOWNTO 0);
        END IF;
        add_cast_4 := resize(add_cast_3, 17);
        add_temp_9 := add_cast_4 + resize(media, 17);
        IF (add_temp_9(16) = '0') AND (add_temp_9(15) /= '0') THEN
          media := X"7FFF";
        ELSIF (add_temp_9(16) = '1') AND (add_temp_9(15) /= '1') THEN
          media := X"8000";
        ELSE
          media := add_temp_9(15 DOWNTO 0);
        END IF;
        --%Suma de todos los datos
        --'TWHAAR:27' media = sum(CoefAprox4,'native')+media;

        FOR t_4 IN 0 TO 1 LOOP
          add_temp_12(t_4) := resize(par_2(t_4), 17) + resize(c_3(t_4),
17);
          IF (add_temp_12(t_4)(16) = '0') AND (add_temp_12(t_4)(15) /=
'0') THEN
            coefaprox4_0(t_4) := X"7FFF";
          ELSIF (add_temp_12(t_4)(16) = '1') AND (add_temp_12(t_4)(15) /=
'1') THEN
            coefaprox4_0(t_4) := X"8000";
          ELSE
            coefaprox4_0(t_4) := add_temp_12(t_4)(15 DOWNTO 0);
          END IF;
          C(t_4) := coefaprox4_0(t_4);
          C(t_4 + 2) := coefdetail4_0(t_4);
```

```
        END LOOP;

        add_temp_10 := resize(coefaprox4_0(0), 17) +
resize(coefaprox4_0(1), 17);
        IF (add_temp_10(16) = '0') AND (add_temp_10(15) /= '0') THEN
          add_cast_5 := X"7FFF";
        ELSIF (add_temp_10(16) = '1') AND (add_temp_10(15) /= '1') THEN
          add_cast_5 := X"8000";
        ELSE
          add_cast_5 := add_temp_10(15 DOWNTO 0);
        END IF;
        add_cast_6 := resize(add_cast_5, 17);
        add_temp_11 := add_cast_6 + resize(media, 17);
        IF (add_temp_11(16) = '0') AND (add_temp_11(15) /= '0') THEN
          media := X"7FFF";
        ELSIF (add_temp_11(16) = '1') AND (add_temp_11(15) /= '1') THEN
          media := X"8000";
        ELSE
          media := add_temp_11(15 DOWNTO 0);
        END IF;
        --%Suma de todos los datos
        --'TWHAAR:28' C = [CoefAprox4 CoefDetail4 CoefDetail3 CoefDetail2
CoefDetail];

        FOR t_5 IN 0 TO 3 LOOP
          C(t_5 + 4) := coefdetail3_0(t_5);
        END LOOP;


        FOR t_6 IN 0 TO 7 LOOP
          C(t_6 + 8) := coefdetail2_0(t_6);
        END LOOP;


        FOR t_7 IN 0 TO 15 LOOP
          C(t_7 + 16) := coefdetail_0(t_7);
        END LOOP;

        --'TWHAAR:29' mediari=bitshift(media,-5);
        media_0 := SHIFT_RIGHT(resize(media, 32) , 5);
        mediari := media_0(15 DOWNTO 0);
        --;Calculo de la media aritmetica de los datos
        --'TWHAAR:30' C = threshold(C,mediari);
        --'threshold:2' promedio = int16(zeros(1,32));
        --'threshold:3' acumulador = int16(0);
        acumulador := to_signed(16#0000#, 16);
        --'threshold:4' for a = 1:32

        FOR a IN 0 TO 31 LOOP
          --'threshold:5' temporal = umbral-CoefDetail(a);
          sub_temp_3(a) := resize(mediari, 17) - resize(C(a), 17);
          IF (sub_temp_3(a)(16) = '0') AND (sub_temp_3(a)(15) /= '0')
THEN
```

```
            temporal := X"7FFF";
          ELSIF (sub_temp_3(a)(16) = '1') AND (sub_temp_3(a)(15) /= '1')
THEN
            temporal := X"8000";
          ELSE
            temporal := sub_temp_3(a)(15 DOWNTO 0);
          END IF;
          --'threshold:6' if (temporal)<0
          IF temporal < to_signed(16#00000000#, 16) THEN
            --'threshold:7' promedio(a) = CoefDetail(a)-int16(umbral);
            sub_temp_4(a) := resize(C(a), 17) - resize(mediari, 17);
            IF (sub_temp_4(a)(16) = '0') AND (sub_temp_4(a)(15) /= '0')
THEN
              promedio(a) := X"7FFF";
            ELSIF (sub_temp_4(a)(16) = '1') AND (sub_temp_4(a)(15) /=
'1') THEN
              promedio(a) := X"8000";
            ELSE
              promedio(a) := sub_temp_4(a)(15 DOWNTO 0);
            END IF;
          ELSE
            --'threshold:8' else
            --'threshold:9' promedio(a) = int16(umbral)-CoefDetail(a);
            sub_temp_5(a) := resize(mediari, 17) - resize(C(a), 17);
            IF (sub_temp_5(a)(16) = '0') AND (sub_temp_5(a)(15) /= '0')
THEN
              promedio(a) := X"7FFF";
            ELSIF (sub_temp_5(a)(16) = '1') AND (sub_temp_5(a)(15) /=
'1') THEN
              promedio(a) := X"8000";
            ELSE
              promedio(a) := sub_temp_5(a)(15 DOWNTO 0);
            END IF;
          END IF;
          --'threshold:11' acumulador = promedio(a)+acumulador;
          add_temp_13(a) := resize(promedio(a), 17) + resize(acumulador,
17);
          IF (add_temp_13(a)(16) = '0') AND (add_temp_13(a)(15) /= '0')
THEN
            acumulador := X"7FFF";
          ELSIF (add_temp_13(a)(16) = '1') AND (add_temp_13(a)(15) /=
'1') THEN
            acumulador := X"8000";
          ELSE
            acumulador := add_temp_13(a)(15 DOWNTO 0);
          END IF;
          --%Desviación estandar absoluta
        END LOOP;

        --'threshold:13' temporal2 =bitshift(acumulador,-5);
        acumulador_0 := SHIFT_RIGHT(resize(acumulador, 32) , 5);
        temporal2 := acumulador_0(15 DOWNTO 0);
        -- Calculo de la MAD
        --'threshold:14' THR1 = bitshift(int16(temporal2),1);
```

```
        temporal2_0 := resize(temporal2, 32) sll 1;
        THR1 := temporal2_0(15 DOWNTO 0);
        --Umbral Sqtwolog
        --
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
        --'threshold:16' for i = 1:32

        FOR i IN 0 TO 31 LOOP
          --'threshold:17' if ((CoefDetail(i))>=THR1 ||(-
CoefDetail(i))>=THR1 )
          c_8(i) :=  - (resize(C(i), 17));
          IF (c_8(i)(16) = '0') AND (c_8(i)(15) /= '0') THEN
            cast_3(i) := X"7FFF";
          ELSIF (c_8(i)(16) = '1') AND (c_8(i)(15) /= '1') THEN
            cast_3(i) := X"8000";
          ELSE
            cast_3(i) := c_8(i)(15 DOWNTO 0);
          END IF;
          IF (C(i) >= THR1) OR (cast_3(i) >= THR1) THEN
            --%Umbralización dura
          ELSE
            --'threshold:18' else
            --'threshold:19' CoefDetail(i) = int16(0);
            C(i) := to_signed(16#0000#, 16);
          END IF;
        END LOOP;

        --%Modificación de los coeficientes con Sqtwolog, y Hard
thresholding
        --'TWHAAR:31' CoefAprox4 = C(uint8(1):uint8(2));
        --'TWHAAR:32' CoefDetail4 = C(uint8(3):uint8(4));

        FOR t_8 IN 0 TO 1 LOOP
          CoefAprox4(t_8) := C(t_8);
          CoefDetail4(t_8) := C(2 + t_8);
        END LOOP;

        --'TWHAAR:33' CoefDetail3 = C(uint8(5):uint8(8));

        FOR t_9 IN 0 TO 3 LOOP
          CoefDetail3(t_9) := C(4 + t_9);
        END LOOP;

        --'TWHAAR:34' CoefDetail2 = C(uint8(9):uint8(16));

        FOR t_10 IN 0 TO 7 LOOP
          CoefDetail2(t_10) := C(8 + t_10);
        END LOOP;

        --'TWHAAR:35' CoefDetail = C(uint8(17):uint8(32));

        FOR t_11 IN 0 TO 15 LOOP
```

```
          CoefDetail(t_11) := C(16 + t_11);
        END LOOP;

        --'TWHAAR:36' [recuperada4] = ilwthaar(CoefAprox4,CoefDetail4);
        --'ilwthaar:2' Par = CoefAprox-(bitshift(int16(CoefDetail),-1));

        FOR m IN 0 TO 1 LOOP
          a1_3 := CoefDetail4(m);
          a1_11(m) := SHIFT_RIGHT(resize(a1_3, 32) , 1);
          c_4(m) := a1_11(m)(15 DOWNTO 0);
        END LOOP;

        --Creación de los componentes pares (UNDO UPDATE)
        --'ilwthaar:3' Impar = (Par)+(CoefDetail);
        --Creación de los componentes impares (UNDO PREDICT)
        --'ilwthaar:4' recuperada = [(Impar);(Par)];

        FOR t_12 IN 0 TO 1 LOOP
          sub_temp_6(t_12) := resize(CoefAprox4(t_12), 17) -
resize(c_4(t_12), 17);
          IF (sub_temp_6(t_12)(16) = '0') AND (sub_temp_6(t_12)(15) /=
'0') THEN
            par_3(t_12) := X"7FFF";
          ELSIF (sub_temp_6(t_12)(16) = '1') AND (sub_temp_6(t_12)(15) /=
'1') THEN
            par_3(t_12) := X"8000";
          ELSE
            par_3(t_12) := sub_temp_6(t_12)(15 DOWNTO 0);
          END IF;
          add_temp_14(t_12) := resize(par_3(t_12), 17) +
resize(CoefDetail4(t_12), 17);
          IF (add_temp_14(t_12)(16) = '0') AND (add_temp_14(t_12)(15) /=
'0') THEN
            impar_3(t_12) := X"7FFF";
          ELSIF (add_temp_14(t_12)(16) = '1') AND (add_temp_14(t_12)(15)
/= '1') THEN
            impar_3(t_12) := X"8000";
          ELSE
            impar_3(t_12) := add_temp_14(t_12)(15 DOWNTO 0);
          END IF;
          recuperada(0, t_12) := impar_3(t_12);
          recuperada(1, t_12) := par_3(t_12);
        END LOOP;

        --(MERGE)
        --'ilwthaar:5' recuperada = (recuperada(:).');
        --%(MERGE)
        --% Cuarto nivel recons
        --'TWHAAR:37' [recuperada3] = ilwthaar(recuperada4,CoefDetail3) ;
        --'ilwthaar:2' Par = CoefAprox-(bitshift(int16(CoefDetail),-1));

        FOR i_0 IN 0 TO 1 LOOP
          FOR i2 IN 0 TO 1 LOOP
```

```
            cast_temp := unsigned(resize(to_signed(i_0, 32), 8));
            cast_temp_0 := unsigned(resize(to_signed(i2, 32), 16));
            mul_temp := cast_temp * to_unsigned(16#02#, 8);
            oldIdx := to_integer(mul_temp + cast_temp_0);
            recuperada4(oldIdx) := recuperada(i2, i_0);
            a1_4 := CoefDetail3(oldIdx);
            a1_12(oldIdx) := SHIFT_RIGHT(resize(a1_4, 32) , 1);
            c_5(oldIdx) := a1_12(oldIdx)(15 DOWNTO 0);
          END LOOP;
        END LOOP;

        --Creación de los componentes pares (UNDO UPDATE)
        --'ilwthaar:3' Impar = (Par)+(CoefDetail);
        --Creación de los componentes impares (UNDO PREDICT)
        --'ilwthaar:4' recuperada = [(Impar);(Par)];

        FOR t_13 IN 0 TO 3 LOOP
          sub_temp_7(t_13) := resize(recuperada4(t_13), 17) -
resize(c_5(t_13), 17);
          IF (sub_temp_7(t_13)(16) = '0') AND (sub_temp_7(t_13)(15) /=
'0') THEN
            par_4(t_13) := X"7FFF";
          ELSIF (sub_temp_7(t_13)(16) = '1') AND (sub_temp_7(t_13)(15) /=
'1') THEN
            par_4(t_13) := X"8000";
          ELSE
            par_4(t_13) := sub_temp_7(t_13)(15 DOWNTO 0);
          END IF;
          add_temp_15(t_13) := resize(par_4(t_13), 17) +
resize(CoefDetail3(t_13), 17);
          IF (add_temp_15(t_13)(16) = '0') AND (add_temp_15(t_13)(15) /=
'0') THEN
            impar_4(t_13) := X"7FFF";
          ELSIF (add_temp_15(t_13)(16) = '1') AND (add_temp_15(t_13)(15)
/= '1') THEN
            impar_4(t_13) := X"8000";
          ELSE
            impar_4(t_13) := add_temp_15(t_13)(15 DOWNTO 0);
          END IF;
          recuperada_0(0, t_13) := impar_4(t_13);
          recuperada_0(1, t_13) := par_4(t_13);
        END LOOP;

        --(MERGE)
        --'ilwthaar:5' recuperada = (recuperada(:).');
        --%(MERGE)
        --% Tercer nivel recons
        --'TWHAAR:38' [recuperada2] = ilwthaar(recuperada3,CoefDetail2);
        --'ilwthaar:2' Par = CoefAprox-(bitshift(int16(CoefDetail),-1));

        FOR i_1 IN 0 TO 3 LOOP
          FOR i2_0 IN 0 TO 1 LOOP
            cast_temp_1 := unsigned(resize(to_signed(i_1, 32), 8));
            cast_temp_2 := unsigned(resize(to_signed(i2_0, 32), 16));
```

```
            mul_temp_0 := cast_temp_1 * to_unsigned(16#02#, 8);
            oldidx_0 := to_integer(mul_temp_0 + cast_temp_2);
            recuperada3(oldidx_0) := recuperada_0(i2_0, i_1);
            a1_5 := CoefDetail2(oldidx_0);
            a1_13(oldidx_0) := SHIFT_RIGHT(resize(a1_5, 32) , 1);
            c_6(oldidx_0) := a1_13(oldidx_0)(15 DOWNTO 0);
          END LOOP;
        END LOOP;

        --Creación de los componentes pares (UNDO UPDATE)
        --'ilwthaar:3' Impar = (Par)+(CoefDetail);
        --Creación de los componentes impares (UNDO PREDICT)
        --'ilwthaar:4' recuperada = [(Impar);(Par)];

        FOR t_14 IN 0 TO 7 LOOP
          sub_temp_8(t_14) := resize(recuperada3(t_14), 17) -
resize(c_6(t_14), 17);
          IF (sub_temp_8(t_14)(16) = '0') AND (sub_temp_8(t_14)(15) /=
'0') THEN
            par_5(t_14) := X"7FFF";
          ELSIF (sub_temp_8(t_14)(16) = '1') AND (sub_temp_8(t_14)(15) /=
'1') THEN
            par_5(t_14) := X"8000";
          ELSE
            par_5(t_14) := sub_temp_8(t_14)(15 DOWNTO 0);
          END IF;
          add_temp_16(t_14) := resize(par_5(t_14), 17) +
resize(CoefDetail2(t_14), 17);
          IF (add_temp_16(t_14)(16) = '0') AND (add_temp_16(t_14)(15) /=
'0') THEN
            impar_5(t_14) := X"7FFF";
          ELSIF (add_temp_16(t_14)(16) = '1') AND (add_temp_16(t_14)(15)
/= '1') THEN
            impar_5(t_14) := X"8000";
          ELSE
            impar_5(t_14) := add_temp_16(t_14)(15 DOWNTO 0);
          END IF;
          recuperada_1(0, t_14) := impar_5(t_14);
          recuperada_1(1, t_14) := par_5(t_14);
        END LOOP;

        --(MERGE)
        --'ilwthaar:5' recuperada = (recuperada(:).');
        --%(MERGE)
        --% Segundo nivel recons
        --'TWHAAR:39' [signal_in_memory] =
ilwthaar(recuperada2,CoefDetail);
        --'ilwthaar:2' Par = CoefAprox-(bitshift(int16(CoefDetail),-1));

        FOR i_2 IN 0 TO 7 LOOP
          FOR i2_1 IN 0 TO 1 LOOP
            cast_temp_3 := unsigned(resize(to_signed(i_2, 32), 8));
            cast_temp_4 := unsigned(resize(to_signed(i2_1, 32), 16));
            mul_temp_1 := cast_temp_3 * to_unsigned(16#02#, 8);
```

```
            oldidx_1 := to_integer(mul_temp_1 + cast_temp_4);
            recuperada2(oldidx_1) := recuperada_1(i2_1, i_2);
            a1_6 := CoefDetail(oldidx_1);
            a1_14(oldidx_1) := SHIFT_RIGHT(resize(a1_6, 32) , 1);
            c_7(oldidx_1) := a1_14(oldidx_1)(15 DOWNTO 0);
          END LOOP;
        END LOOP;

        --Creación de los componentes pares (UNDO UPDATE)
        --'ilwthaar:3' Impar = (Par)+(CoefDetail);
        --Creación de los componentes impares (UNDO PREDICT)
        --'ilwthaar:4' recuperada = [(Impar);(Par)];

        FOR t_15 IN 0 TO 15 LOOP
          sub_temp_9(t_15) := resize(recuperada2(t_15), 17) -
resize(c_7(t_15), 17);
          IF (sub_temp_9(t_15)(16) = '0') AND (sub_temp_9(t_15)(15) /=
'0') THEN
            par_6(t_15) := X"7FFF";
          ELSIF (sub_temp_9(t_15)(16) = '1') AND (sub_temp_9(t_15)(15) /=
'1') THEN
            par_6(t_15) := X"8000";
          ELSE
            par_6(t_15) := sub_temp_9(t_15)(15 DOWNTO 0);
          END IF;
          add_temp_17(t_15) := resize(par_6(t_15), 17) +
resize(CoefDetail(t_15), 17);
          IF (add_temp_17(t_15)(16) = '0') AND (add_temp_17(t_15)(15) /=
'0') THEN
            impar_6(t_15) := X"7FFF";
          ELSIF (add_temp_17(t_15)(16) = '1') AND (add_temp_17(t_15)(15)
/= '1') THEN
            impar_6(t_15) := X"8000";
          ELSE
            impar_6(t_15) := add_temp_17(t_15)(15 DOWNTO 0);
          END IF;
          recuperada_2(0, t_15) := impar_6(t_15);
          recuperada_2(1, t_15) := par_6(t_15);
        END LOOP;

        --(MERGE)
        --'ilwthaar:5' recuperada = (recuperada(:).');

        FOR i_3 IN 0 TO 15 LOOP
          FOR i2_2 IN 0 TO 1 LOOP
            cast_temp_5 := unsigned(resize(to_signed(i_3, 32), 8));
            cast_temp_6 := unsigned(resize(to_signed(i2_2, 32), 16));
            mul_temp_2 := cast_temp_5 * to_unsigned(16#02#, 8);
            oldidx_2 := to_integer(mul_temp_2 + cast_temp_6);
            TWHAAR_signal_in_memory_next(oldidx_2) <= recuperada_2(i2_2,
i_3);
          END LOOP;
        END LOOP;
```

```
        --%(MERGE)
        --% Primer nivel recons
        --'TWHAAR:40' contador = uint8(1);
        contador_temp := to_unsigned(16#01#, 8);
      END IF;
    END IF;

    FOR t_16 IN 0 TO 31 LOOP
      TWHAAR_signal_out_memory_next(t_16) <=
signal_out_memory_temp(t_16);
    END LOOP;

    TWHAAR_contador_next <= contador_temp;
  END PROCESS TWHAAR_1_output;


  ce_out <= clk_enable;

END rtl;
```