Universidad Rey Juan Carlos

Escuela Superior de Ciencias Experimentales y Tecnología

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Trabajo Fin de Grado

DISEÑO E IMPLEMENTACIÓN DEL FILTRADO DE SEÑALES EMG MEDIANTE HERRAMIENTAS DE SÍNTESIS DE ALTO NIVEL BASADAS EN FPGA

ALEJANDRO LÓPEZ DEL PESO

Tutor: Rubén Nieto Capuchino

Curso Académico 2023/24

Grado en Ingeniería Electrónica Industrial y Automática

Trabajo de Fin de Grado

El presente trabajo, titulado *DISEÑO E IMPLEMENTACIÓN DEL FILTRADO DE SEÑALES EMG MEDIANTE HERRAMIENTAS DE SÍNTESIS DE ALTO NIVEL BASADAS EN FPGA*, constituye la memoria correspondiente a la asignatura Trabajo de Fin de Grado, que presenta D./Dª. *Alejandro López del Peso,* como parte de su formación para aspirar al Título de Graduado/a en Ingeniería Electrónica Industrial y Automática. Este trabajo ha sido realizado en el *Departamento de Matemática Aplicada, Ciencia e Ingeniería de los Materiales y Tecnología Electrónica* bajo la dirección de Rubén Nieto Capuchino.

Móstoles, 6 de marzo de 2024

LISTADO DE ACRÓNIMOS

EMG	Electromiografía, Electromiográfica.
iEMG	EMG Intramuscular.
sEMG	EMG superficial.
PL	Low Pass (filter).
HP	High Pass (filter).
F_c	Frecuencia de corte.
F_s	Frecuencia de muestreo.
FIR	Finite Impulse Response.
IIR	Infinite Impulse Response.
FPGA	Field Programmable Gate Array.
HDL	Hardware Description Language.
VHDL	Verilog Hardware Description Language.
HLS	High Level Synthesis.
RTL	Register Transfer Level.
IP	Intellectual property.
ILA	Integrated Logic Analyzer.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. Resumen

En 2020 una encuesta del Instituto Nacional de Estadística (INE), reveló que 4,38 millones de personas en España (casi 95 de cada mil habitantes) afirmaban tener algún tipo de discapacidad (figura 1). Según la nota de prensa del INE [1], los problemas de movilidad eran los más habituales, con una tasa de 54 de cada millar de habitantes. Estudios similares de años anteriores demuestran que las cifras de personas con discapacidad han ido en ascenso desde finales del siglo pasado.

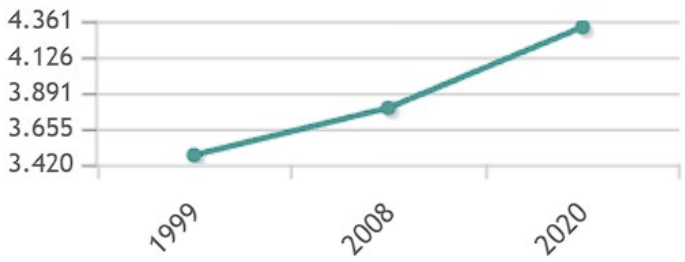

Figura 1: Valor absoluto en miles de personas de 6 o más años con discapacidad. Datos del INE [2].

Los desplazamientos en coche, los accidentes laborales y el estilo de vida junto con agentes del entorno podrían explicar el creciente número de personas con problemas de movilidad. En algunos casos los afectados consiguen recuperar parte o la totalidad de las funciones motoras que habían perdido gracias a la rehabilitación, aunque a menudo es un proceso lento que requiere gran esfuerzo físico y mental por parte del paciente.

Las técnicas de rehabilitación son numerosas, desde sencillos juegos que impliquen ciertos movimientos, hasta sistemas de cintas ergométricas con barandillas y arneses, pasando por ejercicios fisioterapéuticos. Cada uno puede ser más apropiado en unos casos según el tipo y la gravedad de la lesión y las características de los pacientes.

En particular, cuando se trata de recuperar la movilidad en las extremidades, es habitual encontrar fisioterapia pasiva en la primera etapa del tratamiento, como puede ser la electroestimulación, los masajes, movilizaciones de los miembros, o la aplicación de frío o calor, donde en todas ellas, la participación del paciente es mínima. Posteriormente se añade fisioterapia activa, requiriendo que el paciente realice cierto esfuerzo físico para realizar los ejercicios, aunque siempre es asistido por el personal o algún sistema de soporte como barandillas, arneses e incluso prótesis o exoesqueletos.

Cuando se acompaña el movimiento que realiza el paciente, es importante que la ayuda sea proporcional a la intención de movimiento de la persona. De esta forma se evita que el paciente se distraiga de la sesión de rehabilitación y, por tanto, ésta pierda eficacia. Sin embargo, ¿cómo se puede determinar el momento exacto en el que el paciente quiere mover un músculo y en qué medida?

Las señales de electromiografía (EMG) permiten medir la actividad eléctrica de los músculos y nervios, siendo ampliamente utilizadas para la detección de problemas musculares y nerviosos. Un aspecto destacado de las señales EMG es que pueden utilizarse para determinar la intencionalidad de un movimiento en pacientes con lesiones medulares parciales o que han sufrido un ictus, donde las funciones motoras pueden estar afectadas, pero las conexiones con el sistema nervioso se mantienen. Incluso si el paciente no puede llevar a cabo el movimiento o solo logra hacerlo parcialmente, se registra la actividad muscular y es posible brindarle ayuda de manera proporcional con un exoesqueleto (figura 2).

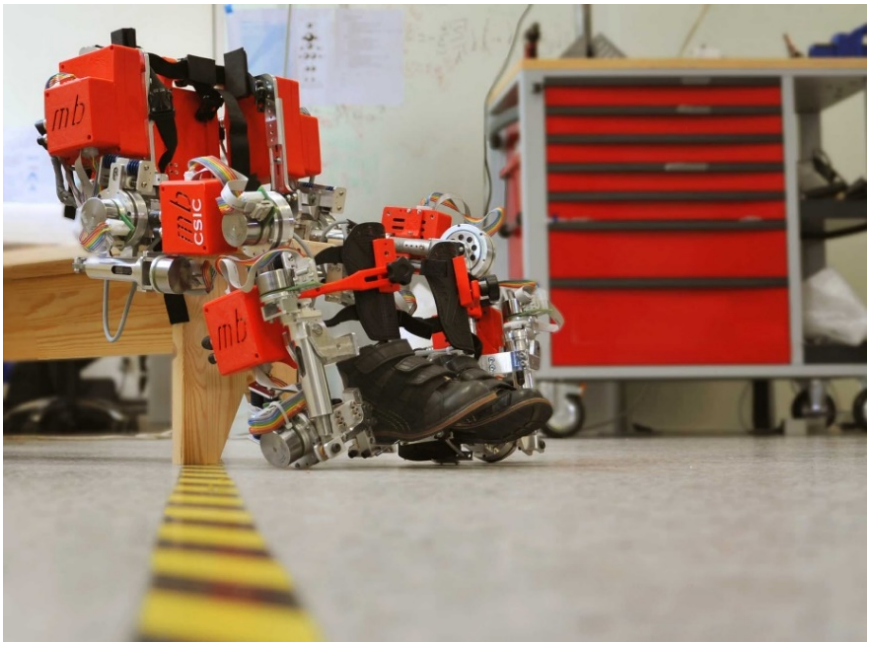

Figura 2: Exo-esqueleto Atlas desarrollado en el CSIC para niños con AME [3].

Sin embargo, Las señales EMG tienen una muy baja relación señal a ruido y su amplitud es reducida, además, son muy susceptibles al ruido electromagnético de los circuitos u otras fuentes cercanas. Por ello, para realizar una correcta interpretación de las señales EMG es necesario aplicar un procesamiento a la señal que permita determinar el instante en el que el músculo se ha activado, así como, conocer la amplitud de la activación, ya que, está fuertemente relacionada con la intensidad del movimiento.

En este Trabajo Fin de Grado se realiza el diseño e implementación de los algoritmos de detección de activación de señales EMG, basados en la rectificación y filtrado de la señal, mediante herramientas de síntesis de alto nivel (*High Level Synthesis* – HLS) para su implementación en FPGA con el objetivo de alcanzar las restricciones temporales impuestas por el procesamiento en tiempo real.

2. Introducción

La rehabilitación es una especialidad médica que involucra el diagnóstico y el tratamiento de discapacidades con el fin de mitigar sus efectos perjudiciales o devolver la capacidad funcional en la medida de lo posible. Existen numerosas técnicas con menor o mayor beneficio para el paciente, pero en general, el proceso de recuperación es lento y con pequeños logros separados en el tiempo. Esto, sumado a un monótono pero significativo esfuerzo físico y mental, puede llevar a los pecientes a un estado de desánimo y cansancio que haga disminuir aún más la efectividad de las sesiones. Frente a este problema, la rehabilitación con exoesqueletos se presenta como una potente solución para lograr el equilibrio perfecto entre el esfuerzo del paciente y la ayuda externa justa y necesaria.

Las técnicas de rehabilitación de movimiento asistido se basan en la detección de la activación muscular. La señal EMG sirve de interfaz humano – robot informando del grado de excitación de los nervios y los músculos [4], por ello, estas señales se utilizan para inferir la intención del movimiento del paciente. Al tratarse de movimientos lentos y tener un lazo de control poco restrictivo en el exoesqueleto, se tiene una ventana de tiempo suficiente para procesar las señales EMG y generar una respuesta equivalente en los motores del exoesqueleto para acompañar el movimiento de manera proporcional.

Actualmente, los métodos de detección de señales EMG que apuntan al control de un exoesqueleto en tiempo real son los llamados *Onset Detection Methods* [5], basados en la detección del inicio de la activación muscular para inferir la intención de movimiento y desencadenar una respuesta en los motores del exoesqueleto.

En cuanto a los métodos de detección *Onset* de señales EMG, se pueden distinguir tres grupos principales de métodos: los métodos visuales, los métodos basados en umbrales y los métodos estadísticos.

En cuanto a los métodos visuales, requieren de la evaluación de un profesional y pueden ser subjetivos, pero resultan útiles para validar otros métodos automáticos. Por otro lado, los métodos basados en umbrales (*Threshold methods*) establecen uno o varios umbrales para diferenciar la actividad muscular de la amplitud medida en reposo a través de un procesamiento electrónico de señal. Son los más empleados en la práctica clínica, de investigación y en la robótica, especialmente los de un solo umbral o los de umbral adaptativo. Por último, los métodos estadísticos comparan estadísticas de las señales EMG antes y después de un posible cambio.

Por lo general, en la detección *Onset* de señales EMG, no se aplican los métodos directamente a las señales EMG sin procesar, conocidas como señal en bruto o *raw*, como la que se muestra en la figura 3, donde se identifica una fase de reposo y otra de actividad muscular. En su lugar, se aplica un preprocesamiento a las señales EMG para mejorar su calidad, lo que facilita la extracción de información precisa, pero también conlleva un retardo en la obtención de resultados que debe tenerse en cuenta en un análisis de tiempo del sistema completo.

Cabe destacar que la señal EMG también se caracteriza por tener un índice de relación señal a ruido bajo, lo que dificulta la detección de activación si la señal EMG tiene una baja amplitud, confundiéndola en ocasiones por ruido.

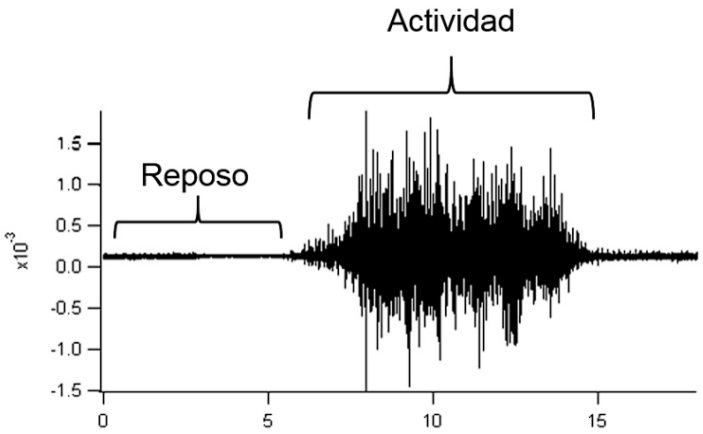

Figura 3: Señal EMG midiendo la contracción de un músculo desde el estado de reposo. El eje Y representa los cambios de amplitud de la señal en función del tiempo (eje X). referencia [4]

Al tratarse de una señal analógica, requiere una etapa de adquisición y digitalización para poder aplicar las técnicas de pre-procesamiento. A continuación, se proporciona una breve descripción de cada una de las etapas que comprende este proceso.

Los sistemas de adquisición se encargan de recopilar información proveniente de los electrodos, filtran las señales EMG de forma analógica y posteriormente se realiza la conversión digital de los datos. Esta fase inicial de filtrado tiene como principal objetivo la eliminación del ruido electromagnético originado en circuitos cercanos u otras fuentes, con el fin de prevenir su interferencia con las señales EMG, lo que podría dificultar el procesamiento y la detección precisa de la actividad muscular.

A continuación, se realiza la digitalización de la señal, que se refiere a un proceso de conversión que consiste en tomar las señales EMG, monitoreadas de manera continua en el tiempo, y transformarlas en valores discretos en términos de amplitud y tiempo. Esto da como resultado un conjunto de muestras que describe cómo se comporta la señal EMG y que pueden representarse utilizando un número finito de bits. La digitalización no solo previene la degradación por interferencias durante la transmisión hacia otros equipos del sistema, sino que es necesario para el procesamiento posterior.

En cuanto al procesado de señales EMG, una de las técnicas más utilizadas es el cálculo de la envolvente, que permite obtener una representación de los cambios lentos en la amplitud de la señal. Se entiende como lentos aquellos cambios que no son atribuibles a fluctuaciones causadas por el ruido y que no aportan información adicional. En la figura 4 se muestra un ejemplo de EMG filtrado, donde se obtiene la envolvente de la señal EMG rectificada.

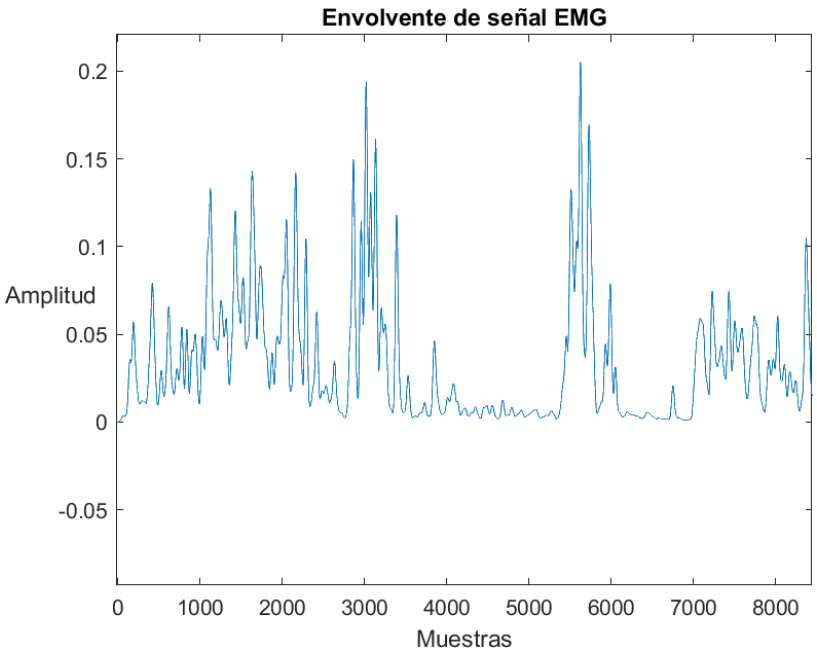

Figura 4: Envolvente calculada a partir de la rectificación y filtrado de una señal EMG. Figura de elaboración propia a partir de la base de datos [6].

Para obtener la envolvente de la señal, se utilizan diversos métodos. Uno de ellos es el método de la media móvil, el cual implica rectificar la señal, es decir, obtener el valor absoluto de la señal, y calcular el promedio de muestras de un ventana o intervalo que va desplazándose sobre la señal.

Existe también una alternativa similar, que consiste en el cálculo del valor cuadrático medio (RMS por sus siglas en inglés, *root mean square*) que se obtiene aplicando la ecuación (1) a cada ventana de muestras. Donde N es el número de muestras de la ventana, y x_i es la i-ésima muestra de la ventana. Como las muestras se elevan al cuadrado, no es necesario rectificar la señal para aplicar este método.

$$\sqrt[2]{\left(\frac{1}{N}\cdot\right)\sum_{i=1}^{N} x_i^2} \qquad (1)$$

Para obtener la envolvente, además de los métodos de promediado comentados anteriormente, se puede hacer pasar la señal EMG directamente por un filtro. Los filtros están diseñados específicamente para el procesamiento de señales que pueden contener componentes armónicas o superponerse con otras señales de naturaleza y frecuencias diversas. En términos simples, la función principal de un filtro es atenuar la intensidad de una señal a ciertas frecuencias mientras que, en otras frecuencias, la señal se amplifica o se mantiene sin cambios. Por lo tanto, en el contexto de las señales EMG, los filtros pueden ser utilizados para eliminar las rápidas oscilaciones debidas al ruido electromagnético.

Los filtros paso bajo (LP) permiten el paso de frecuencias hasta una cierta frecuencia de corte (fc) y atenúan progresivamente las frecuencias que se encuentran por encima de la fc. Por otro lado, los filtros paso alto (HP) hacen lo contrario, dejan pasar las frecuencias a partir de una fc, pero atenúan las frecuencias por debajo de esta fc.
Los filtros paso banda requieren dos frecuencias de corte para definir un intervalo de frecuencias que permiten el paso, y cualquier frecuencia fuera de ese intervalo se atenúa.
Finalmente, los filtros rechazo banda también se definen mediante dos frecuencias de corte, pero su función principal es atenuar las frecuencias que caen dentro de ese intervalo, dejando pasar las que están fuera de él.

Para visualizar la amplitud de una señal compuesta de varias frecuencias al pasar por un filtro, se utilizan unos diagramas de amplitud frente a frecuencia, denominados diagramas de Bode. En los siguientes diagramas de bode se observa el comportamiento de cada tipo de filtro mencionados anteriormente.

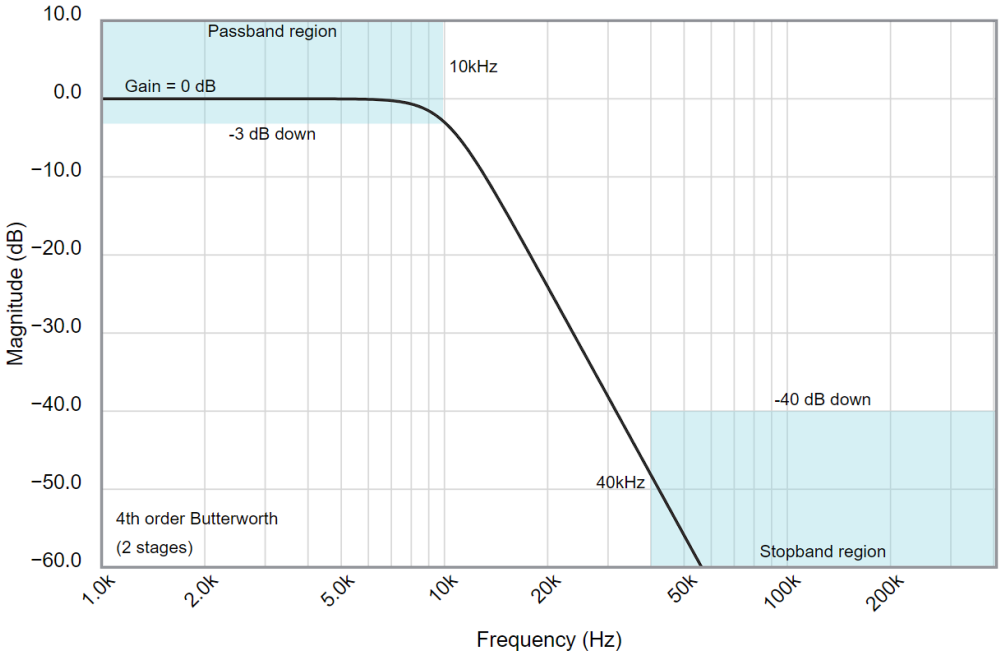

Figura 5: Diagrama de Bode de un filtro analógico paso bajo ideal. Obtenido de la herramienta Filter Wizard de Analog devices.

En el diagrama de bode (figura 5) se observa la región de la banda de paso, marcada con un recuadro azul en la esquina superior izquierda, que como se ha explicado anteriormente, es el intervalo de frecuencias que pueden pasar por el filtro con una atenuación menor de 3 dB, lo que equivale a la mitad de la potencia de la señal.

Existen dos grandes grupos de filtros, los analógicos y los digitales, que se distinguen tanto en funcionamiento como en implementación. Si se conocen las principales ventajas e inconvenientes de cada tipo, se podrá tomar una decisión acertada sobre cuál es más conveniente para la obtención de la envolvente de la señal EMG.

Los filtros analógicos se utilizan para procesar señales continuas en tiempo y amplitud. Esto significa que operan directamente con la señal original y no necesitan una digitalización, haciéndolos más precisos y fieles a la señal de entrada. Este tipo de filtros son utilizados antes de la etapa de adquisición que se comentó anteriormente. En general, tienden a ser más sencillos de implementar, aunque al depender de componentes electrónicos (resistencias, condensadores, bobinas y amplificadores operacionales), son más sensibles a las imperfecciones de los componentes y son muy sensibles al cambio de la temperatura. Además, son difíciles de modificar una vez instalados, ya que supondría cambiar componentes que ya están soldados o impresos en un circuito que incluso podría ser inaccesible una vez incorporado al producto final.

Por el contrario, los filtros digitales se implementan en software o en dispositivos lógicos programables, por lo que son fácilmente reconfigurables y sin ningún costo extra de componentes, a menos que se necesite cambiar de dispositivo por ciertas limitaciones de recursos computacionales.
Este tipo de filtros solo pueden procesar señales discretas en tiempo y amplitud, y se construyen con operaciones aritméticas. Por ello, necesitan una etapa de conversión analógico-digital cuyas características pueden limitar la precisión o velocidad del filtro.

Existen dos grandes grupos de filtros digitales: filtros FIR (*Finite Impulse Response*) cuya respuesta depende únicamente de la entrada, ya que no se realimentan con un lazo cerrado; y filtros IIR (*Infinite Impulse Response*) que sí tienen un lazo de realimentación, por lo que su salida depende tanto de la entrada como de los resultados de filtrar muestras anteriores. Estos últimos, aunque tienen mejores prestaciones, también son más complejos de implementar.

Volviendo al área concreta de aplicación de las señales EMG, es previsible que se necesiten modificar algunos parámetros del sistema, sobre todo en un marco de investigación y desarrollo. Por ejemplo, el número de canales de adquisición o la frecuencia de muestreo son algunas variables que pueden requerir modificaciones en la etapa de filtrado, por lo que un filtro analógico sería más costoso en tiempo y recursos.

Un filtro digital es mejor opción por varias razones. En primer lugar, al no depender de las imperfecciones de los componentes, ofrece una mayor repetibilidad. Por lo que de haber varios canales midiendo músculos distintos, todas las señales se filtrarían con la misma precisión. Además, al implementarse en un dispositivo reprogramable, permitiría configurar los parámetros del filtro para adaptarse a los cambios de requerimientos del sistema.

Sin embargo, ¿dónde y de qué manera se implementa un filtro digital? Hasta ahora, se ha dado a entender que los filtros analógicos consisten en circuitos impresos y componentes electrónicos soldados, mientras que los filtros digitales presentan una naturaleza más abstracta, ya que pueden implementarse tanto en software como en hardware, sin que los componentes, ni la topología del circuito, sean necesariamente visibles en este último caso.

Poniendo el foco en la implementación hardware, esta puede llevarse a cabo en varios tipos de dispositivos, siendo los más habituales los microcontroladores y las FPGAs (*Field-Programmable Gate Array*). Una de las diferencias clave entre estos dos tipos de dispositivos y la razón por la cual se opta por una FPGA en esta aplicación, es su capacidad para ejecutar procesos de manera concurrente. Aunque en general un microcontrolador puede ejecutar una secuencia de instrucciones con mayor rapidez, una FPGA permite realizar procesos en paralelo y brinda un mayor control sobre cómo se implementan. Esto es especialmente beneficioso cuando el sistema cuenta con procesos sistemáticos y que pueden llevarse a cabo en paralelo o de forma segmentada (pipeline). Además, algunas FPGAs cuentan con lógica de procesamiento (microprocesadores y memorias) que puede utilizarse para realizar ciertas tareas mediante software; a elección del diseñador. La figura 6 muestra un dispositivo basado en FPGA.

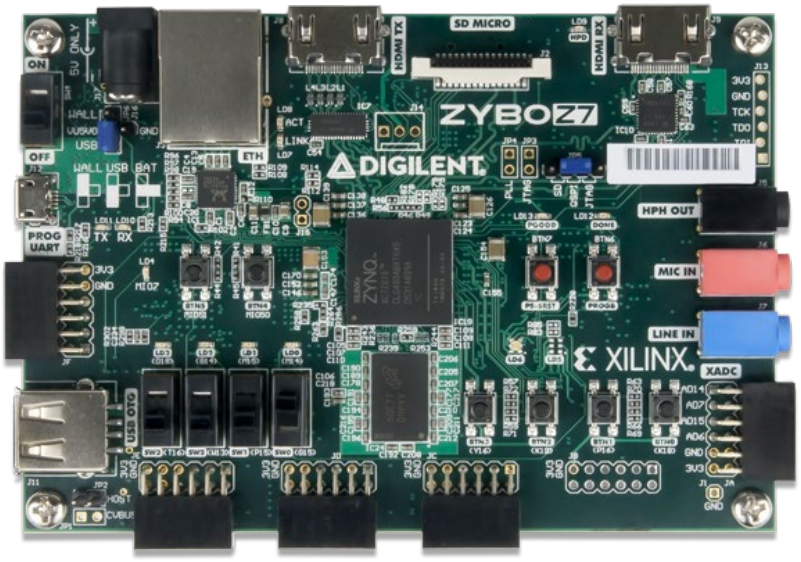

Figura 6: Placa de desarrollo con FPGA de AMD-Xilinx. Modelo Z7-10 [7].

Dado que este TFG se desarrolla sobre dispositivos basados en FPGA, a continuación, se explican brevemente los conceptos de FPGA y del diseño electrónico digital con el propósito de proporcionar los conocimientos necesarios para este trabajo.

Una FPGA es un dispositivo electrónico que contiene una matriz de bloques lógicos configurables, o CLBs (*Configurable Logic Block*). Los CLBs pueden programarse para realizar diversas operaciones lógicas y matemáticas. La programación se realiza mediante conexiones eléctricas, y la información se representa en forma binaria, utilizando voltaje alto (1) o bajo (0). Esto permite una gran flexibilidad en la implementación de funciones específicas.

Los lenguajes de descripción de Hardware (HDL) permiten a los diseñadores describir un circuito en un nivel más abstracto mediante líneas de código en RTL (*Registers Transfer Level*), es decir, con un código que describe el circuito a nivel de operaciones lógicas y de transferencia de datos. Este código es interpretado por los programas de diseño para configurar los CLBs de manera que emulen el comportamiento descrito en el código.

Una vez que el diseñador ha completado el código que describe la funcionalidad deseada, siguen varios procesos, que a veces se ejecutan simultáneamente mediante el programa de diseño.

El primero de ellos es el proceso de síntesis lógica, que consiste en que el programa de diseño interprete el código en HDL para generar una representación lógica equivalente y utilice los componentes del dispositivo, como los CLBs. Esta representación lógica se resume en una estructura llamada "*netlist*".

La *netlist* se utiliza en el proceso de implementación para determinar qué recursos de la FPGA deben utilizarse y de qué manera. La implementación está restringida por una lista de reglas o *constraints* que pueden ser físicas, las más habituales son las que designan que pines o puertos deben utilizarse y a que se conectan internamente; o temporales, que se utilizan para determinar frecuencias de reloj o tiempos de propagación de las señales entre componentes.
Simplificando, se puede interpretar que en la síntesis se lista de forma genérica los elementos necesarios para crear el circuito, mientras que en la implementación se asignan los recursos específicos disponibles en la FPGA que se va a usar.

Por último, en el proceso de "*Layout*" se trazan las conexiones entre los distintos elementos para crear un circuito que cumpla con las restricciones impuestas y con el propósito del diseño. La configuración de dichas conexiones también se conoce como enrutamiento.

El resultado del *layout* es un archivo binario de formato "*bitstream*" que se utiliza para configurar la FPGA a través de un cable y, por ejemplo, mediante el protocolo serie JTAG (JTAG, de las siglas en inglés *Joint Test Action Group*, es un protocolo estandarizado que inicialmente fue definido para testeo y verificación de circuitos integrados y circuitos impresos, aunque ha evolucionado para permitir la programación de dispositivos programables, como FPGAs a través de este protocolo serie).
Actualmente, las herramientas de software pueden realizar estos procesos automáticamente y están muy optimizadas, por lo que permiten al diseñador abstraerse de los aspectos más técnicos y complejos y centrarse solo en definir el funcionamiento.

No obstante, en ocasiones el diseñador puede intervenir y configurar algunos aspectos para que el resultado se ajuste a lo que necesita.

Al igual que en otras disciplinas, como el diseño software, en el diseño electrónico digital existen varios niveles de abstracción, es decir, varios niveles en los que el diseñador puede realizar la implementación con mayor o menor complejidad y que son proporcionales al control que tendrá sobre el resultado a nivel interno.

Al aumentar el nivel de abstracción, el diseñador puede obviar la arquitectura hardware para considerar aisladamente el funcionamiento, agilizando la descripción del último y permitiendo que la herramienta de diseño se hago cargo de la síntesis en mayor medida.

En el diseño electrónico basado en FPGAs se consideran los siguientes niveles de abstracción en orden ascendente:

1) RTL (*Register Transfer Level*): es el nivel habitual con el que trabajan los diseñadores, permite el uso de sentencias de mayor nivel que el sintetizador utiliza para inferir lógica y generar *netlist* estructurales.

2) *Behavioural* (Comportamiento): Describe el comportamiento de un módulo o una interfaz de forma general y no tanto a nivel de operaciones específicas entre registros. Suele utilizarse para crear modelos de referencia no sintetizables (no implementables en el hardware) para ayudar a verificar un modelo real.

3) Arquitectural: El nivel arquitectural se enfoca en la estructura y organización de un sistema digital en términos de sus componentes funcionales más grandes y sus interconexiones. Por ejemplo, un diseñador puede crear varios módulos con funciones únicas y después unirlos en una interfaz mayor conectando las entradas y salidas de cada bloque. Este nivel permite que el diseño de sistemas complejos pueda compartimentarse y separar funcionalidades por bloques de diseño.

4) Diseño de sistema: abarca la descripción completa del sistema y su interacción con otros componentes o subsistemas. Varias FPGAs conectadas, periféricos externos, etc.

5) *High Level Synthesis*: La síntesis de alto nivel o HLS por sus siglas en inglés tiene el objetivo de optimizar los procesos de diseño electrónico digital sin comprometer la posterior verificación. Es el más alto nivel de abstracción, está basado en lenguajes como C, C++ y requiere una síntesis de alto nivel que analice el código y lo traduzca a HDL para su posterior síntesis lógica de la que obtener una *netlist*.

   Es útil para implementar diseños menos intuitivos o para personas que no están tan familiarizadas con el diseño hardware, aunque siempre requerirá una base de conocimiento para realizar un diseño sintetizable y saber cómo optimizarlo.

Algunos programas de diseño tienen interfaces gráficas que ayudan en el diseño arquitectural y de sistema al representar cada módulo o conjunto de módulos como bloques individuales y cada interconexión entre las interfaces como cables o buses de datos.
Antes de integrar la FPGA en un sistema completo y real, es necesario asegurarse de que el diseño funciona correctamente en todos los casos de funcionamiento, tantos los esperados como aquellos que puedan darse por accidente.

Existen numerosos métodos de verificación HDL, los más comunes son aquellos en los que se crea un entorno de simulación o *testbench* también en HDL, aunque por lo general, suelen incluir elementos no sintetizables. En dicho entorno se estimula el bloque final que contiene a todo el diseño a través de las señales de entrada, tratando de cubrir cada posible escenario y verificando que internamente funciona como debe y que en las señales de salida se obtienen valores esperados.

En cuanto a la síntesis de alto nivel, esta no sustituye los conocimientos que debe poseer el diseñador, ya que, aunque puede realizar una síntesis y generar una IP en base a un código en C, un usuario de la herramienta sin conocimientos de diseño hardware no sabría interpretar los resultados de la síntesis ni integrar la IP en un sistema mayor. La utilidad de las herramientas de síntesis de alto nivel radica en la facilidad de expresar algoritmos complejos en lenguajes de alto nivel como C o C++ que serían más complejos de implementar en los niveles de abstracción habituales (más bajos) como filtros digitales o cálculos de FFT.

No obstante, no en todas las aplicaciones e industrias se busca la sencillez o rapidez por encima de otros criterios; en algunos ámbitos con reglas de diseño muy exigentes, es necesario tener control total sobre cómo se implementa cada proceso y cada sentencia de VHDL y sobre lo que está ocurriendo en cada registro en cada ciclo de reloj. Este control y visibilidad durante la etapa de diseño, previo a la verificación, es incompatible con un nivel de abstracción tan alto como el de HLS.

3. Objetivos

Antecedentes

Este TFG se enmarca en el proyecto NIMBLE (Ref. PID2021-123657OB-C32) orientado a la ingeniería biomédica aplicada, en concreto a la rehabilitación de pacientes que sufren lesiones de médula parcial y a la mejora del proceso de rehabilitación mediante exoesqueletos. Con este fin, se pretende utilizar la señal EMG para determinar la intencionalidad de movimiento de los diferentes grupos musculares e introducir la señal procesada en el lazo de control del exoesqueleto. Además, está también fuertemente ligado al proyecto ExoSen-SoC (M2998) de la URJC, que tiene como objetivo realizar arquitecturas heterogéneas basadas en System-on-Chip (SoC) para el procesado de las señales EMG.

Para la realización de este TFG se ha hecho uso de una base de datos [6] que integra las señales EMG de diferentes grupos musculares de personas caminando a velocidades conocidas.

Objetivo general

En concreto, este TFG aborda la tarea de diseñar e implementar un filtro digital FIR paso bajo para obtener la envolvente de la señal EMG. Esto es, obtener una señal que represente las variaciones en amplitud de la señal EMG discriminando aquellas oscilaciones causadas por el ruido en las altas frecuencias. El modelo debe ser sintetizable y programable en una FPGA, ya que será integrado en una arquitectura mayor de procesado y detección de inicio de la activación muscular.

Para llevar a cabo el desarrollo de este objetivo, se propone que el algoritmo se implemente en Vitis HLS en un lenguaje de alto nivel, como es C++, haciendo uso de directivas para optimizar el diseño y después realizar la síntesis de alto nivel desde el mismo software.

Objetivos Parciales

Para la planificación de las actividades, el objetivo general se descompone en los objetivos parciales listados a continuación:

1. Antes de implementar un algoritmo específico, se estudian los diferentes métodos de preprocesado explicados en la introducción: media móvil, RMS y filtrado.
2. Haciendo uso del software Matlab, se comparan los resultados de los métodos entre sí, utilizando como entrada una señal EMG de la base de datos.
3. Una vez elegido el algoritmo que se va a implementar, se escribe un código en Matlab que sirva como primera aproximación no sintetizable del código que se va a implementar para la FPGA. Haciendo referencia a los niveles de abstracción del diseño digital explicados en la introducción, el código en Matlab es comparable con el modelo *behavioural*.
4. La salida del modelo *behavioural* se compara con la salida de las funciones de Matlab. Si coinciden, se procede con la implementación en Vitis HLS del algoritmo sintetizable.
5. Antes de implementar el diseño para FPGA, se modifica el modelo *behavioural* para incluir la cuantificación de las señales y comprobar que no hay desbordamiento.
6. Se implementa el diseño en C++, después se realiza la verificación del diseño comparando su respuesta con la del modelo *behavioural* implementado en Matlab.
7. Se lleva a cabo un análisis de tiempos del sistema y se evalúa si el código puede optimizarse con directivas para reducir el consumo de recursos de la FPGA o disminuir la latencia del modelo.
8. Con el diseño ya optimizado, se comprueba que la salida sigue siendo correcta y se realiza la síntesis de alto nivel.

Tras este último punto, el diseño ya es exportable para integrarlo con el resto de los módulos de la arquitectura global.

4. Descripción de la propuesta

Hasta este punto, se han discutido las consideraciones teóricas y conceptuales relacionadas con el procesamiento de las señales EMG en el contexto de la rehabilitación con exoesqueletos. En esta sección, se abordan los aspectos técnicos de la propuesta de este TFG y se describe la implementación.

Sistema de adquisición

Antes de tomar una decisión sobre el algoritmo a implementar para el cálculo de la envolvente de la señal EMG, es importante conocer el dispositivo de adquisición de señales que se va a usar. La plataforma ADS1298R [8] pertenece a una familia de dispositivos de adquisición de datos que reúnen todas las características necesarias para aplicaciones médicas, entre las que se encuentran la electromiografía. Esta plataforma es capaz de muestrear a través de ocho canales simultáneamente, con un ADC diferencial en cada uno, y cuantizar los valores de amplitud con 24 bits de resolución y una frecuencia de muestreo configurable de hasta 32.000 muestras por segundo o sps (Samples Per Second). La figura 7 es un plano esquemático simplificado del ADS1298R:

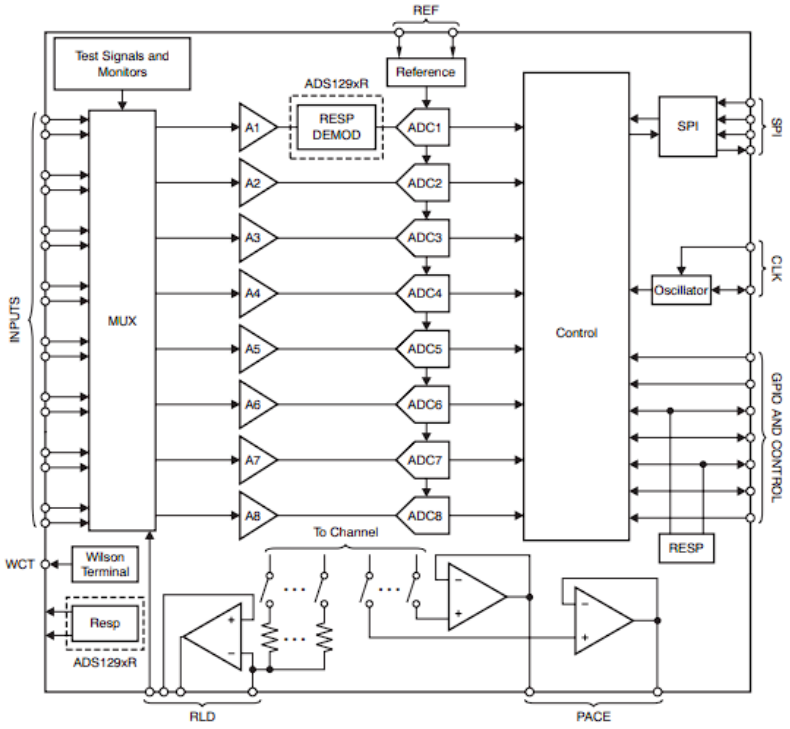

Figura 7: Plano esquemático simplificado del interior de una plataforma de adquisición ADS1298R. Captura de hoja de datos de Texas Instruments.

En el esquemático se observa ocho convertidores analógico-digital (ADC) diferenciales que corresponden a los ocho canales. No obstante, solo posee un único puerto de salida de datos SPI (Interfaz periférica serie) para transmitir la información digitalizada.

El protocolo de comunicaciones SPI envía una trama de bits para transmitir la información. En este caso, las tramas estarán compuestas de agrupaciones de datos provenientes de los ocho canales, por lo que no puede conectarse el esclavo (receptor) de la comunicación SPI directamente al módulo de preprocesamiento, de ser así, entrarían muestras provenientes de diferentes grupos musculares y la envolvente resultante sería incoherente. Para solventar esto, en el diseño arquitectural deberá colocarse un búfer[1] para acumular los datos que llegan por SPI, de esta forma el algoritmo puede procesar los canales por separado. Esto es un factor relevante que afectará a la latencia máxima del diseño, ya que debe procesar todos los canales a tiempo para no perder tramas de datos. Más adelante se profundizará en las restricciones temporales.

Elección y descripción del algoritmo a implementar

Siguiendo el flujo lógico de datos, el bloque que precede al sistema de adquisición es el del propio cálculo de la envolvente, objetivo principal de este TFG.

De los diferentes métodos que se han expuesto hasta ahora, la media móvil y el valor RMS móvil son los más sencillos de implementar, sobre todo en un lenguaje de alto nivel como C++. Estos algoritmos calculan el valor promedio (media móvil) o la media cuadrática (RMS móvil) de sucesivos segmentos de la señal digital. El resultado que persiguen es el de suavizar la señal reduciendo los efectos de las oscilaciones más rápidas y conservando los cambios lentos en la amplitud que corresponden a la información útil, tal y como funciona un filtro paso bajo.

[1] Un búfer o buffer, es un pequeño registro de desplazamiento (una pequeña memoria) dedicada a almacenar una cantidad limitada de datos temporalmente. El objetivo de estas memorias es gestionar la transmisión de datos entre módulos que funcionan a distintas velocidades o frecuencias de reloj.

En estos métodos de promediado, el tamaño de los segmentos o ventanas se refiere a la cantidad de muestras que contienen. Aumentar el tamaño de ventana significa que los promedios se calculan a partir de un mayor número de muestras, lo que mejora el suavizado de la señal, aunque puede llegar a perderse información. (Figura 8)

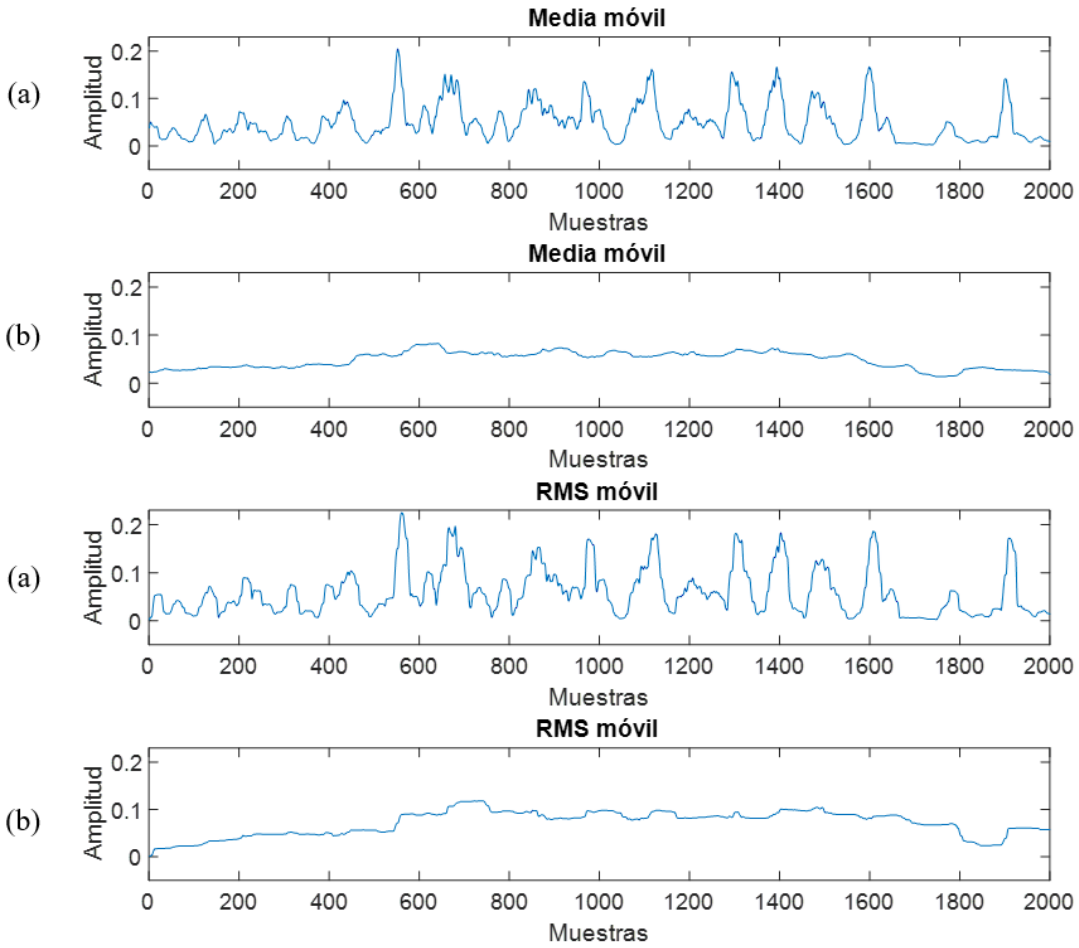

Figura 8: Envolvente de señal EMG calculada con los métodos de Media móvil y RMS móvil con un tamaño de ventana de (a) 200 muestras y (b) 20 muestras en Matlab.

La relación entre el tamaño de la ventana y las frecuencias filtradas es empírica, es decir, no existe una relación cuantitativa directa, como una ecuación, para calcular el tamaño de ventana según los parámetros frecuenciales. Esta característica hace que estos métodos sean poco atractivos si lo que se busca es una solución precisa, versátil y fácilmente adaptable a los cambios.

En contraste con lo anterior, los coeficientes de los filtros digitales se calculan en base a una frecuencia de corte específica, y si se requiere ajustar la frecuencia de corte o la frecuencia de adquisición, los coeficientes pueden ser recalculados utilizando los nuevos parámetros frecuenciales. Además, modificar el orden del filtro ya sea para aumentarlo o disminuirlo, se reduce a variar el número de coeficientes antes de recalcularlos. En cualquier combinación de estos casos, no sería necesario modificar la estructura del filtro.

Para un algoritmo de media móvil o de valor RMS móvil, también sería posible modificar el tamaño de la ventana de cálculo sin necesidad de reestructurar el código, pero la respuesta en frecuencia sigue siendo menos controlable.
A partir de la información anterior, se ha elegido implementar un filtro digital para el cálculo de la envolvente por sus mejores prestaciones y mayor versatilidad. Pero antes de pasar al diseño, deben identificarse los tipos de filtros digitales, así como sus las ventajas e inconvenientes.

Las dos grandes categorías de filtros digitales se diferencian principalmente por su respuesta al impulso: Los filtros FIR (*Finite Impulse Response*) tienen una respuesta finita, es decir, si la entrada se hace cero, al poco tiempo la salida también valdrá cero. Esto se debe a que los filtros FIR son inherentemente causales y su salida solo depende de la entrada. Los filtros IIR (Infinite Impulse Response), al contario que los filtros FIR, exhiben una respuesta infinita una vez que son estimulados. Esto se debe a que cada muestra de salida se calcula no solo en función de los datos de entrada actuales, sino también de las muestras de salida previas. Como resultado, una vez que se estimulan, estos filtros siempre generarán una salida, por pequeña que esta sea.

La realimentación que caracteriza a los filtros IIR implica un proceso de diseño mucho más preciso y cuidadoso. Esto se debe a que estos filtros tienen el potencial de volverse inestables y generar una salida que oscile infinitamente sin converger al valor deseado. Sin embargo, la ventaja es que, en comparación con un filtro FIR de prestaciones similares, los filtros IIR requieren un orden mucho menor. Esto tiene un efecto proporcional en la reducción del número de coeficientes y operaciones necesarias, lo que a su vez disminuye el consumo de recursos computacionales.

Por otro lado, el filtro FIR se presenta como una opción segura en términos de estabilidad. La ausencia de retroalimentación hace que los filtros FIR tengan una respuesta acotada y no hay posibilidad de que oscile infinitamente.

Otra ventaja de los filtros FIR es que, al realizar una suma finita de productos, es más difícil que ocurran desbordamientos por falta de bits, siempre que se haya hecho un dimensionamiento correcto. Los desbordamientos ocurren cuando los datos toman valores imposibles de representar con el número de bits que se han asignado en la cuantificación, es decir, cuando los valores se salen de escala, por lo que es un fenómeno que debe evitarse para conservar la precisión y coherencia en los resultados. La cuantificación también determina la precisión en cifras decimales, por lo que, a la hora de diseñar el filtro, debe tenerse en cuenta el rango de valores de los datos y el de los resultados de las operaciones para evitar la saturación (recortes en los valores al salirse de la escala representable).

Los filtros IIR y FIR también se diferencian en el tratamiento de las frecuencias. En los filtros FIR el retraso temporal es el mismo a cualquier frecuencia (esto es independiente de cuales se filtren y cuales no), mientras que en los filtros IIR pueden aparecer distorsiones en la señal de salida. Aunque en algunas aplicaciones no es relevante, en sistemas de tiempo real como este es importante la alineación temporal y la conservación de la forma de onda a la salida del filtro.

Tabla 1: Resumen de las principales diferencias entre los filtros digitales FIR y los IIR.

Filtro FIR	Filtro IIR
Respuesta al impulso finita	Respuesta al impulso infinita
Siempre estables	Estabilidad no asegurada
Mayor consumo de recursos	Menor consumo de recursos
Rango de valores acotados	Posibilidad de desbordamiento en las operaciones
Fase lineal	Fase no lineal

Con este análisis y las simulaciones en Matlab, se ha determinado que el filtro FIR es más adecuado para esta aplicación. Antes de pasar a su implementación se va a describir su estructura y funcionamiento:

Los filtros FIR, en esencia, son comparables a un registro de desplazamiento. Estos filtros constan de una memoria de tamaño limitado que desplaza de manera secuencial los valores almacenados de una posición a otra, desde la entrada hasta la salida. Cada posición en la memoria se asocia con un coeficiente fijo que se utiliza para multiplicar el valor correspondiente en esa posición. Luego, todos estos productos se suman para generar una única salida. Posteriormente, el registro se desplaza al introducir un nuevo dato en el extremo inicial y eliminar el dato más antiguo en el extremo final. Este proceso se repite de manera continua para obtener datos de salida sucesivos. Es importante destacar que el registro de desplazamiento tiene un tamaño finito, el número de celdas de memoria, y, por tanto, el número de coeficientes siempre es igual al orden del filtro más uno. Esto se debe a que los coeficientes se numeran desde el coeficiente 0 hasta el coeficiente número (L-1), donde L es el orden del filtro.

Un filtro FIR de orden L se describe mediante la ecuación en diferencias:

$$\boldsymbol{y(n) = a_0 \cdot x(n) + a_1 \cdot x(n-1) + a_2 \cdot x(n-2) + \cdots + a_L \cdot x(n-L)} \quad (2)$$

Donde:

- L : Es el orden del filtro FIR y coincide con el número de coeficientees menos uno.
- $\mathrm{x(n)}$: Es la enesima muestra de entrada que ocupa la primera posición del filtro.
- $\boldsymbol{a_l}$: Es el coeficiente fijo que ocupa la posición l dentro del filtro (desde la 0 a la L).
- $\mathrm{y(n)}$: Es la muestra de salida correspondiente a una muestra de entrada x(n).

Que expresado como sumatorio queda:

$$\boldsymbol{y[n] = \sum_{l=0}^{L} x[n-l] \cdot a[l]} \quad (3)$$

Se observa que la salida y[n] es un único dato que se calcula a partir de la suma ponderada del dato de entrada x[n] y de los [n – L] datos previos. Esto significa que el filtro no empieza a funcionar eficazmente hasta que todas sus posiciones se han llenado. Existe por tanto un retraso en la salida respecto de la entrada (Figura 9).

Como los filtros FIR son de fase lineal, el retraso temporal es igual a cualquier frecuencia, y por tanto se habla de retraso de grupo. En un filtro de orden L impar, con coeficientes cuyos valores son simétricos respecto al coeficiente central, se cumple que el retraso de la salida es de $\frac{L-1}{2}$ muestras, que si sé multiplica por el periodo de muestreo (se divide por la frecuencia de muestreo) resulta en el retardo de grupo de la salida del filtro en segundos.

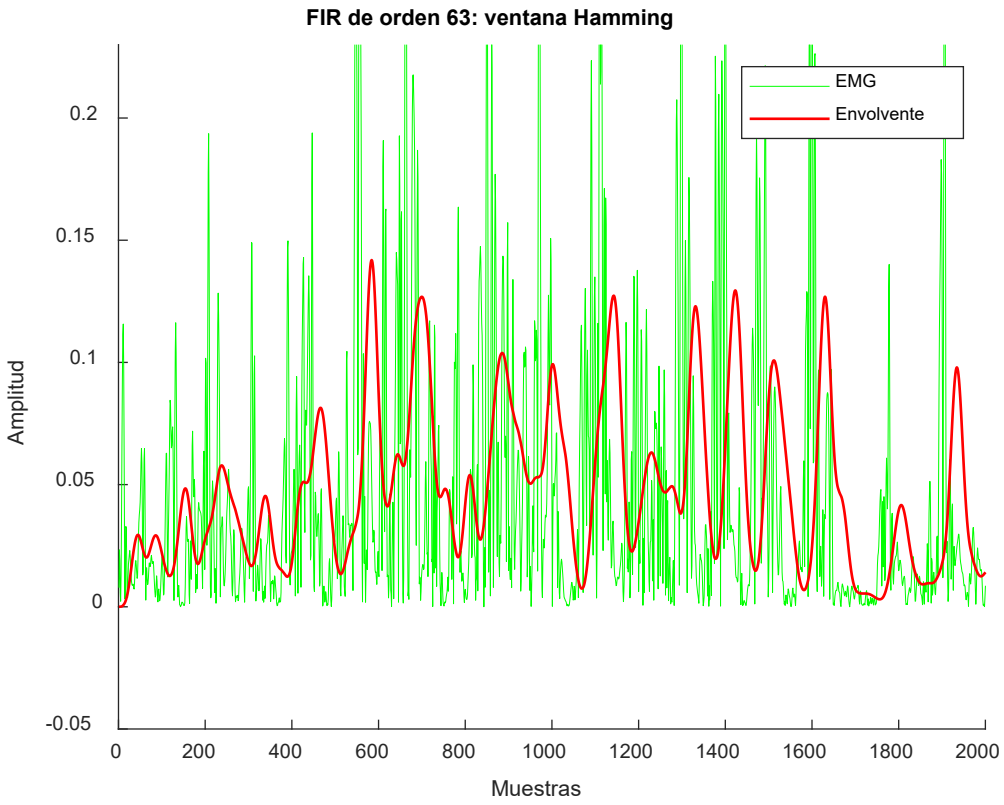

Figura 9: Gráfica de señal EMG digitalizada en crudo (naranja) junto a su envolvente (en azul). La envolvente se ha calculado con un filtro FIR de orden 63 y (63 + 1 coeficientes) en Matlab.

En la figura 10 se muestra un ejemplo visual de un filtro de orden 3 (4 coeficientes), donde Z^{-1} hace referencia a un retardo temporal de una unidad de las muestras. Así, cada muestra nueva x[n] que entre al filtro, se ubicará en la posición 3 para multiplicarse por el coeficiente b_3, mientras que la muestra del instante anterior, $x[n-1] = x[n]\cdot z^{-1}$, se multiplica por el coeficiente b_2; la muestra de dos instantantes anteriores, $x[n-2] = x[n]\cdot z^{-2}$, se multiplica por el coeficiente b_1; la muestra de tres instantantes anteriores, $x[n-3] = x[n]\cdot z^{-3}$, se multiplica por el coeficiente b_0.

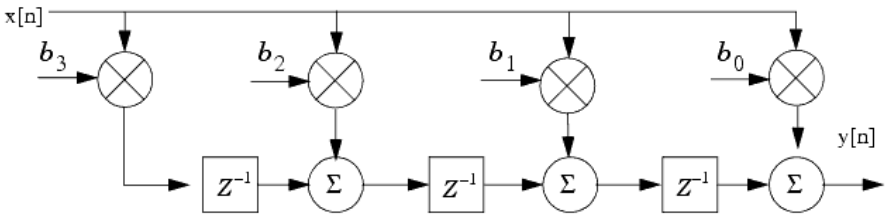

Figura 10: Esquema de filtro FIR. La muestra de salida y[n] es la suma de los productos de las muestras de entrada x[n] por los coeficientes bi. Imagen obtenida de internet

La ecuación que describe el filtro de la figura 10 es la siguiente:

$$y[n] = \mathrm{x[n]} \cdot b_3 + \mathrm{x[n]} \cdot z^{-1} \cdot b_2 + \mathrm{x[n]} \cdot z^{-2} \cdot b_1 + \mathrm{x[n]} \cdot z^{-3} \cdot b_0 \qquad (4)$$

Que es equivalente a la expresión (5)

$$y[n] = \mathrm{x[n]} \cdot b_3 + \mathrm{x[n-1]} \cdot b_2 + \mathrm{x[n-2]} \cdot b_1 + \mathrm{x[n-3]} \cdot b_0 \qquad (5)$$

Restricciones temporales del filtro

Para implementar el filtro, primero hay que definir los parámetros de diseño en base a los requerimientos del sistema y la naturaleza de las señales EMG.

Lo primero es establecer las frecuencias de corte. El espectro de frecuencias de las señales EMG depende de factores individuales como la edad o la salud, [6] recoge que la información útil de la señal EMG se sitúa entre los 10 Hz y los 500 Hz, por lo que el filtro debería dejar pasar al menos ese rango. Es importante conocer la frecuencia máxima de la señal para realizar un muestreo correcto.

El teorema de Nyquist dicta que una señal debe muestrearse por lo menos al doble de su frecuencia máxima, esto se hace para evitar la pérdida de información por el fenómeno de solapamiento o *aliasing.* Este fenómeno puede estudiar tanto en el dominio temporal de la señal como en el frecuencial. En el dominio temporal se observa que la señal muestreada no es fiel a la señal original debido a la falta de información y no es posible reconstruir la señal original a partir de la muestreada (figura 11).

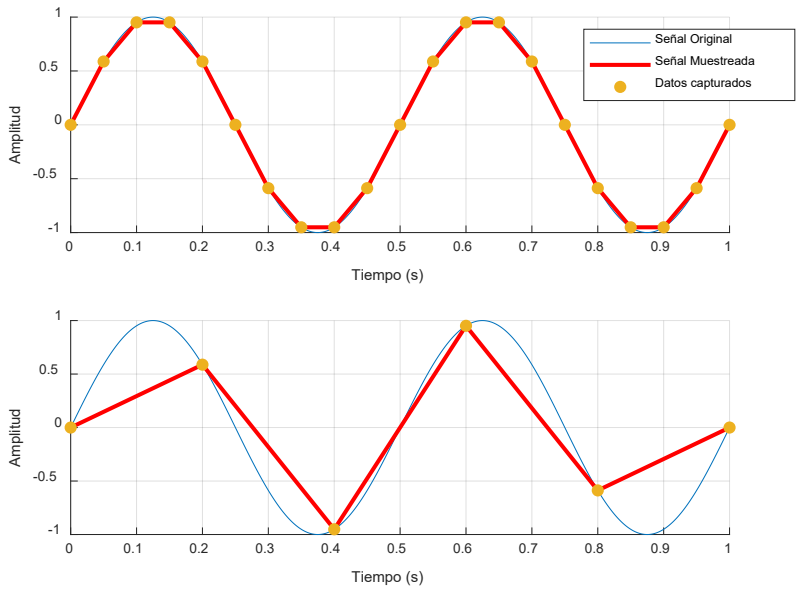

Figura 11: Señal senoidal de 2 Hz muestreada a 20 Hz (superior) y a 5 Hz (inferior). Se observa que conforme aumenta la frecuencia de muestreo, la señal muestreada es más parecida a la original. Elaboración propia en Matlab.

La explicación en el dominio frecuencial es más abstracta, cuando se hace la convolución de la señal original con un tren de deltas para muestrear la señal, en el dominio de la frecuencia se observan réplicas del espectro trasladadas a la frecuencia de muestreo, si el tren de deltas tiene una frecuencia inferior a la frecuencia máxima de la señal original, esas réplicas se solaparán con el espectro de la frecuencia fundamental, provocando *aliasing* (figura 12).

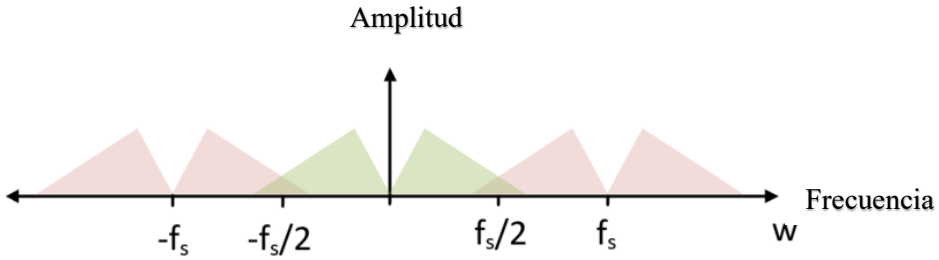

Figura 12: Ejemplo de solapamiento del espectro por submuestreo. fs es la frecuencia de muestreo. Elaboración propia

Recordando que la frecuencia máxima de interés de las señales EMG está en los 500 Hz, el sistema de adquisición debería tener una frecuencia de muestreo mínima de 1 kHz para cumplir el teorema de Nyquist, aunque en el marco del proyecto ExoSen-Soc, las señales se van a muestrear a 8 ksps (8.000 muestras por segundo). Esta frecuencia es mucho mayor que la frecuencia máxima de las señales EMG, pero forma parte de los objetivos del proyecto de investigación comentados anteriormente.

Para explorar la banda de frecuencias de los 4 kHz, deberían filtrarse todas las frecuencias superiores a esta para eliminar el ruido. Es recomendable situar la frecuencia de corte una década por debajo de la frecuencia a partir de la cual se quiere filtrar, ya que la atenuación no es inmediata, sino que sigue una pendiente de atenuación como la que se observa en la figura 5. Por tanto, la frecuencia de corte se debe situar en 400 kHz. Nos obstante, las simulaciones y la implementación se han realizado tomando la base de datos [6] como referencia, y las señales EMG que contiene han sido muestreadas a 2 kHz, con lo que se ha propuesto filtrar a partir de los 500 Hz y, por tanto, se ha situado la frecuencia de corte en 50 Hz.

La frecuencia de muestreo también determina la latencia máxima del filtro a implementar. La latencia de un sistema se refiere al tiempo que transcurre desde que se introduce una entrada hasta que se obtiene una respuesta (salida) correspondiente a esa entrada. Por tanto, la latencia del filtro es el tiempo que requiere para procesar las muestras, operar con ellas, generar un resultado y tenerlo listo en la salida, con todo lo que ello conlleva en cuanto a propagación de las señales, tiempos de establecimiento[2] y de *hold*[3] (retención).

Muestrear a 8 ksps significa que el sistema adquiere una muestra con un periodo de 125 µs, por tanto, la latencia máxima admisible es de 125 µs:

$$T = \frac{1}{8.000\,\frac{muestras}{s}} = 125 \cdot 10^{-6}\,\frac{s}{muestra} = 125\,\frac{\mu s}{muestra} \qquad (6)$$

Esto implica que el filtro debe procesar todos los canales que le inyecten datos, y tener listo un dato de salida en menos de 125 µs.

[2] El tiempo de establecimiento de una señal eléctrica se refiere al tiempo que tarda en estabilizarse en un valor tras sufrir un cambio, por ejemplo, una señal digital que conmuta de '0' a '1' tendrá un tiempo de establecimiento desde que deja de valer '0' hasta que se mantiene fija en '1', y, entre medias, su valor es desconocido y se considera no estable.

[3] El tiempo de hold de una señal eléctrica hace referencia al tiempo que una señal debe mantenerse estable para poder leerse correctamente, siguiendo el ejemplo de la señal digital, esta debe mantenerse con el valor de '1' durante el tiempo de hold para que pueda ser validada. En general, no respetar los tiempos de establecimiento y de hold puede provocar metaestabildiad, hacer que el sistema entre en estados desconocidos e incluso dañar los componentes internos de la FPGA.

Restricciones de área del filtro

En términos de diseño mediante dispositivos basados en FPGA, el área se refiere al porcentaje de recursos disponibles en la FPGA, como Flip-Flops (FF), bloques DSP, LUTs (*Look-Up Tables*) y memoria, que se utiliza para implementar la arquitectura diseñada.

Aunque no se puede conocer con exactitud los recursos que van a ser necesarios para implementar un diseño, aún menos cuando consta de varias partes que desarrollan varios ingenieros, es recomendable hacer una estimación y minimizar el consumo de recursos siempre que sea posible. Esto evitará que en una etapa avanzada del proyecto resulte necesario reestructurar módulos que ya estaban validados o tener que cambiar de dispositivo a uno con mayores prestaciones y, por lo tanto, de mayor coste.

Con relación al filtro FIR, este deberá procesar señales EMG provenientes de ocho canales, y esto, junto con la latencia máxima, condiciona el diseño del filtro y su optimización. Una posibilidad es procesar todos los canales en paralelo, lo cual optimizaría el tiempo de procesamiento, pero aumentaría significativamente el consumo de recursos. Otra opción es procesar los canales en serie, lo cual minimizaría el consumo de recursos, pero aumentaría el camino crítico y puede llegar a limitar la latencia máxima más que el propio periodo de adquisición. También se puede combinar procesamiento en serie y en paralelo para encontrar un equilibrio adecuado. En cualquier caso, si existe algún problema en el diseño arquitectural, se pueden aplicar directivas al código del filtro para tratar de solventarlos.

Otro condicionante es la resolución de los datos, tanto de entrada como de salida y de los procesos intermedios. Se sabe que el sistema de adquisición envía los datos digitalizados en 24 bits, esto quiere decir que los valores de amplitud de la señal EMG están cuantificados en 2^{24} niveles de amplitud. Considerando que las señales EMG varían desde 1 microvoltio hasta los 50 milivoltios [9] y que debe dedicarse un bit para indicar el signo, se puede calcular la resolución de los datos con la ecuación (7):

$$Q = \frac{\text{Fondo de escala } [\mu V]}{2^n} \tag{7}$$

Donde n es el número de bits y el fondo de escala, en este caso, serían 50.000 microvoltios. Por tanto, la resolución en voltios de los datos de entrada es de 0,00669 μV.

Para determinar la cuantificación de los datos del filtro (coeficientes, resultados de sumas y multiplicaciones y datos de salida) es necesario analizar algunas especificaciones de la FPGA que se va a utilizar.

La FPGA de AMD-Xilinx en la que se va a implementar el filtro, es la XC7Z010-1CLG400C integrada en la placa de desarrollo Zybo Z7-10 (figura 6). Entre sus especificaciones, se encuentran 35.200 Biestables, componentes encargados de almacenar la información en bits y 80 bloques DSP (*Digital Signal Processor*), necesarios para las operaciones de suma y multiplicación que debe realizar el filtro. Los bloques DSP son la base del funcionamiento del filtro y un factor limitante para este diseño debido a la reducida cantidad disponible.

Los DSPs de la FPGA en cuestión poseen varias entradas de datos de distintas resoluciones (A, B, C y D) y una única salida (P). Internamente, incluye un módulo sumador (*pre-adder*) y el multiplicador (*multiplier*) de 25x18 bits y se utilizarán para realizar la suma ponderada y obtener un resultado de un máximo de 48 bits (figura 13).

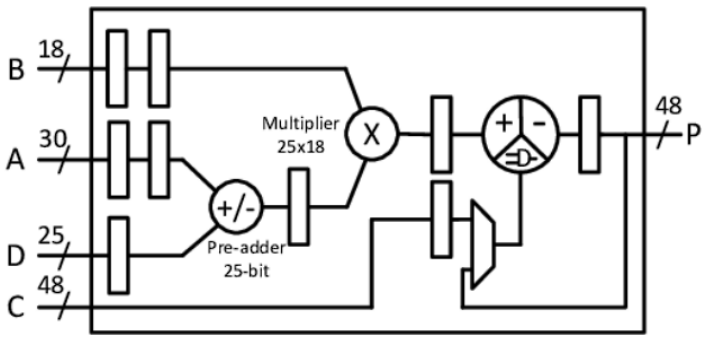

Figura 13: Bloque de procesamiento de señales digitales (DSP) de FPGA de AMD-Xilinx.

Teniendo en cuenta la estructura del DSP y los valores que pueden tomar los coeficientes del filtro, puede establecerse el número de bits con el que se representará cada dato:

Coeficientes (18 bits): Los coeficientes calculados en Matlab son todos positivos y están normalizados entre 0 y 1. Para introducirse al multiplicador debe hacerse por la entrada B, por tanto, su resolución será de 18 bits, dejando un bit para el bit de signo y 17 bits para representar la parte decimal.

Datos de entrada: La plataforma de adquisición envía los datos codificados en 24 bits, por lo que puede introducirse el dato al pre-adder por cualquiera de sus dos entradas.

Salida del multiplicador: En general, el resultado de una multiplicación necesita como máximo tantos bits como la suma de los bits de los factores. En este caso la parte entera no es necesario sumarla, ya que ambos factores son, en valor absoluto, menores o iguales a uno. Sin embargo, la parte decimal si debe tenerse en cuenta, que requiere 41 bits (17 + 23). Incluyendo el bit de signo de la parte entera, la salida del multiplicador resulta de 42 bits.

Acumulador: Independientemente de la cuantificación de entrada al sumador, dado que este realizará la suma consecutiva de varias muestras, es necesario calcular cual es el máximo de bits que puede necesitar en cualquiera de sus iteraciones. El número de bits extra que se deben añadir a la cuantificación de entrada al sumador para evitar desbordamientos es de logaritmo en base 2 del número de sumas. En este filtro de 64 coeficientes se sumarán 64 productos, por lo que son necesarios $\log_2(64) = 6$ bits extra.

En realidad, al ser un filtro de ganancia unidad, el resultado final del sumador no puede ser mayor que la entrada, pero esto no garantiza que una o varias de las sumas intermedias pueda necesitar una cuantificación mayor de la parte entera.

Datos de salida: Con el fin de asegurar la compatibilidad con la mayoría de los buses y facilitar la integración con el resto del sistema, se opta por truncar el tamaño total del dato a 32 bits. La parte entera del dato de salida constará de 7 bits (1 bit de signo + 6 bits necesarios en el acumulador), mientras que la parte decimal se reducirá a 25 bits (32 en total - 7 de la parte entera). Esto implica una pérdida de precisión, dado que, como se explicó en el contexto del multiplicador, el tamaño de la parte decimal para conservar la máxima precisión debería ser de 41 bits.

La figura 14 resume la cuantificación escogida para cada sección del *datapath* (Camino de los datos) con la notación Q(IL,FL), donde IL (*Integer Length*) es el tamaño de la parte entera y FL (*Fraction Length*) es el tamaño de la parte decimal.

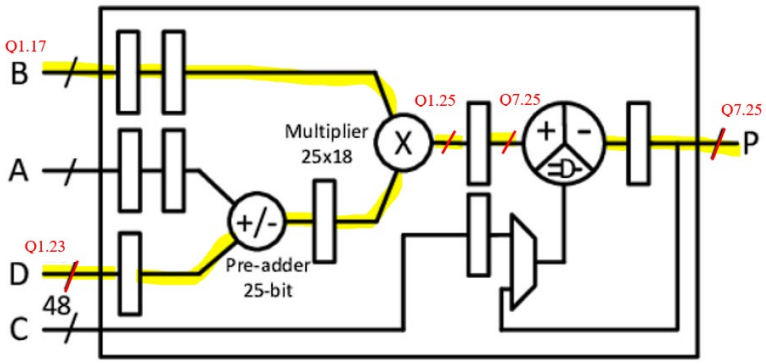

Figura 14: Cuantificación del datapath a través del DSP.

Para implementar la suma de productos existen varias alternativas, la más sencilla sería utilizar DSPs en cascada, de forma que cada bloque realice una multiplicación y el resultado sea la entrada del bloque siguiente. Esta opción podría dejar pocos o ningún DSP disponible para el resto de los módulos de procesado y podría comprometer el diseño más adelante por falta de recursos. Otra opción es implementar un sumatorio en bucle con un único DSP llevando la salida P a la entrada C del sumador, pero esto implica realizar solo una multiplicación por iteración y aumentar en gran medida la latencia del filtro. También pueden implementarse combinaciones de ambos métodos. En realidad, la implementación y el rutado depende del sintetizador, pero conocer la estructura de los bloques DSP puede ayudar al diseñador con el uso de las directivas de optimización y con la propia descripción de funcionamiento en lenguaje C.

La figura 15 muestra las posibles implementaciones de la acumulación.

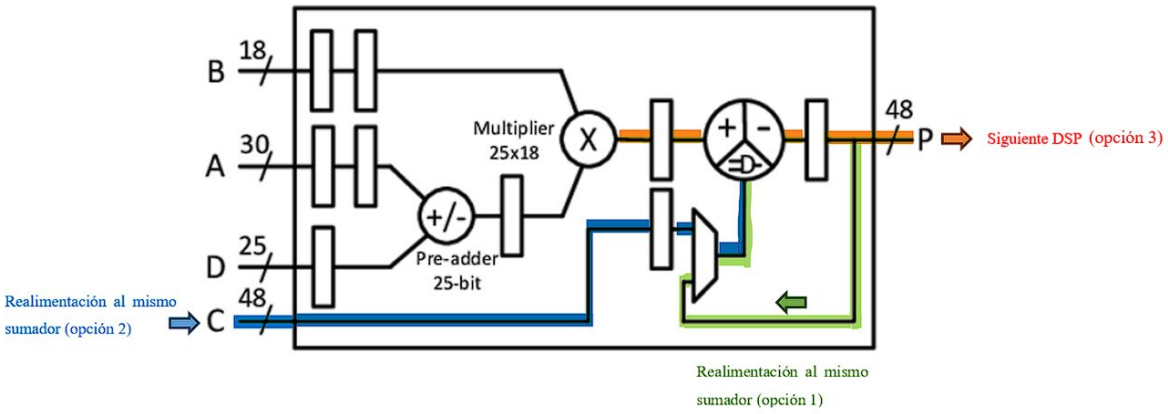

Figura 15: Alternativas de acumulación de sumas en el DSP.

Con la frecuencia de corte fijada en 50 Hz, se ha determinado a partir de simulaciones en Matlab que el número mínimo de coeficientes, potencia de 2, que garantiza una buena calidad en la envolvente es 64, por lo que el filtro debe ser de orden 63. Se ha preferido un número de coeficientes potencia de 2 porque muchas operaciones digitales son más eficientes cuando se trabaja con potencias de 2.

Con 64 coeficientes, si se implementa la suma en cascada se necesitarían 64 DSPs, uno por cada producto, dejando solo 16 DSPs para el resto del sistema. Si además se quisiese paralelizar el procesamiento de los canales no habría suficientes bloques de procesamiento de señales para el propio filtrado, por tanto, el diseño se va a optimizar en área todo lo posible mientras cumpla con la latencia máxima.

Tras la implementación del filtro, se ha realizado una síntesis de alto nivel sin aplicar optimizaciones. Durante esta etapa, se ha confirmado que la latencia del filtro al utilizar un único DSP está por debajo de 1,5 µs (tabla 4), lo cual se encuentra muy por debajo de la latencia máxima admisible. Esto indica que, en teoría, es factible emplear un único DSP incluso si se multiplica por los 8 canales. A pesar de esta posibilidad, se ha realizado una segunda versión con optimizaciones para analizar cómo se ve afectada la latencia. Esto se ha hecho principalmente con el propósito de adquirir experiencia en el uso de las directivas de Vitis HLS.

Las optimizaciones consisten en dividir el registro de coeficientes y el registro de desplazamiento del filtro en partes iguales, de esta forma se pueden generar al mismo tiempo varios accesos de lectura y escritura con datos independientes. Además, se ha aplicado un *pipeline* al bucle de operaciones, lo que permite la paralelización de operaciones de suma y multiplicación siempre y cuando no haya dependencias de datos (cuando dos operaciones necesitan trabajar con el mismo dato simultáneamente), como si se tratase de 4 filtros más pequeños cuyas salidas se unifican en un solo vector de muestras. En la tabla 2 se recogen las características del filtro implementado:

Tabla 2: Resumen de las características del filtro FIR implementado.

Implementación	HLS con lenguaje C++	
Frecuencia de corte	50 Hz (paso bajo)	
Orden	63	
Número de coeficientes	64	Valores normalizados entre 0 y 1
Ventana	Hamming	

5. Resultados

Para estudiar los diferentes métodos, se han utilizado distintas señales de la base de datos, a las que se han aplicado funciones de media móvil (MA), RMS móvil y filtrado, propias de Matlab, variando los parámetros para visualizar el comportamiento y comparar los resultados de cada algoritmo.

Para simular cada uno de los métodos, se han utilizado las funciones recogidas en la tabla 3:

Tabla 3: Funciones de Matlab utilizadas para simular la envolvente con cada método.

Filtro FIR (diseño)	`fir1()`
Filtro FIR (simulación)	`filter()`
Media móvil	`movmean()`
RMS móvil	`dsp.MovingRMS()`

Las señales de la base de datos se han adquirido de sujetos sanos caminando 10 metros a distintas velocidades, los resultados mostrados pertenecen al músculo tibial anterior izquierdo de una persona caminando a 1 km/h. La señal EMG digital se compone de 26070 muestras, pero se han acotado las gráficas a las 2500 primeras muestras para facilitar el análisis (figura 16). Las envolventes se han calculado para la señal EMG rectificada (en valor absoluto).

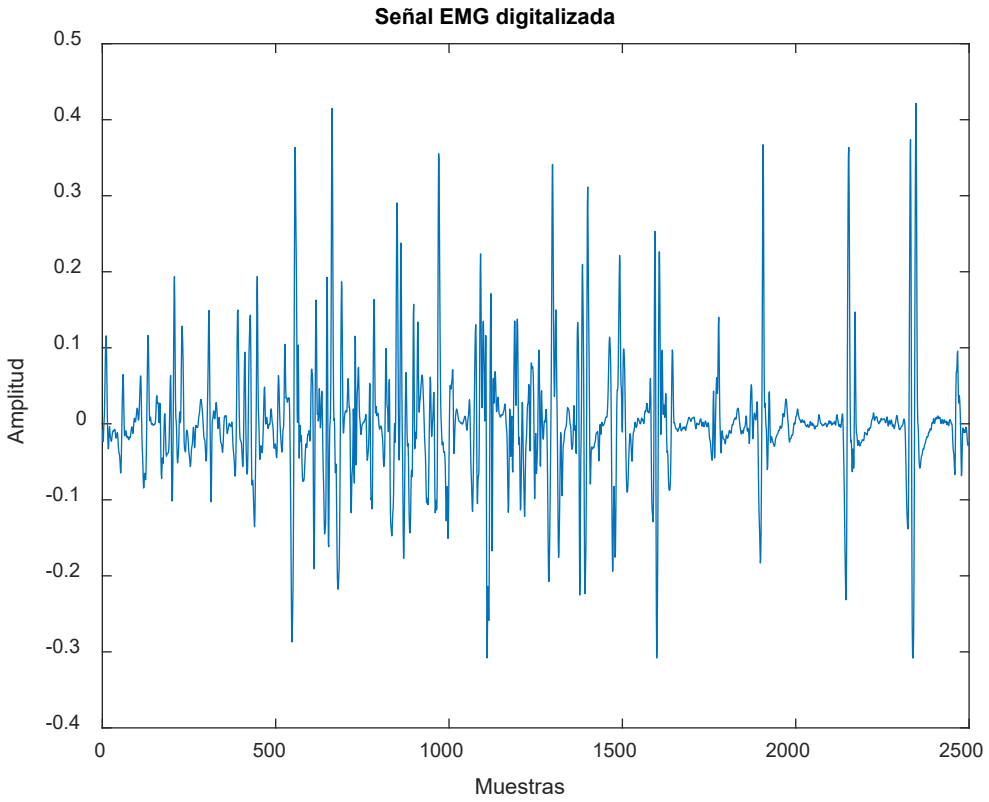

Figura 16: Señal EMG perteneciente al musculo tibial anterior de una persona caminando una distancia de 10 metros a 1 km/h.

Orden 15, Frecuencia de corte 500 Hz.

La primera prueba se ha hecho con un filtro de orden 15 (16 coeficientes), puesto que se busca minimizar la longitud del filtro, y una frecuencia de corte de 500 Hz. El resultado se ha comparado con los de una MA y RMS móvil de un tamaño de ventana similar al del filtro, es decir, de 16 muestras. Los resultados se muestran en la figura 17.

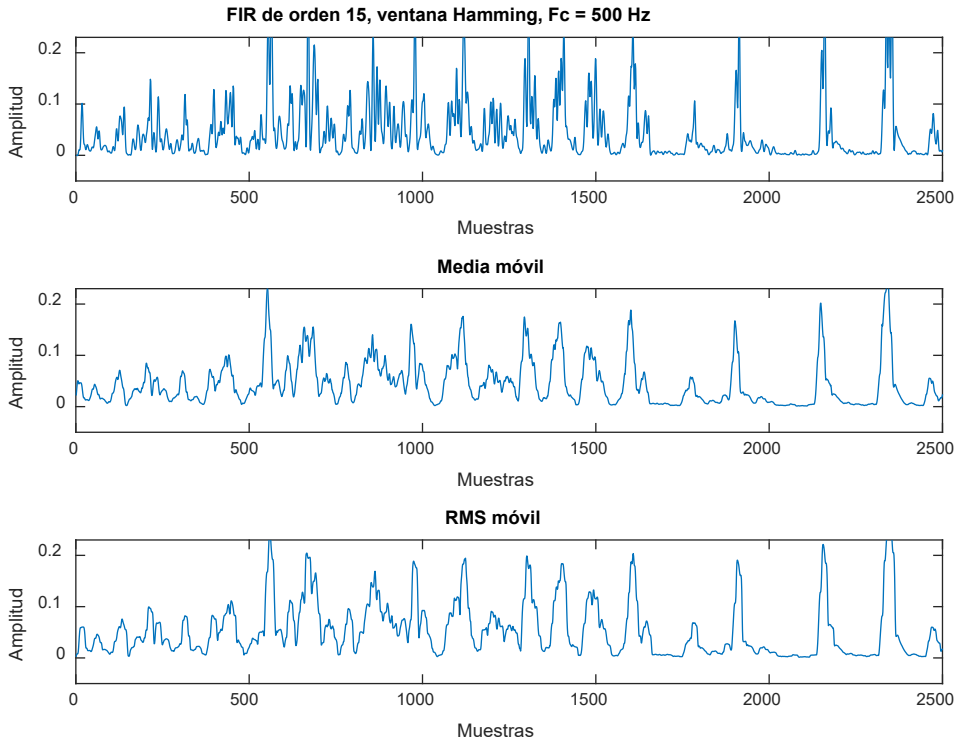

Figura 17: Envolvente resultante de cada método. Filtro FIR con 16 coeficientes y frecuencia de corte de 500 Hz, MA y RMS móvil con tamaño de ventana de 16 muestras.

En este caso, el filtro tiene peores prestaciones que los métodos de promediado. Este es un resultado esperado, ya que, si se quiere filtrar a partir de los 500 Hz, la frecuencia de corte debe situarse al menos un orden de magnitud por debajo.

Los coeficientes del filtro se han calculado siguiendo distintas funciones (formas de ventana) como la ventana Kaiser, triangular, o la de Hamming (figura 18); pero no se aprecia ninguna mejora en la calidad de la señal al cambiar de un tipo de ventana a otra.

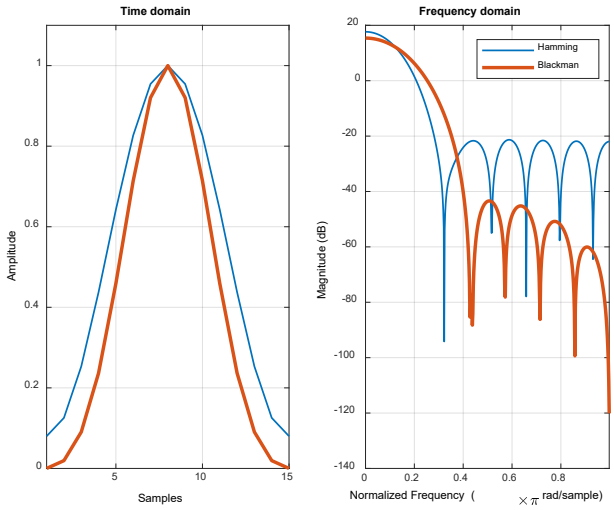

Figura 18: Comparativa de funciones de ventana en el dominio del tiempo y en el dominio de la frecuencia.

En frecuencia, se buscan lóbulos laterales de menor magnitud y mayor decrecimiento, como los que caracterizan a la ventana Blackman. Por otro lado, es preferible un lobulo principal mas estrecho como el de la ventana Hamming, que indica un filtrado mas selectivo en cuanto a frecuencias. Para este filtro, el resultado es similar con cualquiera de las dos ventanas (figura 19).

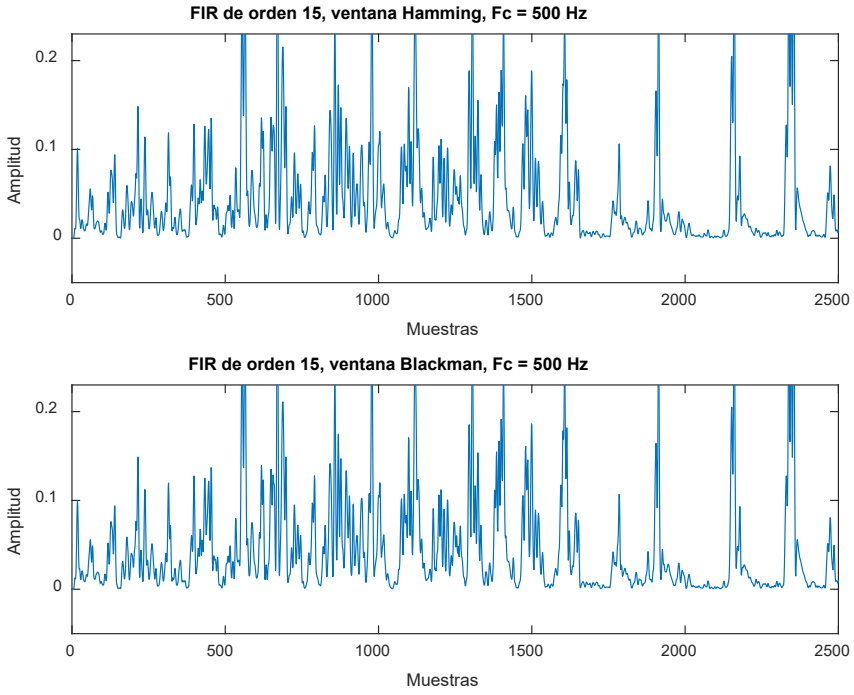

Figura 19: Comparativa de filtro FIR con mismas especificaciones de orden y frecuencia de corte, variando el tipo de ventana (Hamming y Blackman).

Orden 15, Frecuencia de corte 50 Hz.

Se han comparado los resultados con el mismo tamaño de ventana, esta vez situando la frecuencia de corte en 50 Hz para que filtro tenga un margen de actuación en frecuencia. La figura 20 muestra los resultados del Filtro y los métodos de promediado.

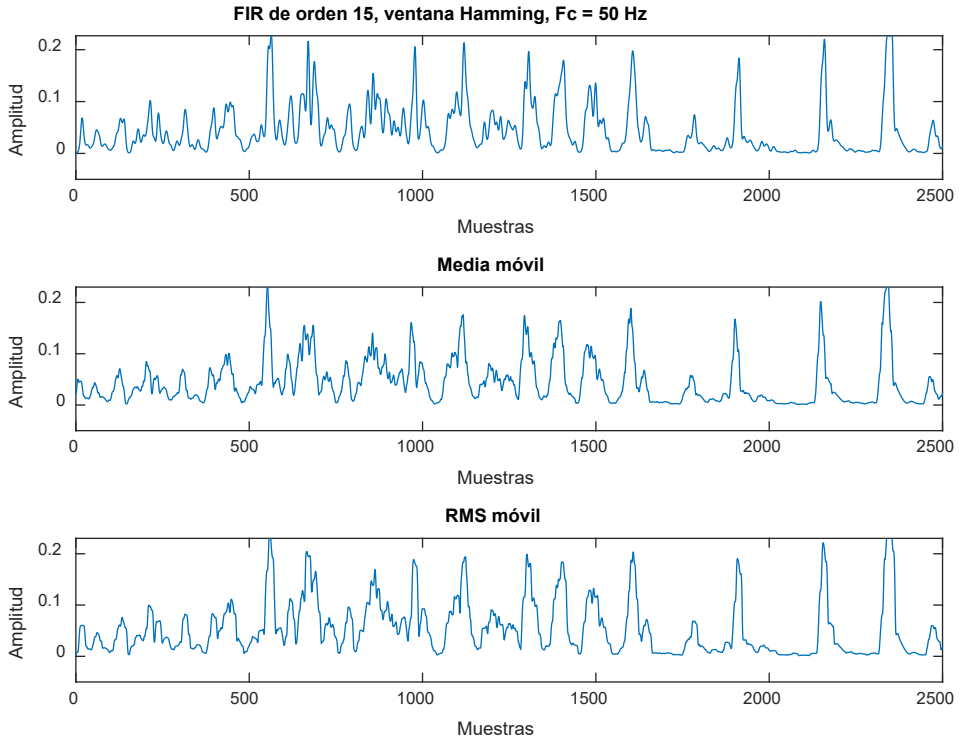

Figura 20: Envolvente resultante de cada método. Filtro FIR con 16 coeficientes y frecuencia de corte de 500 Hz, MA y RMS móvil con tamaño de ventana de 16 muestras.

Con la frecuencia de corte en 50 Hz, se hace visible el efecto del filtro FIR, que ya es comparable con los resultados de la MA y el RMS móvil. Además, en este caso si pueden apreciarse ligeras diferencias en los resultados del filtro cuando se cambia el tipo de ventana, y se ha añadido la ventana Kaiser al análisis (figura 21).

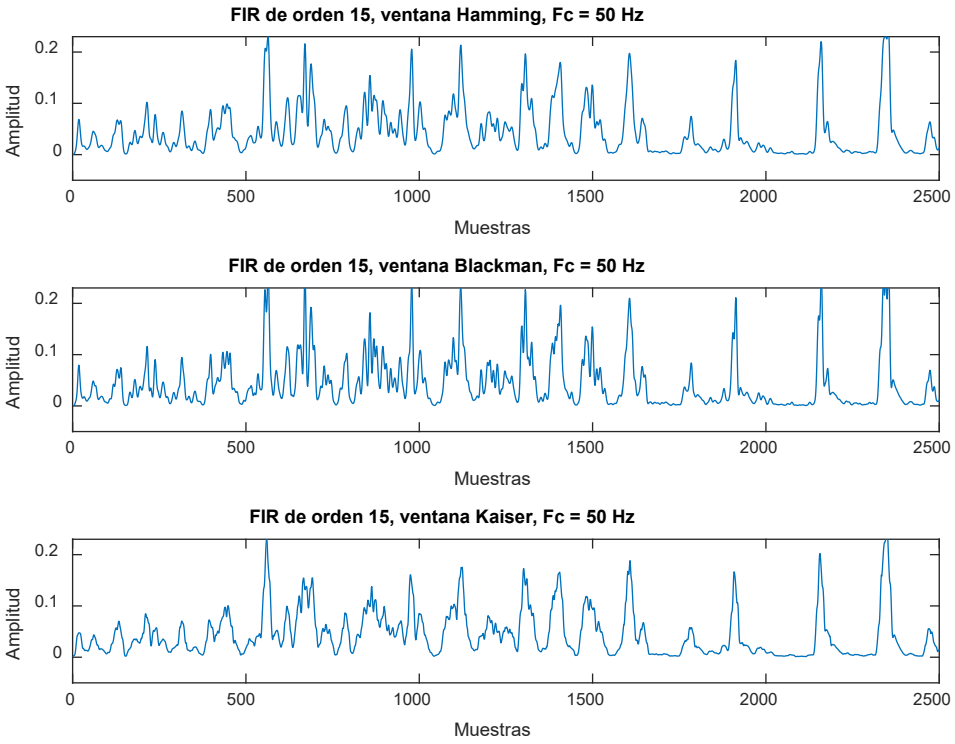

Figura 21: Envolvente resultante del Filtro FIR de 16 coeficientes y Fc de 50 Hz. Los coeficientes se han calculado con las ventanas Hamming, Blackman, y Kaiser, respectivamente.

El filtro cuyos coeficientes se han calculado con la ventana de Kaiser o la de Hamming tienen mejores resultados en cuanto a suavizado de la EMG. Es probable que se deba, en terminos de espectro frecuencial (figura 22), a un mejor compromiso entre la magnitud del lóbulo principal y su anchura, y el tamaño de los lobulos laterales, los cuales afectan al rizado de la señal filtrada. La diferencía entre los resultados es visible aunque reducida.

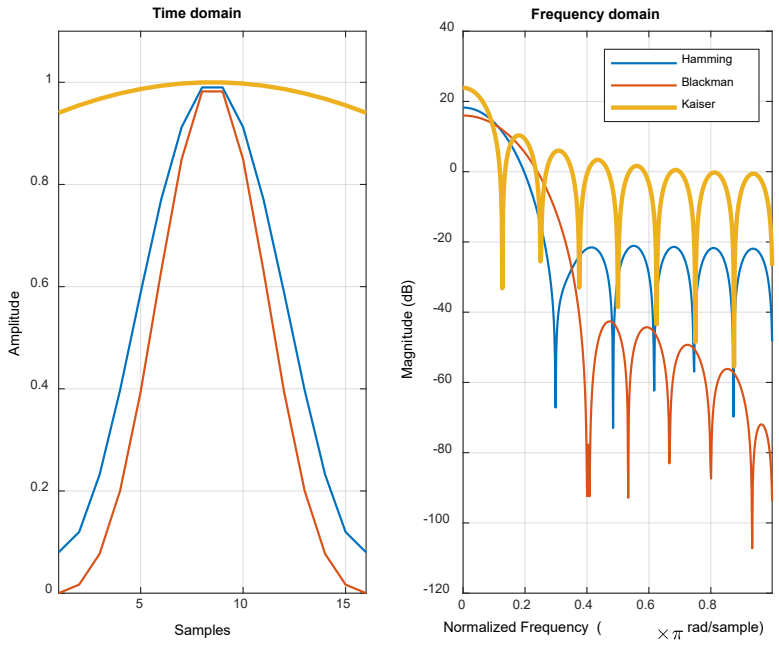

Figura 22: Función de la ventana Kaiser en el dominio del tiempo y en el dominio de la frecuencia contrastada con las ventanas Hamming y Blackman.

Orden 31, Frecuencia de corte 50 Hz.

La envolvente de salida del filtro de orden 15 y 16 coeficientes es aún mejorable, además, todavía hay un amplio margen en lo que a recursos de la FPGA se refiere, teniendo en cuenta que los DSPs se van a utilizar en bucle. Por lo tanto, se amplía la búsqueda de resultados con el siguiente número de coeficientes y tamaño de ventana que sea potencia de 2, es decir, 32 coeficientes y muestras por ventana. La figura 23 muestra los resultados obtenidos de cada método.

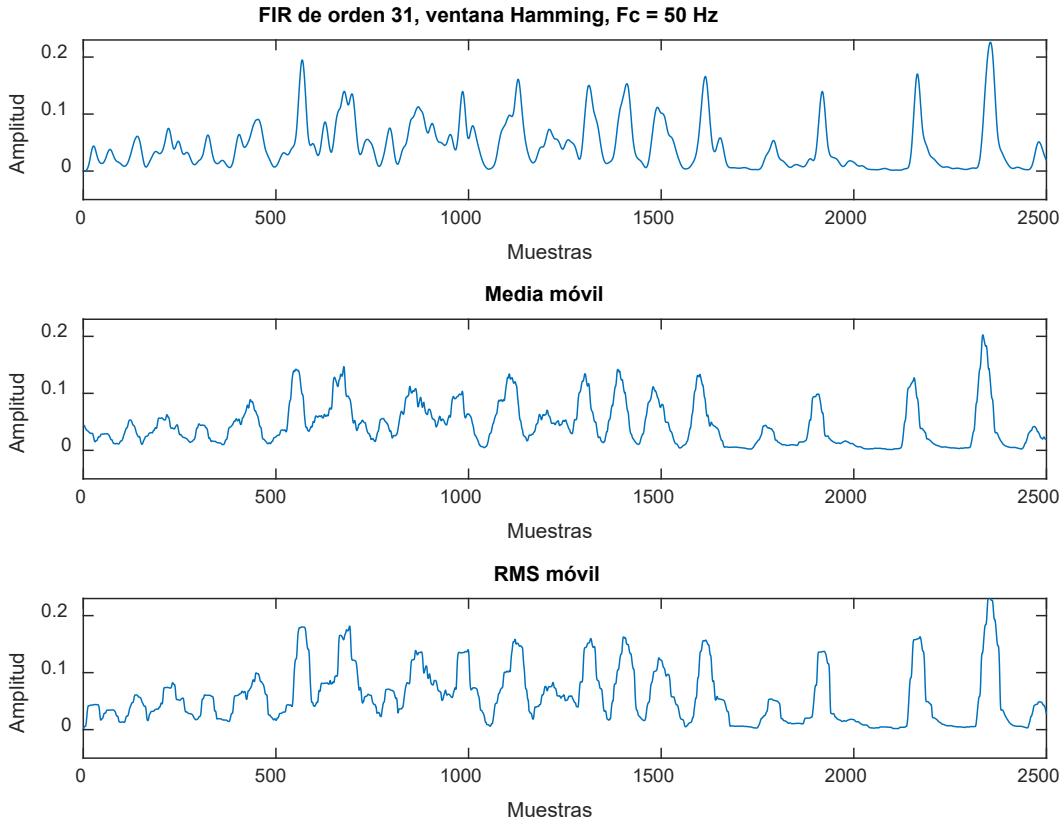

Figura 23: Envolvente resultante de cada método. Filtro FIR con 32 coeficientes y frecuencia de corte de 50 Hz, MA y RMS móvil con tamaño de ventana de 32 muestras.

Se observa que el filtro comienza a imponerse sobre la media móvil y el RMS móvil. La envolvente calculada a través del filtro no presenta el rizado que sí se aparece en algunos instantes utilizando los otros métodos. La figura 24 compara los efectos de las diferentes ventanas que se han analizado.

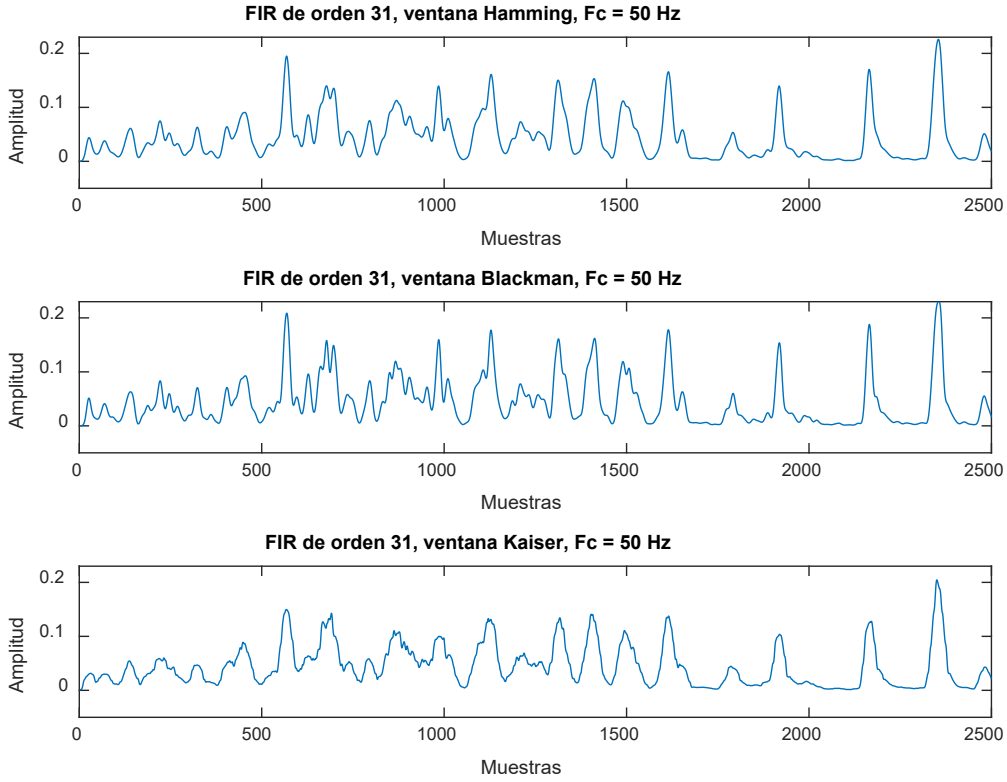

Figura 24: Envolvente resultante del Filtro FIR de 32 coeficientes y Fc de 50 Hz. Los coeficientes se han calculado con las ventanas Hamming, Blackman, y Kaiser, respectivamente.

La ventana de Hamming muestra mejores resultados al diferenciar picos de actividad muy próximos. La venta Kaiser destaca justo en lo contrario, es capaz de unificar pequeñas oscilaciones en un solo pico de activación, aunque también exhibe un rizado mayor debido a la amplitud de los lóbulos laterales de su espectro en frecuencia. Sin el análisis de un profesional en el campo de las EMG ni pruebas de campo, es difícil determinar qué ventana dará mejores resultados en la práctica. Se ha continuado con la comparativa de ventanas en las simulaciones posteriores.

Orden 63, Frecuencia de corte 50 Hz.

Las simulaciones con 64 coeficientes demuestran una importante mejora en cuanto al filtrado de las pequeñas oscilaciones. En las gráficas predominan los grandes picos de activación (figura 25).

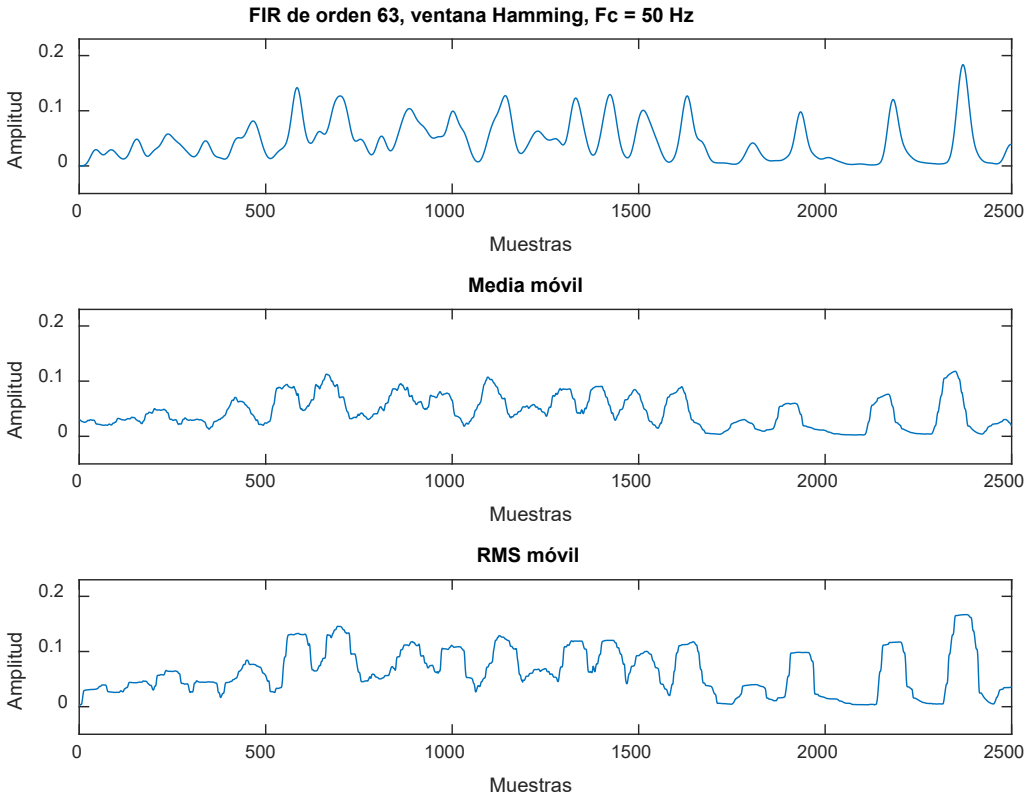

Figura 25: Envolvente resultante de cada método. Filtro FIR con 64 coeficientes y frecuencia de corte de 50 Hz, MA y RMS móvil con tamaño de ventana de 64 muestras.

Aún permanece un ligero rizado en la envolvente de la MA y el RMS, además la respuesta es más lenta frente a la del filtro FIR.

Al comparar el efecto de las diferentes ventanas, se continúa observando el fenómeno mencionado previamente. Las ventanas de Hamming y Blackman logran resaltar algunos picos de activación que la ventana de Kaiser amortigua (figura 26)

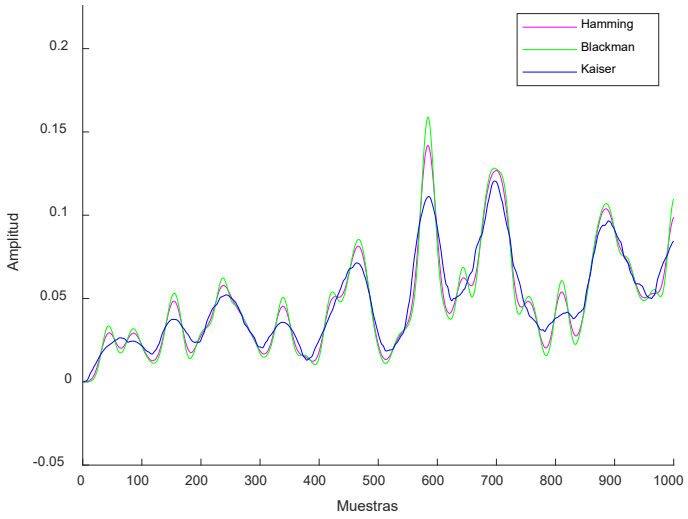

Figura 26: Comparativa de envolvente de señal EMG al aplicar distintas ventanas a un filtro FIR con frecuencia de corte en 50 Hz y de orden 63.

En esta ocasión, la ventana de Kaiser no ofrece ninguna ventaja, ya que las pequeñas oscilaciones que solía manejar con mayor eficacia han desaparecido también en las otras ventanas al incrementar el orden del filtro. Además, Hamming exhibe una mayor capacidad para atenuar los picos más pequeños que se presentan justo en la fase ascendente de los picos principales en comparación con la ventana Blackman.

Incrementar el orden del filtro a 254 y 255 coeficientes (una potencia de 2 inmediatamente superior) no aporta mejoras significativas si se compara con la cantidad adicional de recursos requeridos. Además, resulta en una respuesta más lenta en comparación con el filtro de orden 63 (figura 27), por lo que, finalmente, se implementará el filtro de orden 63.

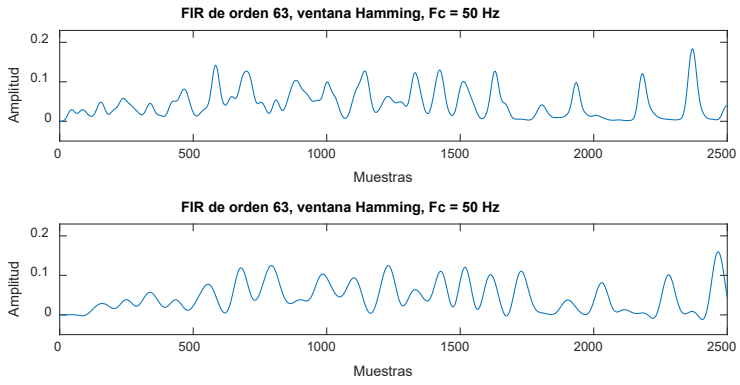

Figura 27: Comparación de envolventes calculadas con un filtro FIR de orden 63 y otro de orden 254.

Modelo de aproximación en Matlab.

Antes de empezar a diseñar el filtro en HLS para la FPGA, primero se ha realizado una aproximación del algoritmo en Matlab. Este modelo *behavioural* ha servido para incorporar la cuantificación a las simulaciones. El código puede encontrarse en GitHub [ver anexos].

El filtro implementado es de orden 63 y frecuencia de corte en 50 Hz, los coeficientes se han calculado nuevamente con la función ***fir1()*** y la ventana Hamming. Además, se ha aplicado la cuantificación descrita en el apartado de restricciones de área del filtro a cada tipo de dato, dejando un bit para las unidades y el resto para la parte decimal. No se utiliza bit de signo porque se ha asumido que los datos ya entran rectificados antes de ser re-cuantificados a 25 bits y los coeficientes son siempre positivos.

La figura 28 compara la envolvente de la señal EMG obtenida a partir del modelo *behavioural* con la obtenida a partir de la función ***filter()*** de Matlab.

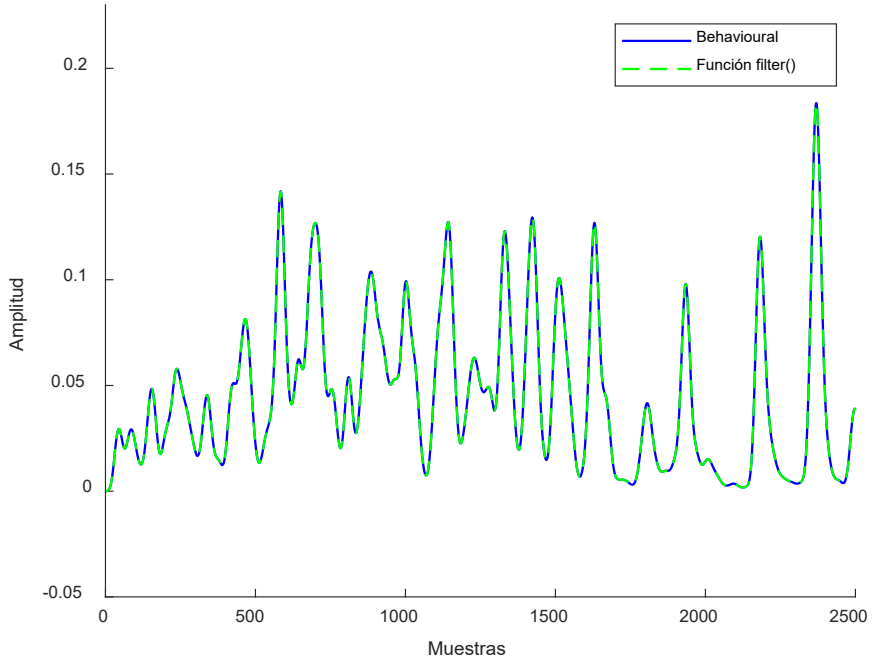

Figura 28: Envolvente de señal EMG calculada con la función filter superpuesta a la envolvente calculada por el código con cuantificación.

El algoritmo escrito en Matlab se aproxima en gran medida al resultado que se obtiene con la función `filter()`. El promedio del valor absoluto del error en cada muestra es de 7.9 e-06 con respecto la función `filter()`. Además, Matlab no ha reportado ningún error de desbordamiento en las operaciones con los datos cuantificados. El siguiente paso es implementar el algoritmo del filtro directamente en Vitis HLS en lenguaje C++.

Implementación final del filtro FIR en HLS

Los coeficientes calculados por la función fir1() se han exportado a un fichero `config.h` de C++ para instanciarlos en el código del filtro. En el mismo fichero, a partir de la librería "`ap_fixed.h`", también se han definido los tipos de datos que se van a usar para cuantificar las variables en coma fija. Otros parámetros como el tamaño del filtro también se definen en el mismo fichero, de esta forma solo habrá que modificar las definiciones en el archivo de configuración en caso de rediseñar el filtro FIR con otras características.

El código en C que describe el comportamiento del filtro está encapsulado en una función llamada `FIR_filter`, que está declarada en el archivo de encabezado `FIR.h` como una función sin retorno, con un parámetro de entrada representando las muestras de la señal EMG, y un parámetro de salida que corresponde a las muestras de la envolvente calculada.

La definición del filtro se encuentra en el archivo FIR.cpp y se compone de dos partes fácilmente diferenciables:

1. La primera parte consiste en un registro de desplazamiento que guarda en la primera posición el dato de entrada y desplaza todos los datos de entrada de instantes anteriores en una posición, descartando la muestra que estuviera en la última posición cuando llegue un nuevo dato.
2. La segunda parte del filtro está compuesta por un bucle de operaciones de suma y multiplicación para calcular la salida. En este bucle, se realizan las operaciones necesarias sobre los datos almacenados en el registro de desplazamiento para obtener la salida del filtro.

Además, tanto la entrada de muestras de la señal EMG como la salida de muestras de la envolvente se han definido como puertos de entrada y salida del protocolo AXI Stream *(Advanced eXtensible Interface*) mediante directivas específicas de Vitis HLS. Esto permitirá una integración más sencilla en el sistema completo y una conexión sincronizada con los demás módulos.

Para verificar el funcionamiento del modelo en C, se ha creado un *testbench* que realiza llamadas a la función `FIR_filter` con datos reales obtenidos de la base de datos [6] mencionada anteriormente. La salida del filtro generada por la función se compara con los resultados del modelo implementado en Matlab.

La razón por la que no se ha comparado directamente con la envolvente de la base de datos es que no se tiene conocimiento sobre la implementación ni la cuantificación utilizada para adquirirla, además, las señales procesadas de la base de datos no corresponden a las señales en crudo de la misma BB.DD, si no a zancadas aisladas efectuadas en una plataforma de fuerza.

Resultados de la síntesis y verificación del filtro FIR en HLS

En este apartado se muestran cuáles han sido los resultados de latencia y consumo de los recursos más relevantes en la FPGA, así como el error cuadrático medio entre los resultados del filtro en HLS con la referencia de Matlab. La síntesis se ha repetido aplicando diferentes directivas de optimización para estudiar su efecto en el consumo de área, latencia y el error cuadrático medio y máximo.

En una primera síntesis sin aplicar directivas de usuario, se observa una latencia estimada de 1410 ns que es bastante menor que los 125 µs dados por la ecuación (6). Además, el programa ha optimizado por si mismo los bucles del registro de desplazamiento y del acumulador aplicando un pipeline a cada uno de ellos. En cuanto al área, el consumo de FF (Flip-Flops) y LUTs es muy bajo. La cantidad de FF necesaria es mucho menor del 1% del total presente en la FPGA, al igual que ocurre con las LUTs, que apenas utilizan un 2% del total. Además, gracias a la cuantificación *utilizada*, es posible implementar el diseño con solo un bloque DSP, de los *80* disponibles en la FPGA. Por otra parte, la memoria BRAM que aparece se utiliza para almacenar las muestras del registro de desplazamiento, mientras que los coeficientes de los .

Tabla 4: Resultados de área y latencia de la síntesis sin directivas.

Módulos y bucles	Latencia (ciclos)	Latencia (ns)	BRAM	DSP	FF	LUT	Periodo de reloj mínimo estimado (ns)
Total	141	1410	1 (%1)	1 (%1)	199 (%0)	516 (2%)	6,508 ± 2,70
Registro de desplazamiento	65	650	-	0	15	62	
Acumulador	70	700	-	1	151	313	

Después de aplicar las directivas `ARRAY_PARTITION` y `UNROLL` al registro y bucle de desplazamiento, se observa una reducción en el periodo de reloj mínimo estimado respecto a la estimación de la primera síntesis, lo que significa que esta configuración se beneficia de un margen mayor de frecuencia de reloj a la que puede funcionar.

El *unroll* de factor 4 del registro de desplazamiento implica la transformación del bucle principal en 4 múltiples bucles similares, pero de menor longitud. Estos bucles, al operar en paralelo, ejecutan la misma función que el bucle original, pero con la ventaja de reducir la latencia total al paralelizar bucles de menos iteraciones; en concreto el desplazamiento del registro ha pasado de tener una latencia de 650 ns a solo 30 ns, reduciendo a su vez la latencia total del filtro. Para que esta técnica sea efectiva, la memoria que almacena los datos debe estar segmentada para permitir el acceso simultáneo a todos los elementos. De lo contrario, el `UNROLL` no sería completamente eficaz debido a posibles conflictos entre procesos que intenten acceder a recursos a través del mismo puerto.

Se observa en la tabla 5 que, como consecuencia de haber desenrollado el registro de desplazamiento, este se implementa a partir de Flip-Flops en lugar de hacer uso de un bloque de memoria RAM (BRAM); es por eso por lo que el consumo de FF respecto a los resultados de síntesis de la tabla 4 han aumentado considerablemente. Cada FF puede almacenar un bit y el registro de desplazamiento está compuesto por 64 datos (longitud del filtro) de un ancho de 24 bits (cuantificación de los datos de entrada) que resulta en un consumo de 1.536 FF solamente para almacenar los datos; con el almacenamiento de los coeficientes puede hacerse un análisis similar del cual se deduce que necesitaría 1.152 FF al margen de los Flip-Flops necesarios para implementar los procesos y el sincronismo. Por esto se deduce que en la primera síntesis los coeficientes también estaban almacenados en la BRAM.

Tabla 5: Resultados de la síntesis tras aplicar UNROLL y ARRAY_PARTITION al registro de desplazamiento.

Módulos y bucles	Latencia (ciclos)	Latencia (ns)	BRAM	DSP	FF	LUT	Periodo de reloj mínimo estimado (ns)
Total	73	730	0	1/80	3228 (%9)	633 (3%)	5,118 ± 2,70
Registro de desplazamiento (Desenrollado completo)	3	30	-	0	3053	47	
Acumulador	70	700	-	1	175	586	

La misma técnica se puede aplicar al bucle del acumulador. Para ello, es necesario no solo aplicar `ARRAY_PARTITION` al registro de desplazamiento, sino también al espacio de memoria que almacena los coeficientes del filtro. En este caso, ni la partición de los coeficientes ni el desenrollamiento del bucle es completo, ya que, al tratarse de operaciones de sumas y multiplicaciones, una paralelización completa implicaría un consumo elevado de recursos, entre los cuales se incluyen los DSP, de los cuales hay un número reducido en la FPGA. En este bucle también se ha aplicado la directiva PIPELINE para segmentar los procesos y permitir que se pueda realizar el producto de cada muestra por el coeficiente correspondiente a la vez que se suman los productos de instantes previos.

Esta configuración ve aumentado el periodo mínimo de reloj (o reducida la frecuencia máxima) respecto a las otras dos configuraciones, pero aun sumando la incertidumbre de los 2,7 ns, el periodo mínimo queda por debajo de los 10 ns (10 MHz) del reloj que se va a usar. Por otro lado, la latencia del sumador se ha reducido de 700 ns a 220 ns, que es algo más de la cuarta parte de 700 debido al tiempo de *throughput* (tiempo que transcurre desde que entra un dato hasta que puede introducirse el siguiente). Sin la directica de *pipeline*, el *throughput* y la latencia total sería mayor.

Respecto al consumo de recursos, si se compara la tabla 6, que reúne los resultados de la síntesis con optimizaciones de tiempo, con la tabla 4, de la síntesis sin directivas, se observa una relación inversamente proporcional entre la latencia y el uso de DSP, Flip-Flops y LUTs, ya que la segmentación de procesos y la paralelización de bucles requiere un mayor número de componentes.

Tabla 6: Resultados de la síntesis con todas las optimizaciones aplicadas en cada parte del código.

Módulos y bucles	Latencia (ciclos)	Latencia (ns)	BRAM	DSP	FF	LUT	Periodo de reloj mínimo estimado (ns)
Total	33	330	0	4/80	3781 (%10)	1751 (9%)	6,945 ± 2,70
Registro de desplazamiento (Desenrollado completo)	65	650	-	0	3059	61	
Acumulador (Desenrollado de factor 4)	22	220	-	4	684	1462	
Suma de las 4 partes	6	60	-	0	38	228	

Arquitectura global.

El filtro FIR es un módulo que forma parte de un sistema mayor; en este apartado se explican las características de la transferencia de datos entre la plataforma de adquisición y el filtro, y se describe la arquitectura global.

La plataforma ADS1298R (figura 7) cuenta con hasta 8 canales que muestrean de manera simultánea y envía los datos cuantificados en 24 bits por protocolo SPI hacia un buffer o registro de desplazamiento. La comunicación SPI transmite 9 tramas de 24 bits en serie, la primera es el *status word* que transporta información de registros de estado del ADS1298R y las 8 tramas de 24 bits restantes transportan los datos muestreados de los respetivos canales. En el buffer de llegada se extiende el signo a 27 bits, tanto al *status word* como a los datos de los canales, y añade un identificador numérico de 4 bits en la parte más significativa para poder diferenciar los canales y el *status word* (figura 29).

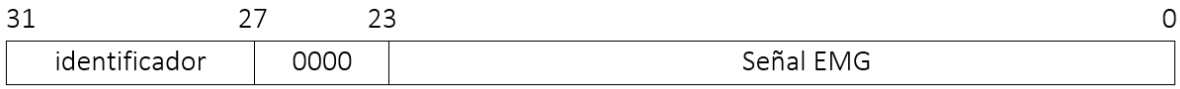

Figura 29: Ejemplo de trama de datos que almacena el buffer receptor de la comunicación SPI. La señal EMG se codifica en los primeros 24 bits (del bit 0 al bit 23) y se extiende el signo hasta el bit 27. El identificador del canal al que corresponde la señal se codifica en los 4 bits más significativos (del bit 27 al bit 31). El resultado es una trama de 32 bits compatible con el bus AXI que se utiliza en la arquitectura global.

La transmisión de datos entre el buffer y el módulo de filtrado se realiza por el protocolo AXIS o AXI Stream de 32 bits de ancho de trama. El bloque de filtrado puede implementar tantas réplicas del filtro como canales hay conectados, o bien utilizar un buffer con el contenido de los canales para ser procesados por el mismo filtro uno a uno; ambas opciones son viables en términos de latencia y de consumo de recursos por los amplios márgenes que se han observado en el apartado anterior.

Respecto al resto del sistema, la figura 30 representa, en un diagrama de bloques, los principales módulos y la conexión simbólica entre ellos, abstrayéndose, por ejemplo, de las señales de control y sincronización de los buses.

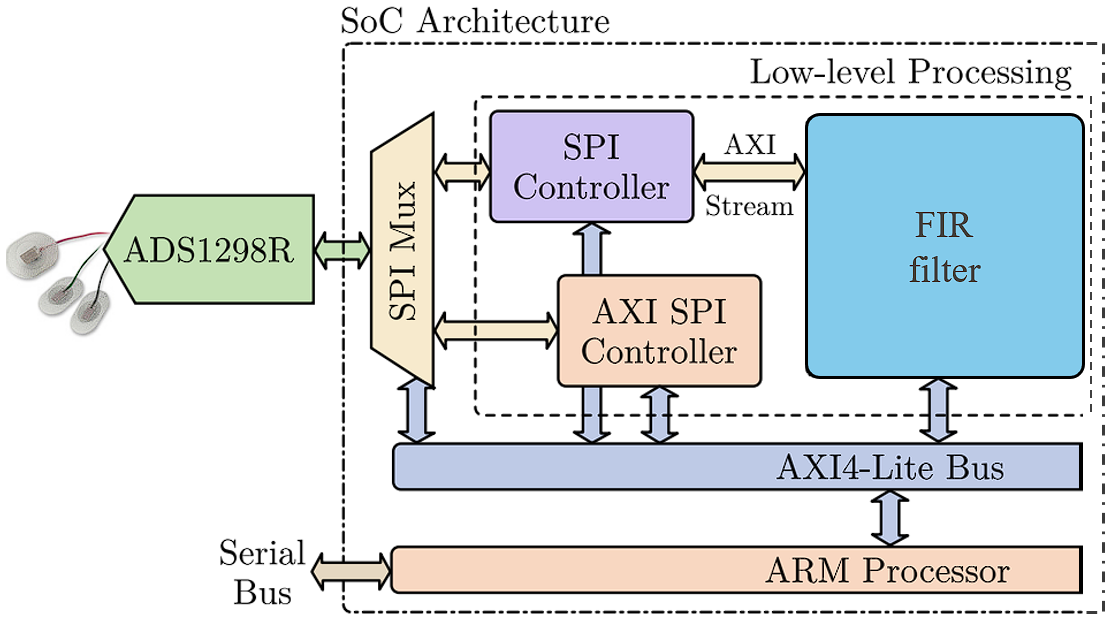

Figura 30: Diagrama de bloques de la arquitectura SoC (System on Chip) del sistema. Imagen adaptada de [10].

La arquitectura también implementa un bus AXI4 – Lite para las señales de control y el intercambio de datos entre los diferentes módulos.

La envolvente de las señales EMG procesada por el filtro FIR, en este caso no se le aplica ningún proceso de umbralización (*Thresholding*), sin embargo, sería posible añadir un nuevo bloque tras la salida del filtro para determinar el inicio de la activación muscular y llevar la información al lazo de control de un exoesqueleto.

Latencia real del filtro.

Para la incorporación del filtro FIR diseñado a la arquitectura mostrada en el apartado anterior, se ha realizado la síntesis e implementación para una frecuencia de reloj de 50 MHz, es decir un periodo de reloj de 20 ns, que es la frecuencia utilizada en el resto de la arquitectura. Para esa frecuencia, los resultados de la síntesis muestran que la latencia del filtro es de 1,74 µs (Figura 31).

Periodo de reloj objetivo	Mínimo estimado	Latencia del filtro máxima
20 ns	17,473 ns	1,74 µs

Tabla 7: Resultados de la síntesis del filtro FIR para una frecuencia de reloj de 50 MHz.

Una vez instanciado el filtro en el diseño de bloques de Vivado junto a los demás componentes, se ha configurado la FPGA y se ha conectado a la plataforma de adquisición. Después, por medio de un analizador lógico embebido en la FPGA (Integrated Logic Analyzer, ILA), se ha podido analizar la forma de onda de la salida del filtro ante una entrada de ruido con el fin de medir la latencia real del filtro FIR implementado en este TFG.

En la Figura 31 se muestran en amarillo se muestran las señales que provienen del módulo del sistema de adquisición por SPI, y que se envían por el bus AXIS hacia el filtro cuando la señal TVALID se activa a nivel alto. Por otro lado, en verde se muestran las señales relacionadas con el filtro FIR implementado.

Mientras la señal TREADY del filtro esté habilitada, se almacenan 9 tramas de 32 bits, que corresponden con los bloques de datos del ADS1298. A continuación, se realiza el cálculo de la envolvente dentro del filtro y el resultado se escribe en la señal TDATA. Cuando la señal TVALID se habilita, el dato del bus TDATA es un dato válido.

Se puede comprobar con los cursores que la latencia del filtro es de 87 ciclos de reloj, que con el reloj de 50 MHz corresponde a una latencia absoluta de 1,74 µs, de acuerdo con los resultados mostrados anteriormente en la Tabla 7.

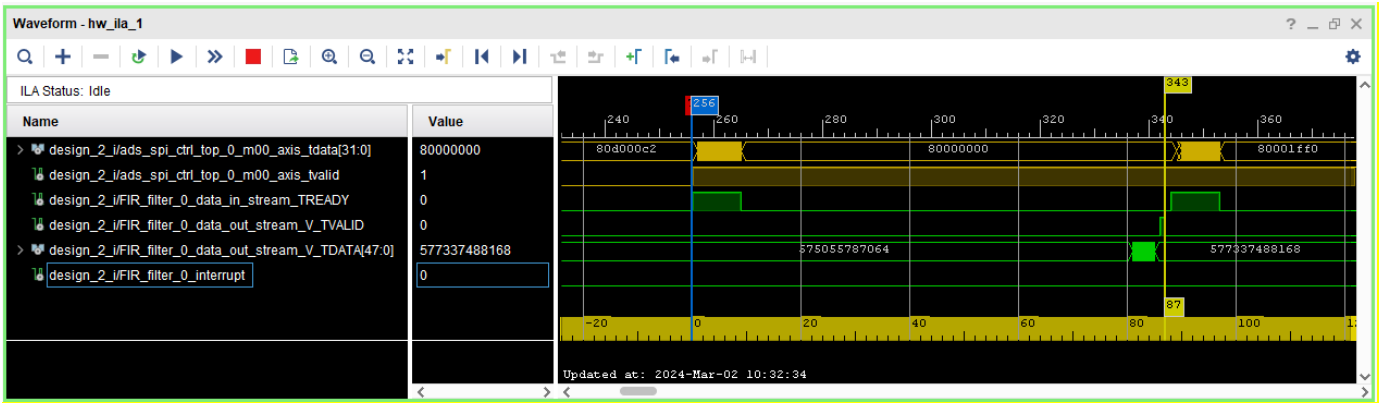

Figura 31: Captura del ILA que muestra el intervalo de tiempo entre la entrada y la salida del filtro.

6.Conclusiones

Este trabajo se ha enfocado en la implementación del pre-procesamiento de señales de electromiografía (EMG) con el objetivo de extraer su envolvente, lo que permite trabajar con una representación más clara y útil en lugar de lidiar con las señales EMG en bruto, las cuales suelen tener una baja relación señal a ruido.

Concretamente, se ha desarrollado un filtro digital de respuesta al impulso finita (FIR) debido al control preciso que ofrece sobre la frecuencia de corte en comparación con técnicas como la media móvil y la RMS móvil. Además, los filtros FIR son más versátiles y reutilizables que un filtro IIR, ya que permite ajustar sus parámetros para adaptarse a diversas necesidades sin comprometer su estabilidad.

Mediante la herramienta de Matlab se han evaluado diferentes configuraciones del filtro, entre ellas, variaciones en el orden del filtro, la frecuencia de corte e incluso el tipo de ventana que afecta al cálculo de los coeficientes. Esta evaluación se ha realizado utilizando una base de datos de señales EMG [6], y se ha concluido que el que mejor comportamiento tiene es un filtro FIR utilizando una ventana de Hamming, comparado con otro tipo de ventanas como Kaiser y Blackman.

Para llevar a cabo la implementación se ha hecho uso de una FPGA, por su capacidad de concurrencia de procesos en el hardware, y el uso de las herramientas de síntesis de alto nivel (HLS) para la implementación del filtro. El código en C que describe el comportamiento del filtro FIR ha superado satisfactoriamente el banco de pruebas, con un error promedio de 9,49e-6 y un error por muestra menor del 1e-3. Además, los resultados de la síntesis indican un consumo mínimo de recursos, proporcionando margen suficiente para replicar este filtro para los diferentes canales, y garantizando al mismo tiempo una latencia viable que cuenta con un amplio margen respecto a la latencia máxima.

La síntesis con directivas de usuario de Vitis HLS ha puesto de manifiesto el impacto que estas tienen en la utilización de recursos, tanto en cantidad como en tipos, así como en la latencia total y de iteración. Sin embargo, debido a la relativa sencillez del filtro implementado, la diferencia en comparación con la síntesis sin directivas no es significativa. Para realizar un análisis más preciso del efecto de las directivas, sería necesario probarlas en un algoritmo más complejo, como el de una transformada rápida de Fourier (FFT).

Por último, la integración del diseño con el resto del sistema supone un pequeño cambio en la arquitectura, para adaptarlo con el formato de entrada de datos. Las pruebas realizadas con el conjunto muestran que el diseño del filtro funciona a falta de validar el sistema con muestras sintéticas.

Para el desarrollo práctico y teórico de este trabajo ha sido necesaria una formación complementaría en el uso del lenguaje de alto nivel para descripción de hardware, conocido como HLS, que no se había abordado en el Grado. También ha requerido una profundización en algunos conceptos que sí se han impartido, como la cuantificación de señales digitales o el diseño de filtros digitales. Como resultado, se ha ampliado el conocimiento en el ámbito del diseño electrónico digital y se ha adquirido una mayor comprensión de los conceptos teóricos involucrados en este trabajo.

7. Bibliografía

[1]. Personas con discapacidad por dificultad en el desplazamiento, tipo de discapacidad y tipo de indicador. *INE*. [consulta: 23 septiembre 2023]. Disponible en: https://www.ine.es/jaxi/Datos.htm?path=/t22/e308/meto_05/modulo/base_2011/2002/l0/&file=00025.px#!tabs-grafico.

[2]. INEbase / Sociedad /Salud /Encuestas de discapacidades / Últimos datos. *INE* [en línea], [sin fecha]. [consulta: 23 septiembre 2023]. Disponible en: https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176782&menu=ultiDatos&idp=1254735573175.

[3]. CSIC Nota de prensa, lunes 13 de febrero de 2017. https://www.csic.es/sites/www.csic.es/files/13febrero20017_exoesqueleto_0.pdf

[4]. GUZMÁN-MUÑOZ, E., MÉNDEZ-REBOLLEDO, G., GUZMÁN-MUÑOZ, E. y MÉNDEZ-REBOLLEDO, G., 2018. Electromiografía en las Ciencias de la Rehabilitación. *Revista Salud Uninorte*, vol. 34, no. 3, ISSN 0120-5552.

[5]. CAMILA R. CARVALHO, J. MARVIN FERNÁNDEZ, ANTONIO J. DEL-AMA, FILIPE OLIVEIRA BARROSO, JUAN C., y MORENO, 2023. Review and Performance Comparison of EMG Onset Detection Methods Towards Real-Time Control of Robotic Exoskeletons.

[6]. MOREIRA, L.C., FIGUEIREDO, J., FONSECA, P., VILAS-BOAS, J.P. y SANTOS, C., 2021. Lower limb kinematic, kinetic, and EMG data during walking motion under controlled speeds [en línea]. 1 abril 2021. S.l.: figshare. [Consulta: 29 septiembre 2023]. Disponible en: https://springernature.figshare.com/collections/Lower_limb_kinematic_kinetic_and_EMG_data_during_walking_motion_under_controlled_speeds/4923162/1.

[7]. zynq-7000-product-selection-guide.pdf. [en línea], [sin fecha]. [consulta: 25 febrero 2024]. Disponible en: https://www.xilinx.com/content/dam/xilinx/support/documents/selection-guides/zynq-7000-product-selection-guide.pdf.

[8]. ads1298.pdf. [en línea], 2015. [consulta: 22 febrero 2024]. Disponible en: https://www.ti.com/lit/ds/symlink/ads1298.pdf?ts=1708427872352&ref_url=https%253A%252F%252Feu.mouser.com%252F.

[9]. ISABEL M. VEGA RODRÍGUEZ, MÓNICA A. VALLEJO VELÁSQUEZ, y FREDDY BOLAÑOS MARTÍNEZ, 2018. Use of trapezius muscle activity for stress determination: A literature Review. [en línea], Disponible en: http://ref.scielo.org/jbh8gq.

[10]. Navarro, Víctor & Nieto, Rubén & Fernández, Pedro & Hernández, Álvaro & del-Ama, Antonio & Borromeo, S.. (2023). SoC Architecture for Acquisition and Processing of EMG Signals. 1-6. 10.1109/DCIS58620.2023.10335993.

[11]. DANIEL MAYOR TOMILLO, [OCTUBREDE 2016]. Trabajo Fin de Grado. Diseño de filtros digitales FIR mediante técnicas de computación evolutiva y estudio de su aplicación al procesado de señales biomédicas. [en línea], Disponible en: https://uvadoc.uva.es/bitstream/handle/10324/20958/TFG-G2270.pdf?sequence=1&isAllowed=y.

[12]. FISIOTERAPIA, Á.T., 2017. ¿Tratamiento pasivo o tratamiento activo en Fisioterapia? *Ángel Troncoso Fisioterapia* [en línea]. [consulta: 23 septiembre 2023]. Disponible en: https://angeltroncosofisioterapia.com/tratamiento-pasivo-o-tratamiento-activo/.

[13]. Diseño de filtros IIR - Matlab & Simulink - MathWorks España. [en línea], [sin fecha]. [consulta: 5 octubre 2023]. Disponible en: https://es.mathworks.com/help/signal/ug/iir-filter-design.html.

[14]. FPGA Programming. [en línea], [sin fecha]. [consulta: 20 marzo 2023]. Disponible en: https://es.mathworks.com/discovery/fpga-programming.html.

[15]. Xilinx - Adaptable. Intelligent | together we advance_. *Xilinx* [en línea], [sin fecha]. [consulta: 20 marzo 2023]. Disponible en: https://www.xilinx.com/.

[16]. fpga4fun.com - JTAG. [en línea], [sin fecha]. [consulta: 20 marzo 2023]. Disponible en: https://www.fpga4fun.com/JTAG.html.

Anexos.

Enlace al código del filtro y el banco de pruebas en Github:

https://github.com/AlexLDP01/EMG_signal_filtering/tree/master/Vitis_HLS/FIR_v2_opt/src