UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA
DE SISTEMAS ELECTRÓNICOS
TRABAJO FIN DE MÁSTER

DISEÑO DE UN SISTEMA DE FILTRADO DE AUDIO BASADO EN FFT PARA SU IMPLEMENTACIÓN EN PLACAS DE DESARROLLO NEXYS A7

CARLOS ALBACETE FUENTES
2024

MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS ELECTRÓNICOS

TRABAJO FIN DE MÁSTER

Título: Diseño de un sistema de filtrado de audio basado en FFT para su implementación en placas de desarrollo Nexys A7.
Autor: D. Carlos Albacete Fuentes
Tutor: D. Juan Antonio López Martín
Ponente: D. Juan Antonio López Martín
Departamento: Ingeniería electrónica

MIEMBROS DEL TRIBUNAL

Presidente: D. ……

Vocal: D. ……..

Secretario: D. ……..

Suplente: D. ……..

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: ……

Madrid, a de de 2024

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS ELECTRÓNICOS
TRABAJO FIN DE MÁSTER

DISEÑO DE UN SISTEMA DE FILTRADO DE AUDIO BASADO EN FFT PARA SU IMPLEMENTACIÓN EN PLACAS DE DESARROLLO NEXYS A7

CARLOS ALBACETE FUENTES

2024

RESUMEN

El presente Trabajo de Fin de Máster se centra en el desarrollo de un sistema de procesamiento de audio implementado en la placa de desarrollo Nexys A7, basada en una FPGA de la familia Artix-7. La función de este sistema es la grabación, ecualización y reproducción con baja latencia de señales de audio. La ecualización, una de las técnicas más comunes de procesamiento de audio dentro de la industria musical, consiste en alterar la amplitud de las distintas frecuencias que componen el espectro de la señal de audio. De esta manera, se aumenta o disminuye el volumen de las frecuencias deseadas, manipulando el timbre de los sonidos.

Para grabar y reproducir la señal de audio se ha recurrido a los componentes ya incorporados en la placa de desarrollo. El micrófono codifica la señal en *Pulse Density Modulation* (PDM), que representa señales analógicas mediante señales digitales de 1 bit compuestas por pulsos de una anchura fija que se suceden con distinta frecuencia, mientras que la salida de audio consiste en un filtro paso bajo capaz de transformar una señal digital codificada en PDM o *Pulse Width Modulation* (PWM) a una señal analógica reproducible por un altavoz.

Una vez registrada la señal de audio, existen múltiples opciones para modificar su respuesta en frecuencia, siendo los filtros digitales una de las más comunes. Sin embargo, para lograr una manipulación más precisa y flexible del espectro de frecuencias, la señal se transforma al dominio de la frecuencia utilizando la Transformada Rápida de Fourier (FFT) y luego se reconvierte al dominio del tiempo mediante la Transformada Inversa (IFFT). La FFT e IFFT se refieren al conjunto de algoritmos que permiten el cálculo de la Transformada de Fourier Discreta (DFT) y su inversa (IDFT) de forma eficiente. A lo largo del trabajo, se ha estudiado en profundidad este algoritmo y sus implicaciones a la hora de implementarlo, de modo que pueda ser adaptado y optimizado según las necesidades del proyecto.

El empleo de la FFT para ecualizar la señal conlleva una serie de implicaciones que han sido abordadas a lo largo del trabajo. Las más destacadas son, en primer lugar, que la FFT no puede aplicarse directamente sobre la señal PDM, por lo que es necesario convertirla previamente a *Pulse Code Modulation* (PCM), que representa la señal con N bits. En segundo lugar, el resultado de la FFT se genera en *bit-reversed order* por lo que es necesario adaptar tanto el ecualizador como la entrada de la IFFT para procesar los datos correctamente y evitar el uso de módulos adicionales para el reordenamiento de datos.

Además de la ecualización y la FFT, este trabajo aborda todo el tratamiento necesario para la señal de audio, desde su muestreo inicial hasta su modulación final para la salida, centrándose en la optimización de área y consumo, así como el desarrollo de una interfaz hombre-máquina sencilla y efectiva basada en los elementos disponibles en la placa de desarrollo.

PALABRAS CLAVE

FPGA, DSP, FFT, IFFT, Ecualizador, Diezmado, Interpolación, FIR, CIC, PDM, PWM, PCM, ADC, DAC, Modulador Sigma-Delta, IP Core, Throughput, Integrador.

SUMMARY

The present Master's Thesis focuses on the development of an audio processing system implemented on the Nexys A7 development board, based on an Artix-7 family FPGA. The purpose of this system is to equalize audio signals captured by a microphone for low-latency playback. Equalization, one of the most common audio processing techniques in the music industry, involves altering the amplitude of the different frequencies that make up the audio signal spectrum. This allows for the increase or decrease of the volume of specific frequencies, thereby manipulating the timbre of the sounds.

To record and play back the audio signal, the components already integrated into the development board were used. The microphone encodes the signal in Pulse Density Modulation (PDM), representing analog signals as 1-bit digital signals composed of pulses with a fixed width that occur at varying frequencies. The audio output consists of a low-pass filter capable of converting a digital signal encoded in PDM or Pulse Width Modulation (PWM) into an analog signal that can be played through a speaker.

Once the audio signal is recorded, there are multiple options for modifying its frequency response, with digital filters being one of the most common. However, to achieve a more precise and flexible manipulation of the frequency spectrum, the signal is transformed into the frequency domain using the Fast Fourier Transform (FFT) and then converted back to the time domain using the Inverse Fast Fourier Transform (IFFT). The FFT and IFFT refer to the set of algorithms that efficiently compute the Discrete Fourier Transform (DFT) and its inverse (IDFT). Throughout this work, this algorithm and its implications for implementation have been thoroughly studied, allowing for adaptation and optimization according to the project's needs.

The use of FFT to equalize the signal involves a series of considerations that have been addressed throughout this work. The most notable are, first, that the FFT cannot be directly applied to the PDM signal, so it must first be converted to Pulse Code Modulation (PCM), which represents the signal with N bits. Second, the FFT result is generated in bit-reversed order, making it necessary to adapt both the equalizer and the IFFT input to correctly process the data and avoid the need for additional modules for data reordering.

In addition to equalization and FFT, this work covers the entire processing of the audio signal, from its initial sampling to its final modulation for output, with a focus on optimizing area and power consumption. It also involves the development of a simple and effective human-machine interface based on the elements available on the development board.

KEYWORDS

FPGA, DSP, FFT, IFFT, Equalizer, Decimation, Interpolation, FIR, CIC, PDM, PWM, PCM, ADC, DAC, Sigma-Delta Modulator, IP Core, Throughput, Integrator.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN Y OBJETIVOS

En este capítulo se presenta brevemente el contexto y la necesidad de este trabajo, junto con un breve resumen de las distintas partes que componen el sistema diseñado. Además, se enumeran los objetivos y las especificaciones que han servido como referencia para las decisiones de diseño tomadas a lo largo del trabajo.

1.1.INTRODUCCIÓN

En sus inicios, el proceso de grabación de audio, ya fuese mediante dispositivos acústicos, eléctricos o magnéticos, era puramente analógico. Esto conlleva una serie de limitaciones en la manipulación y transmisión del audio, que requiere de una serie de circuitos electromecánicos cuyo alcance estaba limitado a estudios de grabación profesionales. Aunque se había teorizado con la posibilidad de digitalizar el audio, no fue hasta 1962 que se llevó a cabo la primera grabación y transmisión digital de audio, gracias a los desarrollos en semiconductores y ADCs de alta velocidad [1].

A partir de la aparición del Compac Disc en 1982 [2], la relevancia del audio digital frente al analógico no ha hecho más que aumentar a lo largo de los años, apareciendo cada vez formatos de audio más comprimidos y de mayor calidad, como .mp3 o .flac, que han facilitado el almacenamiento y la distribución masiva de música. A medida que esto ocurre, se han ido popularizando cada vez más las estaciones de trabajo de audio digital (EADs), abaratando y aportando flexibilidad y eficiencia al proceso la grabación y manipulación de audio.

Muchas de las técnicas de manipulación de audio digital requieren una transformación de la señal para pasar del dominio del tiempo al de la frecuencia y viceversa, la cual se realiza mediante la FFT e IFFT. Estos algoritmos requieren una gran cantidad de operaciones y capacidad de almacenamiento para los valores intermedios y coeficientes, por lo que, si la operación es muy común, puede conllevar un gran consumo de recursos y tiempo en un procesador DSP, ya que por lo general está limitado a una ejecución secuencial de instrucciones. Sin embargo, si se realiza una implementación hardware del algoritmo, se abre la posibilidad de ejecutar operaciones en paralelo y de alcanzar un alto grado de *pipelining*, logrando mejoras en el rendimiento y en el *throughput* de varios órdenes de magnitud [3].

En este trabajo, se ha implementado un circuito digital en una placa de desarrollo basada en FPGA, capaz de grabar, procesar y ecualizar audio en tiempo real, minimizando el área y el consumo de potencia. Para ello, además de utilizar la FFT, IFFT y el ecualizador, se han diseñado distintos módulos que adaptan las características de la señal según lo requieran las diferentes partes del circuito. En la Figura 1.1, se muestra un diagrama de bloques del circuito completo. En primer, lugar el audio es captado por el micrófono de la placa, que lo modula en *Pulse Density Modulation* (PDM) a 2.5 MHz. Posteriormente, esa señal es codificada en un tipo de modulación adecuada para la entrada de la FFT, en este caso, *Pulse Code Modulation* (PCM) de 16 bits con una frecuencia de muestreo de 48 kHz, mediante diezmado. Tras ecualizar la señal en el dominio de la frecuencia, se vuelve a transformar al dominio del tiempo mediante la IFFT. Finalmente, la salida de audio, compuesta por un filtro paso bajo analógico y un puerto *jack*, requiere de nuevo una señal PDM, la cual se obtiene mediante la interpolación de la señal PCM y un modulador Sigma-Delta.

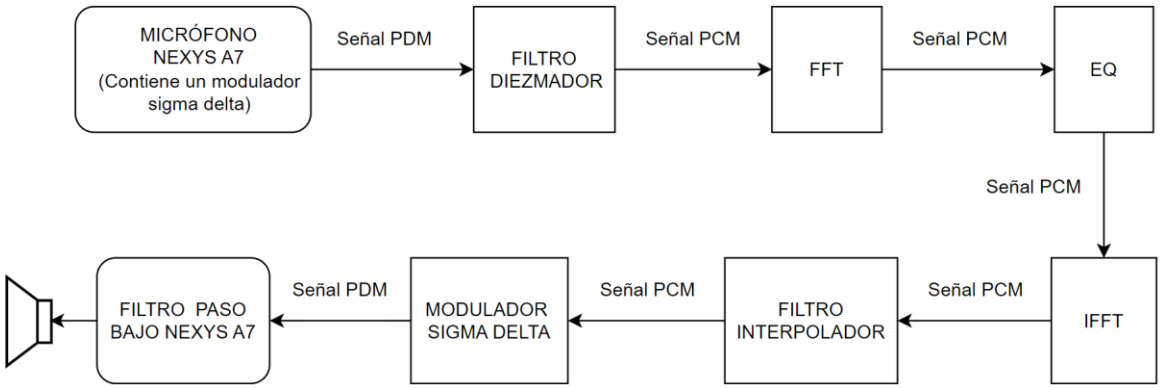

Figura 1.1. Diagrama de bloques del circuito completo.

1.2.OBJETIVOS Y ESPECIFICACIONES

El objetivo principal de este trabajo es implementar un sistema de grabación y ecualización de audio en tiempo real completamente funcional en la placa de desarrollo Nexys A7. No obstante, este objetivo final ha de ser concretado en especificaciones y objetivos más pequeños que permitan al diseñador tomar decisiones fundamentadas y cuantificar el progreso. A continuación, se enumeran los objetivos secundarios que permitirán alcanzar el objetivo principal:

- Estudio y comprobación del correcto funcionamiento de los componentes de la placa esenciales para este trabajo: micrófono y salida de audio.
- Estudio, implementación y validación por separado de cada uno los procesos de adaptación de la señal: muestreo del micrófono, modulación PDM, modulación PCM, diezmado e interpolación. Una vez validados por separado, se implementará y se validará el sistema formado por estos elementos, que ha de ser capaz de muestrear la señal del micrófono, codificarla en PCM, decodificarla de nuevo a PDM y reproducirla.
- Estudio del algoritmo de la FFT y de la arquitectura SDF para su implementación, adaptación a las características del sistema. Además, para minimizar el área se diseñará e implementará la IFFT con arquitectura SDF y entrada en *bit-reversed order*. Cada uno de estos módulos se validará por separado y posteriormente en serie para comprobar su precisión.
- Diseño, implementación y validación del ecualizador con entrada en serie, dividido en un número óptimo de bandas de frecuencia.
- Integración y validación del sistema completo mediante simulaciones y en la placa.

En cuanto a las especificaciones del sistema, se han concretados las siguientes:

- Alcanzar un compromiso adecuado entre *throughput* y área, permitiendo el procesamiento de un dato cada ciclo de reloj con la mínima área y consumo de potencia.
- Utilización de anchos de palabra, frecuencias de muestreo y demás parámetros lo más estandarizados posibles en el ámbito del audio digital.
- Implementación de una interfaz hombre-máquina que permita seleccionar las diferentes bandas de frecuencia a ecualizar de una forma sencilla e intuitiva para el usuario.

1.3.ESTRUCTURA DE LA MEMORIA

La estructura de esta memoria está dividida en 6 capítulos, incluyendo el presente, correspondiente a la introducción y a los objetivos del trabajo. A continuación, se muestra un breve resumen del contenido de cada capítulo:

- Capítulo 2 (Estado del arte): se explora el estado del arte de las placas de desarrollo basadas en FPGAs, así como de las herramientas de síntesis de circuitos digitales. Además, se han analizado y explicado las diferentes técnicas de procesamiento y de modulación digital que han sido usadas en este proyecto.

- Capítulo 3 (Materiales y métodos): se explica el proceso de desarrollo del proyecto, centrándose en las principales decisiones de diseño tomadas, con su correspondiente razonamiento, y el funcionamiento de cada módulo.
- Capítulo 4 (Experimentos y resultados): se detallan los experimentos realizados para validar el sistema, tomar las distintas decisiones de diseño y analizar su efecto sobre las prestaciones del circuito. Además, se han discutido y valorado los resultados obtenidos.
- Capítulo 5 (Conclusiones y líneas futuras): se extraen las principales conclusiones a partir de los resultados del capítulo anterior en relación con los objetivos y especificaciones fijados al inicio. Asimismo, se proponen futuras líneas de trabajo y mejoras para posibles continuaciones del proyecto.
- Capítulo 6 (Referencias y bibliografía): se enumeran las referencias y bibliografía empleadas para la investigación y desarrollo del proyecto.

2. ESTADO DEL ARTE

A lo largo de este capítulo, se explicarán los fundamentos teóricos de los diferentes módulos del circuito, así como el hardware sobre el que se han implementado y las herramientas de síntesis empleadas. El capítulo está dividido en tres partes principalmente: la sección 2.1, *FPGAS y placas de desarrollo*, donde se introduce el concepto de FPGA y su relevancia en el campo del desarrollo de circuitos digitales; la sección 2.2, *Herramientas de síntesis*, donde se explican brevemente las herramientas y lenguajes utilizados para sintetizar el circuito; y la sección 2.3, *Algoritmos de procesamiento digital utilizados*, donde se explican los fundamentos de las técnicas de procesamiento y modulación utilizadas en el diseño del sistema.

Para facilitar la comprensión durante la lectura y tener una visión global del circuito y las partes que lo componen se recomienda visitar la Figura 1.1.

2.1.FPGAS Y PLACAS DE DESARROLLO

FPGA son las siglas de *Field Programmable Gate Arrays*, es decir, matrices de puertas lógicas programables en campo. Son dispositivos programables que pueden implementar cualquier circuito electrónico digital mediante la interconexión de bloques lógicos configurables. Desde su primera aparición comercial en 1985 [4], su relevancia en el mercado internacional de semiconductores no ha hecho más que crecer, alcanzado un valor de mercado de 8.74 billones de dólares en 2024 [5].

Las FPGAs suponen un enorme ahorro de tiempo y costes de desarrollo y manufacturación frente a los circuitos integrados para aplicaciones específicas (ASIC), ya que estos últimos suelen ser *full-custom* o, en su defecto, *semi-custom*. Este tipo de circuitos completamente adaptados a la aplicación están altamente optimizados en área y eficiencia energética, pero requieren un tiempo de desarrollo y unos costes de manufacturación muy altos. Este problema es resuelto por las FPGAs, que permiten una implementación instantánea con unos costes muy bajos, aportando mucha más flexibilidad y limitando los riesgos financieros [6].

Además, las FPGAs presentan ciertas ventajas en las prestaciones respecto a los procesadores. Por lo general, las FPGAs son más usadas a la hora de implementar algoritmos complejos debido a su capacidad de paralelización de operaciones y su gran flexibilidad para adaptarse a estándares de diseño cambiantes [7]. No obstante, los procesadores de señales digitales también tienen ventajas, como un *time to market* más rápido, menos coste y una implementación muy eficiente de operaciones comunes en el procesado de señales como las aritméticas en punto flotante [7].

Es por estos motivos, especialmente por la capacidad de paralelización y optimización de algoritmos, que este proyecto ha sido implementado en una FPGA. No obstante, el empleo de una FPGA requiere de una serie de circuitos y periféricos que permitan su conexión a un ordenador para cargar el circuito deseado. Este conjunto de circuitos adicionales puede alcanzar una gran complejidad y depende de factores externos al circuito digital, como el lugar en el que se posicionará la FPGA, los componentes con los que interactuará, las interfaces que usará, etc. Por estos motivos, para el proceso de diseño del circuito digital, se suelen utilizar placas de desarrollo basadas en FPGAs, que contienen el hardware externo necesario para que el diseñador pueda probar las distintas versiones del circuito de forma casi inmediata, facilitando el acceso a todas las funcionalidades de la FPGA y abstrayéndolo en gran medida del diseño hardware externo a ella.

2.2.HERRAMIENTAS DE SÍNTESIS

Para la implementación de un circuito digital en una FPGA, es necesario realizar una descripción del circuito mediante lenguajes de descripción de hardware como VHDL o Verilog. Estos lenguajes permiten especificar, a alto nivel, el comportamiento y la estructura del circuito diseñado. Una vez que se ha realizado esta descripción, es necesario emplear un software especializado en síntesis y simulación de circuitos digitales capaz de convertir la descripción del circuito en una representación a nivel de puertas lógicas que puede ser implementada en la FPGA. Durante el proceso de síntesis, se llevan a cabo optimizaciones en términos de área y consumo de energía, se asignan las señales y operaciones a los diferentes bloques de la FPGA, se realiza un análisis temporal, y se genera un archivo *netlist*. Este archivo describe la configuración y las interconexiones entre los bloques lógicos que componen el circuito, sin especificar aún la disposición física de cada bloque dentro de la FPGA. A este archivo se le conoce como *unrouted netlist*.

Tras la síntesis, se lleva a cabo el proceso de implementación, donde se determina la disposición exacta de los recursos utilizados para conectarlos mediante las rutas más cortas posibles. Dado que estas rutas introducen retardos, se vuelve a realizar un análisis temporal para comprobar que el camino crítico cumple con las especificaciones del diseño. En esta fase, se genera un nuevo *netlist*, denominado *routed netlist*, que especifica la disposición precisa de los recursos y las conexiones dentro de la FPGA.

Finalmente, a partir del *routed netlist* se genera un archivo de configuración binario que contiene toda la información del circuito. Esta información se carga en FPGA mediante el circuito de programación de la placa, que en el caso de la Nexys A7 está basado en una interfaz USB-JTAG. Al cargar el archivo, la FPGA configura su hardware para implementar el circuito digital especificado.

Puesto que la placa de desarrollo elegida es de Xilinx, se ha optado por utilizar su entorno de desarrollo, Vivado. Este software, especialmente adaptado para dispositivos de la serie Artix-7, ofrece herramientas avanzadas para el diseño, síntesis, implementación, simulación y análisis de circuitos digitales.

En cuanto al lenguaje de descripción hardware, en este proyecto se ha optado por VHDL. Aunque tanto VHDL como Verilog son ampliamente utilizados en la industria para el diseño de circuitos digitales, VHDL es especialmente popular en Europa. Una de las principales ventajas de VHDL radica en su enfoque en la descripción detallada del hardware, lo que facilita la legibilidad y estructura del código. A diferencia de Verilog, que tiende a centrarse más en el modelado y puede parecer más conciso, VHDL permite una mayor claridad al especificar el comportamiento y la arquitectura de los circuitos. Esta claridad es particularmente útil en proyectos complejos, donde la precisión y la organización del código son esenciales para garantizar un diseño eficiente y libre de errores.

Además del software de síntesis y simulación de circuitos digitales, se ha hecho uso del software de cómputo numérico MATLAB para complementar y analizar las simulaciones de Vivado. Esto es posible puesto que el simulador de Vivado permite la entrada de datos desde archivos de texto y puede registrar los valores de salida en este mismo formato. Con MATLAB, es posible generar señales sintéticas que sirven como datos de entrada para las simulaciones y, posteriormente, realizar un análisis exhaustivo de las señales de salida. Esto último es especialmente útil para la validación del procesamiento de señales de audio, ya que MATLAB permite transformar las señales desde el dominio del tiempo al dominio de la frecuencia, facilitando el examen detallado de los cambios en el espectro de la señal, lo que es crucial para verificar la efectividad de las operaciones de ecualización y otros procesos de manipulación del audio.

2.3.ALGORITMOS DE PROCESAMIENTO DIGITAL UTILIZADOS

A continuación, se explican los fundamentos de las distintas técnicas de procesamiento y modulación necesarias para comprender el diseño y funcionamiento del sistema.

2.3.1.FFT/IFFT

La transformada rápida de Fourier (FFT) comprende el conjunto de algoritmos que permiten calcular eficientemente la transformada de Fourier discreta (DFT), capaz de transformar una señal discreta en el dominio del tiempo al dominio de la frecuencia. Este algoritmo cuenta con numerosas aplicaciones prácticas, como el procesamiento de audio e imagen y el análisis de señales. La DFT de N puntos de una secuencia de entrada $x[n]$ se define como:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{N-1} x[n]\, W_N^{nk}, \qquad k = 0, 1, \dots, N-1 \qquad (1)$$

Donde $X[k]$ es la secuencia de salida en el dominio de la frecuencia y W_N^{nk} son los llamados *Twiddle factors*, aunque en este trabajo también nos referiremos a ellos como rotaciones, ya que cada uno de ellos lleva a cabo una rotación que vendrá dada por el producto $nk = \phi$. Nótese que si ϕ toma los valores $0, N/4, N/2$ o $3N/4$ la rotación será de 0º, 270º, 180º y 90º respectivamente, que corresponden a multiplicar por $1, -j, -1$ y j. Estas reciben el nombre de rotaciones triviales debido a su sencillez de cálculo [8].

El algoritmo más común para implementar la FFT es el propuesto por Cooley y Tukey, capaz de reducir el número de operaciones de la DFT de N^2 a $Nlog_2N$ [9]. Para adquirir una visión general de las operaciones que lo componen, se suelen usar diferentes representaciones gráficas, siendo el gráfico de flujo la más común. Estos gráficos están compuestos por varias etapas en las que se representan las operaciones de suma, resta y multiplicaciones complejas mediante "mariposas" (Figura 2.1). Además, el gráfico incluye un índice en binario para cada entrada, que indica los pares de datos que se procesan conjuntamente. Para una FFT de N puntos y $n = log_2N$ etapas, siempre se procesarán, juntos y en las mariposas de la etapa s, los datos que difieren en el bit b_{n-s} del índice [8]. Todo esto se puede apreciar en la Figura 2.2.

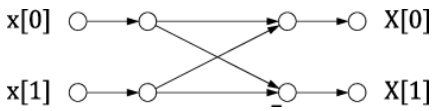

Figura 2.1. Mariposa de radix-2 [8].

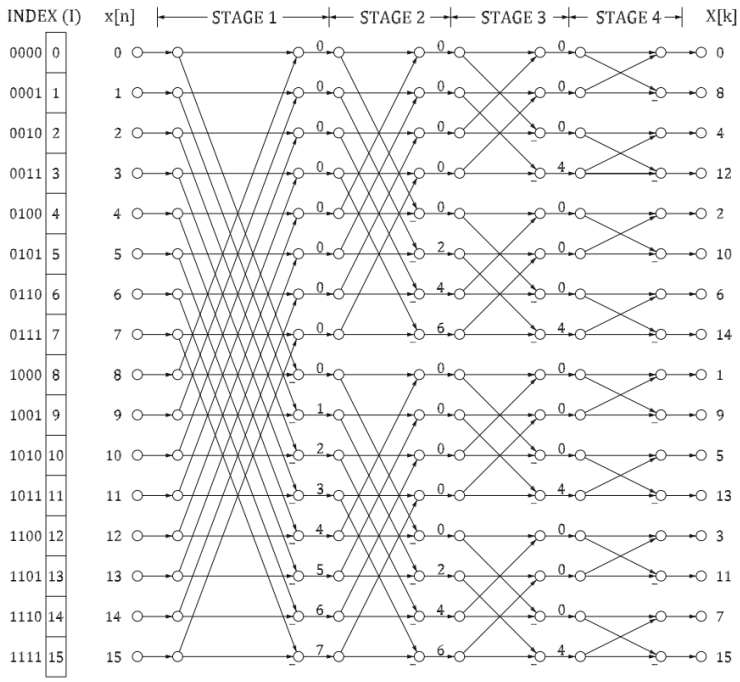

Figura 2.2. Gráfico de flujo de una FFT DIF de radix-2 de 16 puntos [8].

Aparte de lo antes mencionado, en la Figura 2.2 aparecen números entre cada etapa que representan la rotación ϕ de cada mariposa. En este documento, las rotaciones de un estado s corresponden a aquellas situadas a la derecha de sus mariposas.

Además, se observa que el orden de la secuencia de entradas $x[n]$ es diferente al de salida $X[k]$. Si se representan los índices de $X[k]$ en binario se comprueba que son los mismos bits de los índices de $x[n]$ pero en orden inverso. A esto se le conoce como *bit-reversed order* (BR).

Así como la FFT implementa la DFT de forma eficiente, la inversa de la transformada rápida de Fourier (IFFT) implementa la transformada de Fourier discreta inversa (IDFT), capaz de transformar una señal discreta en el dominio de la frecuencia al dominio del tiempo. La IDFT de N puntos de una secuencia de entrada $X[k]$ se define como:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi}{N}nk} = \frac{1}{N}\sum_{n=0}^{N-1} X[k]\,W_N^{nk}, \qquad k = 0, 1, \dots, N-1 \qquad (2)$$

La expresión es muy similar a la de la DFT, pero las rotaciones tienen sentido contrario y el resultado del sumatorio es dividido entre N. Por tanto, el algoritmo para implementar la IFFT es el mismo que el de la FFT, pero con esos ligeros cambios.

Cabe mencionar también que la FFT e IFFT tienen efectos sobre la amplitud de la señal. Al aplicar la FFT se multiplica la amplitud por N mientras que al aplicar la IFFT se divide entre N. Esto es muy conveniente al concatenar la transformada con su inversa, ya que la amplitud de la señal de entrada se mantiene a la salida (Figura 2.3).

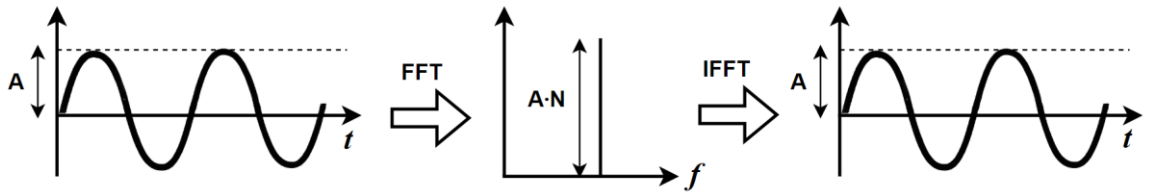

Figura 2.3. Efecto sobre la amplitud de la señal al concatenar FFT e IFFT.

Existe una gran variedad de algoritmos y arquitecturas segmentadas con diferentes características para implementar la FFT. Para este trabajo, se nos ha proporcionado una FFT de 1024 puntos basada en el algoritmo Cooley-Tukey diezmado en frecuencia con mariposas radix-2, este algoritmo es uno de los más utilizados, debido a su alta eficiencia y su facilidad de implementación [3]. El diezmado en frecuencia (DIF) implica que la expresión de la DFT (1) se divide en la suma de dos DFTs: una que computa la primera mitad de las muestras y otra la segunda mitad. Esto da como resultado los coeficientes de la Figura 2.2 para una DFT de 16 puntos, donde se observa cómo, en cada etapa, se divide entre dos el número de FFTs de la etapa anterior, es decir, en cada etapa se divide entre dos el número de coeficientes de la anterior.

En cuanto a la arquitectura de la FFT, se ha utilizado *Single-path Delay Feedback* (SDF). Se trata de una arquitectura serial segmentada capaz de procesar un dato por cada ciclo de reloj. Tal y como se ha mencionado anteriormente, las mariposas de cada etapa s procesan juntos los datos que difieren en el bit b_{n-s} del índice, como estos datos entran en serie, tienen una diferencia de 2^{n-s} ciclos de reloj. Para que estos datos lleguen simultáneamente a la entrada de la mariposa, se usa un buffer de tamaño $L = 2^{n-s}$ [8]. Del mismo modo, la mariposa genera dos resultados: uno de ellos continua hasta el rotador, mientras que el otro se almacena en el buffer para luego llegar hasta el rotador. En la práctica, además se requieren multiplexores y una señal externa para controlar la entrada del buffer, esto se explicará en detalle en la sección 3.7, *FFT e IFFT*. Normalmente, esta arquitectura se representa mediante diagramas de bloques como el de la Figura 2.4.

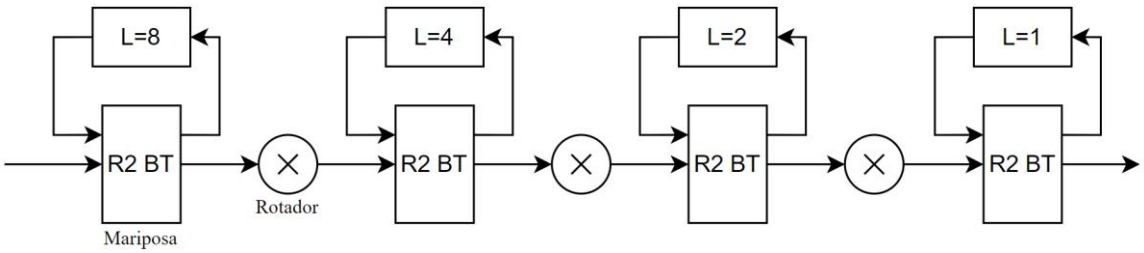

Figura 2.4. Representación de una arquitectura SDF para una FFT de 16 puntos.

2.3.2.ECUALIZACIÓN

Como se ha mencionado con anterioridad, la ecualización consiste en alterar la amplitud de las distintas frecuencias que componen el espectro de la señal de audio. De esta manera, se aumenta o disminuye el volumen de las frecuencias deseadas, manipulando el timbre de los sonidos. Hay varias opciones para implementar un ecualizador digital y alterar la respuesta en frecuencia de la señal, pero se ha optado por transformar la señal y ecualizarla en el dominio de la frecuencia y volverla a transformar al dominio del tiempo. Para este tipo de ecualización solo es necesario multiplicar los valores asociados a cada frecuencia por distintos coeficientes para alterar su amplitud.

Esto conlleva dos ventajas principales respecto a la ecualización en el dominio del tiempo que se podría realizar con filtros digitales: mayor sencillez, puesto que solo se requiere un multiplicador, y más precisión, ya que se puede alterar cada componente frecuencial de la señal por separado, haciendo posible el uso de una amplia variedad de curvas de respuesta en frecuencia.

En la actualidad, los ecualizadores suelen implementarse en software, como parte de las estaciones de trabajo de audio digital y hacen uso de procesadores de señales digitales (DSP) para ejecutar las multiplicaciones por los coeficientes, pero en este trabajo, se ha implementado mediante un circuito digital adaptado a las limitaciones de la placa utilizada.

2.3.3.MODULACIONES

La modulación es un proceso fundamental en la transmisión y procesamiento de señales, que consiste en modificar una señal portadora para que pueda transportar la información deseada. Dependiendo de las características de la señal y del sistema, se utilizan diferentes tipos de modulaciones.

A lo largo de este trabajo se han usado diferentes modulaciones para representar la señal de audio en cada una de las etapas del circuito. Al tratarse de un sistema digital, solo se pueden emplear modulaciones cuya señal portadora es digital y discreta en el tiempo. En la Figura 2.5 se muestra una clasificación de los distintos tipos de modulaciones con portadoras discretas en el tiempo.

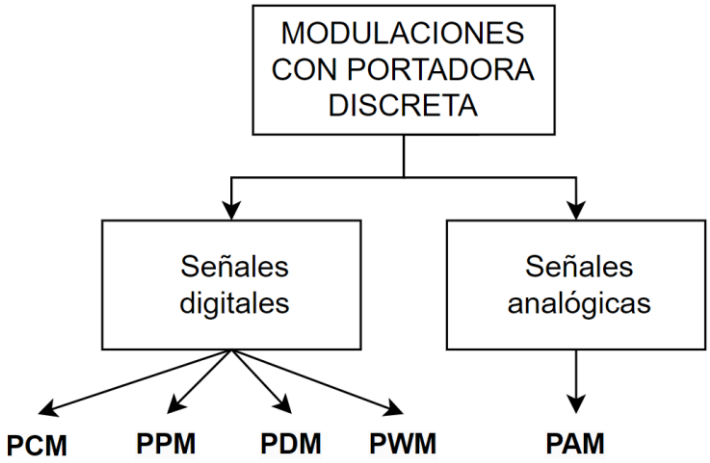

Figura 2.5. Tipos de modulaciones con portadora discreta.

Las modulaciones PCM, PPM, PDM y PWM toman valores discretos mientras que PAM es un tipo de modulación discreta en el tiempo, pero con valores analógicos. A continuación, se explican brevemente estas modulaciones:

- PCM (*Pulse Code Modulation)*: representa una señal analógica muestreada periódicamente mediante un valor discreto de N bits.
- PPM *(Pulse Position Modulation)*: representa una señal analógica mediante una señal de digital de 1 bit cuyos pulsos varían su posición respecto al inicio del periodo de reloj según la amplitud de la señal analógica. Estos pulsos tienen una anchura fija.
- PDM *(Pulse Density Modulation)*: representa una señal analógica mediante una señal digital de 1 bit compuesta por pulsos de una anchura fija que se suceden con distinta frecuencia. La cantidad de pulsos sobre un intervalo de tiempo es proporcional a la amplitud de la señal analógica.
- PWM *(Pulse Width Modulation)*: representa una señal analógica mediante una señal digital de 1 bit compuesta por pulsos de anchura variable, la cual es proporcional a la amplitud de la señal analógica.
- PAM *(Pulse Amplitude Modulation)*: representa una señal analógica mediante una serie de pulsos cuya amplitud es proporcional a la amplitud de la señal analógica.

En este proyecto solo se ha utilizado modulación PDM y PCM, por lo que se han explicado con mayor profundidad en las siguientes secciones.

2.3.3.1. MODULACIÓN PDM

Pulse Density Modulation (PDM) es un tipo de modulación que permite representar una señal analógica mediante una señal digital de 1 bit compuesta por pulsos de una anchura fija que se suceden con distinta frecuencia. Se denominan pulsos positivos a aquellos cuyo valor es '1' y negativos a aquellos de valor '0'. El número de pulsos positivos sobre un intervalo de tiempo, es decir, la densidad de pulsos, es proporcional al valor de la señal analógica que representa. Esto significa que una señal compuesta solo por '1's corresponde al máximo valor analógico mientras que una señal compuesta solo por '0's corresponde al mínimo valor analógico. Alternando pulsos positivos y negativos se obtienen valores analógicos intermedios. Esta relación entre densidad de pulsos y amplitud se observa con más claridad en la Figura 2.6, donde se modula en PDM una señal analógica senoidal. En ella se observa que la densidad de pulsos positivos es mayor cuando la amplitud es máxima y menor cuando es mínima.

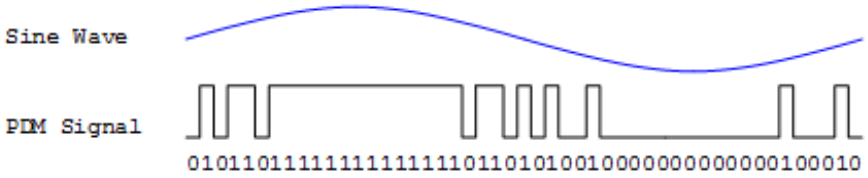

Figura 2.6. Correspondencia entre una señal senoidal y una señal PDM [10].

La modulación PDM de una señal analógica mostrada en la Figura 2.6 se consigue mediante moduladores Sigma-Delta, como el que está incluido en el micrófono de la placa (explicados posteriormente en la sección 2.3.3.3, *Modulador* Sigma-Delta). Sin embargo, para entender más fácilmente el funcionamiento de este tipo de moduladores es necesario comprender antes las distintas formas de modular el audio y su relación con la modulación PDM.

2.3.3.2. MODULACIÓN PCM

La forma más común de modular el audio es mediante *Pulse Code Modulation* (PCM), que representa una señal analógica muestreada periódicamente (normalmente a la frecuencia de Nyquist o ligeramente superior) mediante una señal digital de N bits. En la Figura 2.7 se muestra un ejemplo de modulación PCM de 4 bits (16 posibles valores) de una señal analógica senoidal.

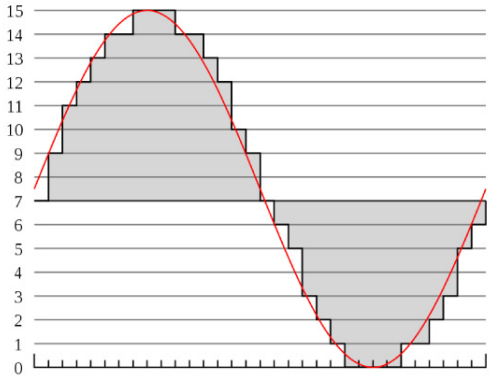

Figura 2.7. Señal analógica(rojo) y su correspondiente modulación PCM de 4 bits [11].

En un convertidor analógico digital (ADC) ideal con una frecuencia de muestreo igual a la de salida, se suele usar el *Signal to Noise Ratio* (SNR) para medir la precisión del convertidor. Para una señal senoidal, se cumple [12]:

$$SNR = 6.02N + 1.76 \quad (3)$$

Donde N es el número de bits con el que se modula la señal PCM.

El resultado para una señal de 16 bits, por ejemplo, es que el SNR es de aproximadamente 98 dB, por lo que la precisión de la conversión es elevada y apta para cualquier aplicación de audio.

2.3.3.3. MODULADOR SIGMA-DELTA

Un modulador Sigma-Delta es un circuito capaz de transformar una señal analógica o digital representada en PCM a una señal PDM. Para comprender su funcionamiento es útil partir de la ecuación (3), que ha permitido demostrar en la sección anterior que la precisión de conversión de un ADC para una señal PCM de 16 bits es elevada. Sin embargo, si en vez de 16 bits, se usa 1 bit para modular la señal, como es el caso del PDM, el SNR es de 7.78 dB, una relación demasiado baja para cualquier tipo de aplicación de audio. La forma de mejorar esta relación en el caso del PDM es emplear la técnica de modelado de ruido combinada con el sobremuestreo de la señal (muestrear a una frecuencia significativamente mayor a la de Nyquist).

En el caso de señales de audio, el modelado de ruido consiste en alterar el espectro de ruido de la señal para reducir su potencia en las frecuencias audibles y aumentarla en las no audibles. Sin embargo, si muestreamos a la frecuencia de Nyquist o ligeramente superior, no basta con modelar el ruido, ya que este será audible en la mayor parte de las frecuencias del ancho de banda del sistema, por este motivo se lleva a cabo un sobremuestreo. Si se sobremuestrea la señal, el ancho de banda del sistema será mucho más amplio y se dispondrá de un gran rango de frecuencias no audibles a las que se puede desplazar el ruido. Esto es precisamente lo que consigue un convertidor Sigma-Delta, cuyo radio de sobremuestreo (OSR) se define según la expresión:

$$OSR = \frac{f_s}{2f_B} \qquad (4)$$

Donde f_s es la frecuencia de muestreo y f_B es el ancho de banda de la señal.

El diagrama de un convertidor Sigma-Delta de primer orden usado como ADC se muestra en la Figura 2.8, donde u es la entrada, analógica o modulada en PCM, y v es la salida, modulada en PDM.

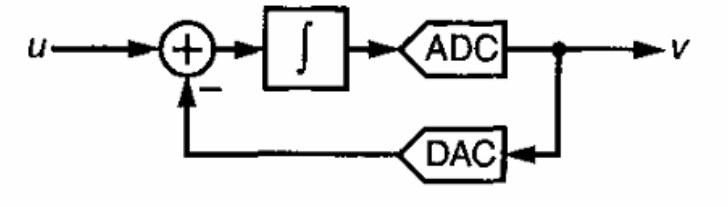

Figura 2.8. Modulador Sigma-Delta de primer orden **[12]**.

El modulador consiste en un lazo de realimentación compuesto por un filtro paso bajo (en este caso, un integrador), un ADC de 1 bit, que inevitablemente introduce ruido de cuantificación al truncar la salida del integrador, y un DAC, que no introduce ruido de cuantificación puesto que, al tener una entrada de 1 bit, su salida solo puede ser el valor máximo o mínimo de la entrada u. El ruido producido por el ADC es no lineal, pero para facilitar el análisis cualitativo se puede usar un modelo lineal en el que el ruido, representado por e, es acumulativo. Ya que la señal de entrada es muestreada periódicamente, se ha representado el mismo circuito en el dominio de tiempo discreto mediante la transformada Z, mostrado en la siguiente figura [12]:

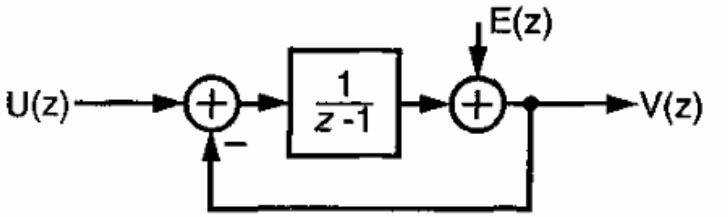

Figura 2.9. Modulador Sigma-Delta de primer orden en el dominio de tiempo discreto [12].

En Figura 2.9, se ha reemplazado el integrador por su función de transferencia y el ADC por su modelo lineal, mientras que el DAC no se representa ya que no introduce ningún ruido de cuantificación. Si se analiza el circuito se extrae la siguiente expresión:

$$V(z) = U(z)z^{-1} + E(z)(1 - z^{-1}) \qquad (5)$$

Se observa que la salida digital está compuesta por la señal de entrada retrasada un ciclo de reloj y el error de cuantificación diferenciado. La componente $(1 - z^{-1})$ recibe el nombre de función de transferencia de ruido (NFT). Si se transforma la función NFT en el dominio de la frecuencia sustituyendo $z = e^{-j2\pi f}$ y se eleva su módulo al cuadrado para obtener la densidad espectral se obtiene la siguiente expresión, donde f corresponde a la frecuencia normalizada f/f_s:

$$\left|NFT\left(e^{-j2\pi f}\right)\right|^2 = 4 \cdot |\sin(\pi f)|^2 \quad (6)$$

Si se representa la ecuación (6) se obtiene la gráfica de la Figura 2.10:

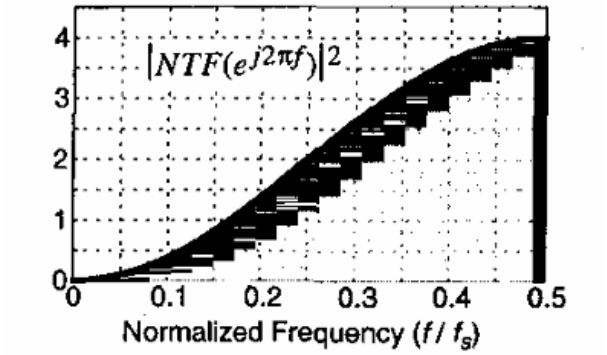

Figura 2.10. Función de transferencia de ruido en función de la frecuencia normalizada [12].

Se observa que el ruido se atenúa en las frecuencias cercanas al 0, mientras que se amplifican las frecuencias alrededor de $f_s/2$, es decir, estamos ante un filtro paso alto que modela el ruido desplazándolo a las altas frecuencias. Al ser un filtro de primer orden, la pendiente no es muy elevada, por lo que seguiría habiendo bastante ruido en las frecuencias audibles. La solución más simple es utilizar filtros de mayor orden, cuanto mayor sean, mayor será el SNR en las frecuencias de interés. La forma más común de implementar filtros de mayor orden en estos moduladores es incluir más integradores y lazos de realimentación. Se define el orden de un modulador Sigma-Delta como el orden del filtro paso alto implementado por la NFT.

Tal y como se ha explicado, PDM es un tipo de modulación que tan solo requiere un bit, pero a cambio, utiliza una frecuencia de muestreo mucho mayor a la de Nyquist, por lo general entre 50 y 70 veces mayor. Esto proporciona algunas ventajas: principalmente simplicidad de transmisión a través del circuito y facilidad de conversión a señal analógica, ya que solo se requiere un filtro paso bajo. Sin embargo, el procesamiento de este tipo de señales es más complejo, ya que la información no está contenida en los bits en sí, sino en la frecuencia con la que se suceden los pulsos. Por este motivo, se usa PCM de varios bits para tratar la señal, ya que es mucho más intuitiva y permite una manipulación más sencilla de los datos.

Como caso práctico de diseño se ha estudiado la modulación una señal analógica con un espectro de frecuencias comprendido entre los 0 y 20 kHz a PDM. En este caso, la frecuencia de Nyquist es 40 kHz. Si la frecuencia de sobremuestreo ha de ser entre 50 y 70 veces superior, entonces la señal analógica muestreará a una frecuencia entre 2 y 2.8 MHz. Como demuestra la Figura 2.10, cuanto mayor sea la frecuencia de sobremuestreo, menor será la densidad de ruido en las frecuencias audibles.

2.3.4.DIEZMADO E INTERPOLACIÓN

Las técnicas de diezmado e interpolación disminuyen o aumentan respectivamente la frecuencia de muestreo de una señal y, combinadas con un filtro paso bajo, permiten la conversión de PDM a PCM.

2.3.4.1. FILTRO DIEZMADOR

En este proyecto se ha llevado a cabo un procesamiento de datos basado en FFTs, cuya entrada ha de estar modulada en PCM, por lo que es imprescindible codificar la señal PDM generada por el micrófono a PCM. Para llevar a cabo esta transformación se usa la técnica de diezmado. "El proceso de diezmado consiste en reducir la frecuencia de muestreo en un factor de diezmado M" [13], para ello se selecciona una de cada M muestras y se descartan las M-1 muestras intermedias. Esto se expresa siguiente ecuación:

$$y[n] = x[nM] \quad (7)$$

Si se aplica la transformada Z sobre la salida del diezmador se obtiene la expresión (8) [13], la cual no se puede relacionar directamente con la señal de entrada puesto que algunas muestras se eliminan.

$$Y(z) = \sum_{n=-\infty}^{\infty} y[n]z^{-n} = \sum_{n=-\infty}^{\infty} x[Mn]z^{-n} \qquad (8)$$

Para poder relacionar la salida con la entrada en el dominio de tiempo discreto se hace uso de una señal auxiliar con el mismo valor que la entrada en los múltiplos de M y valor nulo en el resto de muestras. Esta señal se obtiene multiplicando la entrada por un tren de impulsos con periodo M. Si se le aplica la transformada Z y se simplifica la expresión, el resultado es [13]:

$$Y(z) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(z^{\frac{1}{M}} \cdot e^{\frac{-j2\pi k}{M}}\right) \qquad (9)$$

La ecuación (9) muestra que la señal diezmada en el dominio Z es la suma de versiones escaladas y desplazadas del espectro original. Si se representa una señal diezmada en el dominio de la frecuencia se observa que su espectro tiene la misma forma que el de la original, pero con la amplitud dividida entre M y extendido en frecuencia un factor M. No obstante, el ancho de banda de la señal se mantiene constante, la extensión del espectro se produce solo en relación con la frecuencia de muestreo, que disminuye con el diezmado. Esto se observa mejor en la Figura 2.11.

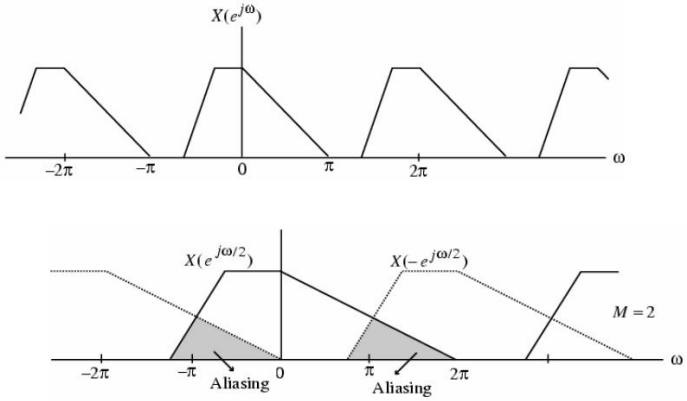

Figura 2.11. Espectro de la señal original (arriba) y la señal diezmada (abajo) con M=2 [14].

En la Figura 2.11, se observa además que la señal diezmada presenta aliasing. Para evitarlo es necesario atenuar las frecuencias de la señal original que estén por encima de $f_s/2M$, lo cual se puede lograr aplicando un filtro paso bajo antes del diezmador. No obstante, la concatenación de un filtro paso bajo con un diezmador es bastante ineficiente, ya que la mayoría de las muestras filtradas serán descartadas. Por esta razón utilizamos los llamados filtros diezmadores, que implementan ambas funciones de forma más eficiente.

El filtro diezmador se ha implementado haciendo uso de los cores IP disponibles en Vivado. El término IP Core hace referencia a bloques lógicos reutilizables que son propiedad intelectual de una de las partes. Son comúnmente utilizados en FPGAs ya que reducen el *time to market* del producto y la brecha entre la tecnología disponible y los conocimientos de los desarrolladores. En este trabajo se han usado para la codificación y decodificación de señal, ya que el diseño de filtros diezmadores no es el principal objetivo en el que se centra este trabajo. Además, nos proporcionarán un nivel de optimización tanto en área como en velocidad que difícilmente se podrían alcanzar con un diseño propio dadas las limitaciones de tiempo del proyecto.

En Vivado, disponemos de dos tipos de filtros digitales que pueden realizar la función de diezmado: los FIR y los CIC. Es preciso elegir aquel que se adapte mejor a las características de nuestro proyecto para obtener una buena relación entre área y precisión. Para ello, es necesario comprender el funcionamiento de ambos, así como sus ventajas y desventajas:

- *Finite Input Response* (FIR): son filtros no recursivos en los que la salida se obtiene a partir de la suma de la entrada actual y previas multiplicadas por una serie de coeficientes. Al no usar realimentación, su respuesta a impulso es finita, ya que se fija a 0 tras un tiempo finito. Su respuesta viene dada por la ecuación (10) [15], donde y corresponde a la salida del filtro, x a la entrada, a son los coeficientes y N es el orden.

$$y[k] = \sum_{n=0}^{N-1} a[n]x[k-n] \qquad k = 0, 1, \dots \qquad (10)$$

  La Figura 2.12 muestra una implementación hardware simplificada del filtro. Su principal desventaja es la gran cantidad de operaciones aritméticas que requiere, las cuales incrementan el área y el consumo de potencia y limitan la velocidad. Esto es especialmente problemático para los multiplicadores, que incrementan en gran medida la complejidad del circuito. Pese a se han desarrollado métodos para aumentar las prestaciones, como la limitación de coeficientes a potencias de 2, sustituyendo los multiplicadores por desplazadores, o el uso de bloques paralelos, la complejidad sigue siendo elevada, especialmente si el orden del filtro es alto [16].

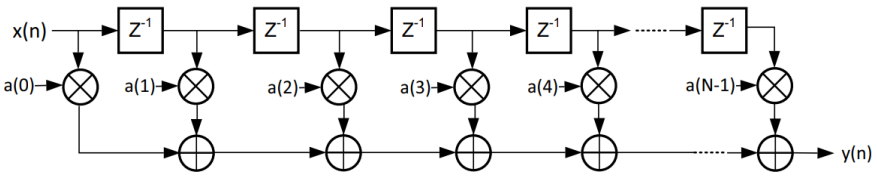

Figura 2.12. Estructura de un filtro FIR [17].

En el compilador FIR de Vivado, para obtener filtros FIR diezmadores se utilizan filtros polifásicos. En este tipo de implementación, el filtro original se divide en M subfiltros polifásicos de N/M coeficientes, tal que:

$$E_i(r) = a(i + Mr) \qquad i = 0, 1, \dots, M-1 \qquad r = 0, 1, \dots, \frac{N}{M} \qquad (11)$$

A cada subfiltro le llega una muestra a $1/M$ de la frecuencia original mediante un conmutador o registros. De esta forma, solo se realizan las operaciones realmente necesarias para el obtener una salida diezmada, reduciendo el número de multiplicaciones de N a N/M. A continuación, se muestra la estructura genérica de un filtro diezmador polifásico, donde los subfiltros son representados por $E_i(z)$.

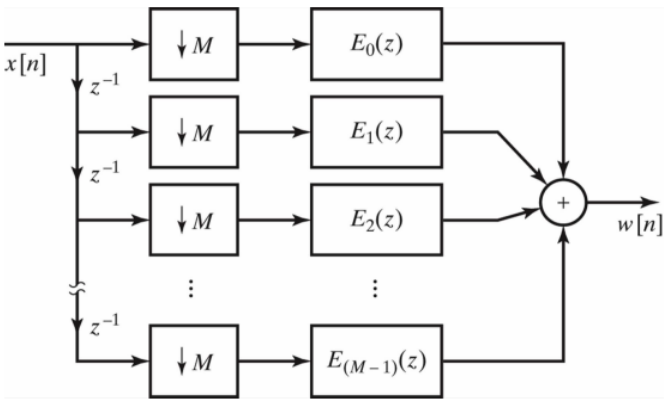

Figura 2.13. Diagrama de un filtro FIR polifásico diezmador [18].

- *Cascaded Integrator-Comb* (CIC): son un tipo de filtros FIR paso bajo computacionalmente eficientes que suelen utilizarse para diezmar o interpolar. Para que sea diezmador, el filtro está compuesto por un integrador en serie con un filtro de peine (*Comb*), ambos de varias etapas. El integrador opera a la frecuencia de sobremuestreo, mientras que el filtro de peine solo recibe una entrada cada M operaciones del integrador, es decir, trabaja a la frecuencia diezmada.

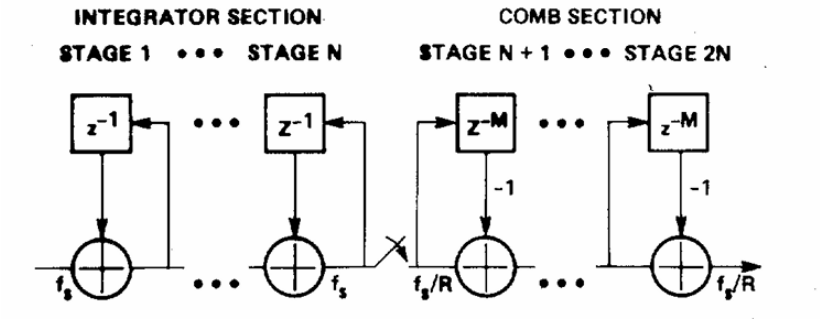

Figura 2.14. Diagrama de un filtro diezmador CIC [19].

En la Figura 2.14 se muestra un filtro CIC diezmador de orden N, en ella, R hace referencia al factor de diezmado, M al delay diferencial del filtro y N al número de etapas de cada sección. Cada sección ha de tener el mismo número de etapas y este número determina el orden del filtro. Analizando el circuito, se obtiene la función de transferencia de la sección integradora del filtro:

$$H_I(z) = \left(\frac{1}{1-z^{-1}}\right)^N \qquad (12)$$

De igual forma, la función de transferencia de la sección de peine tomando como referencia la frecuencia de sobremuestreo es:

$$H_C(z) = (1-z^{-RM})^N \qquad (13)$$

Entonces, la función de transferencia del filtro es [19]:

$$H_{CIC}(z) = H_I(z)\cdot H_C(z) = \left(\frac{1-z^{-RM}}{1-z^{-1}}\right)^N = \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N \qquad (14)$$

Si se compara la ecuación (14) con la función de transferencia de un filtro de media móvil (15), se observa que son muy similares, con la diferencia del factor de diezmado y la división entre M, por lo que se podría interpretar el filtro CIC como una especie de filtro de media móvil que suma los M últimos valores de entrada. No obstante, un filtro de media móvil requiere M-1 sumadores, mientras que un CIC de primer orden solo requiere 2.

$$H_{MovAv}(z) = \frac{1}{M}\cdot\frac{1-z^{-M}}{1-z^{-1}} \qquad (15)$$

La respuesta en frecuencia del CIC se obtiene evaluando la ecuación (16) con $z = e^{j(2\pi f/R)}$, donde f es la frecuencia diezmada. El resultado es [19]:

$$H(f) = \left(\frac{sen(\pi f M)}{sen(\pi f/R)}\right)^{2N} \qquad (16)$$

Si se representa en función de la frecuencia normalizada con N=3, R=52 y M=1, se obtiene la gráfica de la Figura 2.15. En la sección 4.2.1, *Diseño y respuesta de los filtros CIC y FIR*, se

explicará por qué se ha elegido esos valores. Se observa que su respuesta tiene valores nulos en los múltiplos de 1/M y una frecuencia de corte de 0.266, equivalente a 12768 Hz.

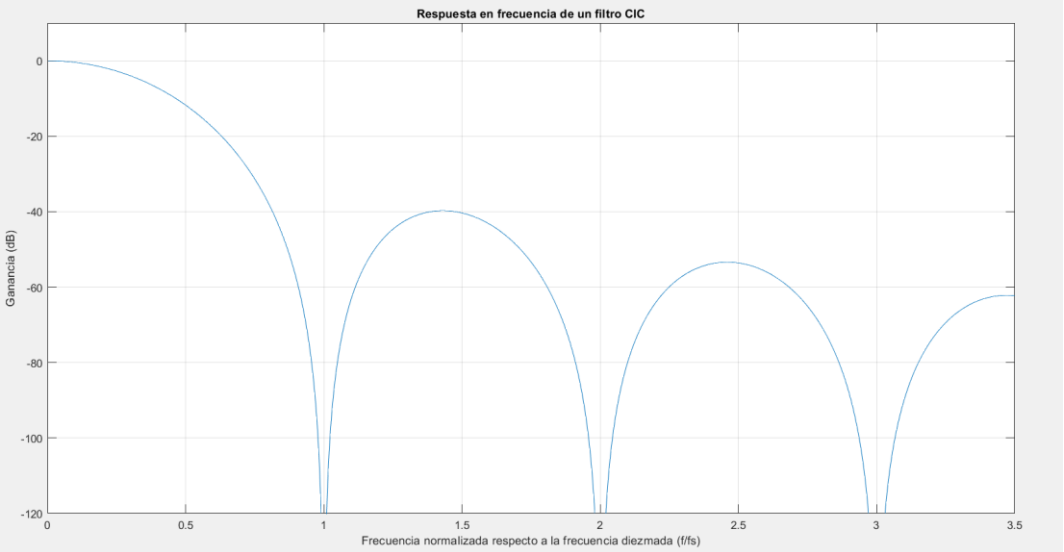

Figura 2.15. Respuesta en frecuencia de un filtro CIC con N=3, R=52 y M=1.

La principal ventaja del filtro CIC frente al FIR es que, al no necesitar multiplicadores, consume menos potencia, requiere menos área y tiene mayor velocidad. Además, no necesita hacer uso de memoria para almacenar coeficientes. Sin embargo, su principal desventaja es la respuesta en frecuencia: es poco flexible, ya que está completamente determinada por los parámetros N, R y M, la atenuación en la banda de rechazo no es tan grande como en los FIR y requieren alcanzar un compromiso entre la banda de paso y de rechazo, puesto que, si se aumenta la atenuación en la banda de rechazo usando más etapas, también se incrementa la atenuación en la banda de paso.

2.3.4.2. FILTRO INTERPOLADOR

La salida de audio de la placa de desarrollo utilizada está diseñada para señales PDM o PWM, de las cuales se ha optado por PDM (está decisión se explicará en la sección 3.4, *Decodificación de PCM y codificación a PDM*), por lo que es necesario convertir la señal PCM ya procesada a PDM. Esto se consigue mediante las técnicas de interpolación y modulación Sigma-Delta digital. Con la interpolación se incrementa la frecuencia de muestreo de la señal, muy similar al diezmado, pero a la inversa. La interpolación consiste en "aumentar la frecuencia de muestreo en un factor de interpolación L" [13], para ello, se insertan L-1 muestras de valor nulo entre las muestras tomadas a la frecuencia original. Esto se expresa en la expresión (17):

$$y[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L \\ 0 & ,en\ el\ resto\ de\ casos \end{cases} \quad (17)$$

Si se le aplica la transformada Z y se simplifica la expresión, el resultado es [13]:

$$Y(z) = X(z^L) \quad (18)$$

A partir de la ecuación (18) se puede evaluar la respuesta en frecuencia con $z = e^{j2\pi f}$ obteniendo:

$$Y\left(e^{j2\pi f}\right) = X\left(e^{j2\pi fL}\right) \quad (19)$$

La ecuación (19) indica que el espectro de la señal interpolada $Y\left(e^{j2\pi f}\right)$ se obtiene al evaluar el espectro de la señal original $X\left(e^{j2\pi fL}\right)$ a una frecuencia fL en lugar de f. Esto significa que cada punto

en el espectro original se mapea a una frecuencia más baja en el espectro interpolado, dando como resultado una compresión en frecuencia por un factor L. Asimismo, al estrechar el espectro, la energía total de la señal debe conservarse. Sin embargo, como el espectro se ha comprimido en un intervalo más pequeño, la densidad espectral de energía (y, por lo tanto, la amplitud del espectro) aumenta por un factor L para mantener la misma energía total. Esto se observa mejor en la Figura 2.16:

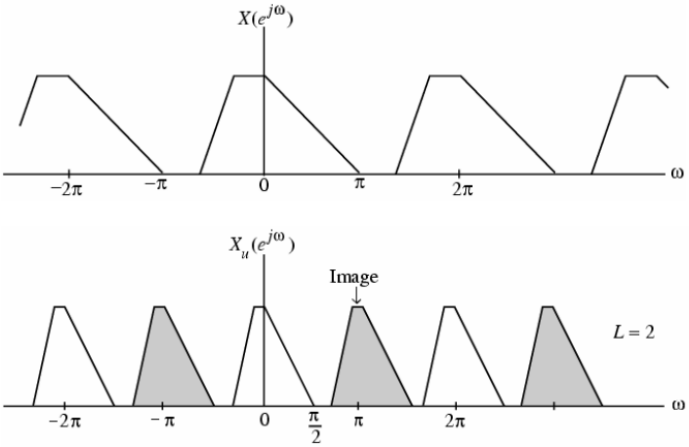

Figura 2.16. Espectro de la señal original (arriba) y la señal interpolada (abajo) con L=2 [14].

Se observa además como aparecen repeticiones del espectro, llamadas imágenes. Para recuperar la señal original basta con usar un filtro paso bajo tras la interpolación que atenúe las frecuencias superiores al ancho de banda al que habíamos limitado la señal original en el diezmado. No obstante, concatenar un interpolador con un filtro no es muy eficiente, ya que L-1 de cada L muestras del filtro valdrán 0. Una implementación más eficiente se consigue mediante filtros interpoladores.

Al igual que con los filtros diezmadores, implementaremos los interpoladores mediante cores IP de Vivado. Las arquitecturas de los filtros interpoladores FIR y CIC son muy similares a las de los diezmadores antes mencionados, así como sus ventajas e inconvenientes.

En los FIR interpoladores, se utilizan filtros polifásicos de modo que la ecuación (11) sigue siendo válida. En este caso, cada muestra llega a cada subfiltro a la frecuencia de diezmado, se introducen los ceros para llegar a la frecuencia de sobremuestreo y llega uno de los valores a la salida mediante un conmutador o registros.

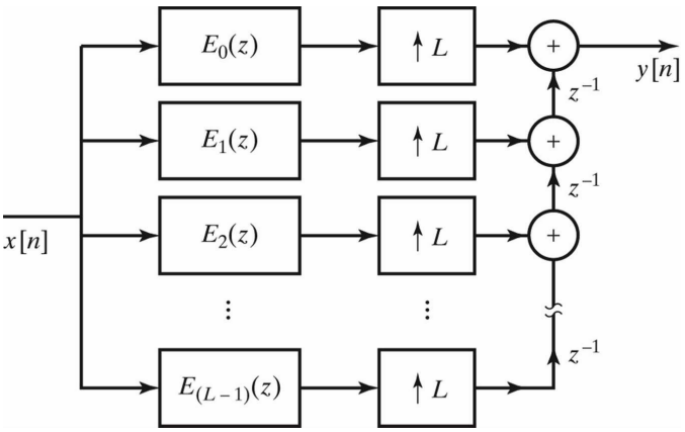

Figura 2.17. Diagrama de un filtro FIR polifásico interpolador [18].

En el caso de los filtros CIC interpoladores, la única diferencia es que el filtro de peine está al principio, seguido de un conmutador que inserta L-1 ceros entre muestras, y finalmente el integrador. Las expresiones y las gráficas del diezmador siguen siendo válidas, ya que el integrador sigue operando a

la frecuencia más alta (la de sobremuestreo) y el filtro de peine a la frecuencia más baja (a la que se diezmó).

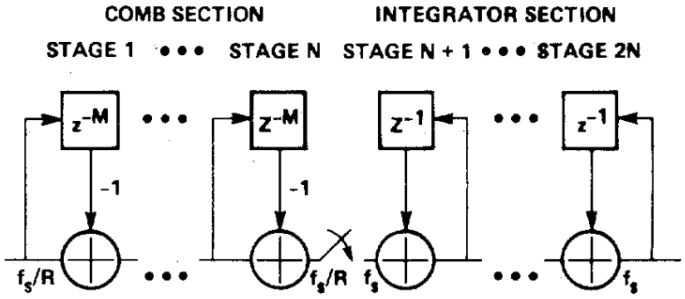

Figura 2.18. Diagrama de un filtro interpolador CIC [19].

Una vez interpolada, la señal es introducida en un modulador Sigma-Delta digital de segundo orden para modularla en PDM.

3. MATERIALES Y MÉTODOS

El capítulo está dividido en varias secciones: la sección 3.1, *Placa de desarrollo*, donde se explica la elección y se analizan la características de la placa utilizada; la sección 3.2, *Muestreo del micrófono*, que analiza las distintas opciones de muestreo del micrófono para obtener una señal PCM; la sección 3.3, *Decodificación de la señal PDM proveniente del micrófono*, donde se especifican las características de la señal PCM que procesa la FFT, así como los requisitos del filtro diezmador; la sección 3.4, *Decodificación de PCM y codificación a PDM*, que explica ; la sección 3.5, *Interpolación*, donde se explican las características principales del filtro interpolador; la sección 3.6, *Modulador Sigma-Delta*, que explica la elección del orden del modulador Sigma-Delta digital y su diseño; la sección 3.7, *FFT e IFFT*, donde se explica la implementación hardware de la FFT y la IFFT; la sección 3.8, *Ecualizador e interfaz hombre-máquina*, que explica el diseño y funcionamiento del ecualizador y de la interfaz hombre-máquina; y finalmente la sección 3.9, *Interfaces entre módulos*, donde se explican las distintas señales utilizadas para comunicar los módulos que componen el circuito.

3.1.PLACA DE DESARROLLO

En esta sección se explica la elección de la placa de desarrollo sobre la que se ha implementado el circuito, así como sus principales características y componentes utilizados en el proyecto.

3.1.1.ELECCIÓN DE LA PLACA DE DESARROLLO

Como se mencionó en la sección 2.1, *FPGAS y placas de desarrollo*, la importancia y el mercado de las FPGAs y las placas de desarrollo basadas en ellas no ha hecho más que incrementar desde su aparición en la década de los 80s. Hoy en día, existen multitud de empresas dedicadas a la fabricación de FPGAs y placas de desarrollo, sin embargo, las más extendidas son Xilinx y Altera, controlando el 90% del mercado de FPGAs en 2016 [20]. Debido a la gran cantidad de modelos disponibles en el mercado, es necesario establecer unos criterios de elección para filtrar entre las distintas opciones. En este caso, es imprescindible que la placa cuente con micrófono y salida de audio, que pueda ser programada mediante USB, y que cuente con botones o interruptores para que el usuario pueda interactuar con el circuito. Respecto a la capacidad lógica, ha de elevada puesto que se desconoce el porcentaje de utilización del circuito de la lógica disponible en la FPGA hasta que se implemente.

La única placa de desarrollo basada en FPGA encontrada que incluye micrófono y salida de audio ha sido el modelo Nexys A7, el resto solo incluyen entradas *jack* para conectar micrófonos externos. Esta placa está basada en la familia de FPGAs Xilinx Artix®-7 y, pese a estar enfocada principalmente a la enseñanza, posee una gran capacidad lógica, una memoria RAM DDR2 externa y numerosos puertos, periféricos e interruptores que la hacen apta para una gran variedad de proyectos. Además, se encuentra disponible en el laboratorio de sistemas electrónicos de la ETSIT, ahorrando costes adicionales.

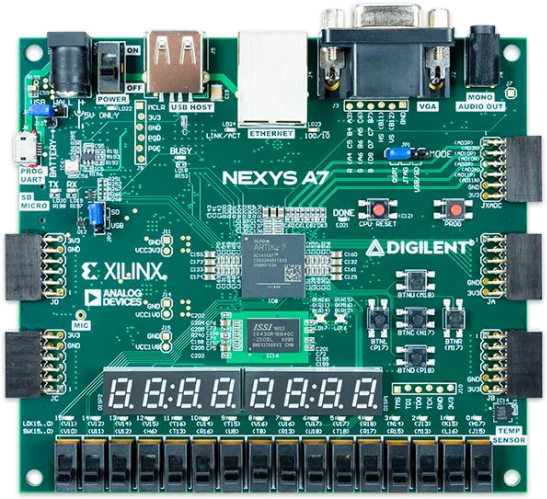

Figura 3.1. Vista frontal de la Nexys A7 [10].

3.1.2.ENTRADAS Y SALIDAS DE LA PLACA DE DESARROLLO

De esta placa, en concreto se ha hecho especial uso del micrófono y del puerto *jack* para salida de audio incorporados en la placa y descritos a continuación:

3.1.2.1. MICRÓFONO PDM ADMP421

Es un micrófono MEMS omnidireccional de baja potencia capaz de codificar la señal analógica de entrada a una señal de audio digital PDM mediante un modulador Sigma-Delta de cuarto orden (véanse las secciones 2.3.3.1, *Modulación PDM* y 2.3.3.3, *Modulador* Sigma-Delta). Su respuesta en frecuencia a la entrada corresponde a un filtro paso banda con una frecuencia de corte inferior de 100 Hz y una superior de 15 kHz, equivalente a casi la totalidad del rango audible por el ser humano, que es hasta 20 kHz en el mejor de los casos.

El micrófono cuenta con cinco pines: dos de ellos dedicados a la alimentación, sobre los que no tenemos control en esta placa, y otros tres dedicados a entradas y salidas, en los que nos centramos:

- DATA: es la salida de la señal PDM generada por el micrófono.
- CLK: es la entrada para la señal del reloj del micrófono, que define la frecuencia los pulsos del PDM (ver sección 2.3.3.1, *Modulación PDM)*, la cual ha de estar comprendida entre 1 y 3.3 MHz. En este caso se ha usado una frecuencia de 2.5 MHz, ya que es divisor de 100 MHz, la frecuencia de reloj de la FPGA.
- L/R Select: es una entrada que permite seleccionar si los datos estarán disponibles en los flancos positivos o negativos de la señal de reloj. Esta opción sería útil en el caso de que hubiera dos micrófonos, ya que se podrían obtener los datos de cada uno en los distintos flancos y generar una señal de audio estéreo. Sin embargo, la placa solo contiene un micrófono por lo que es irrelevante.

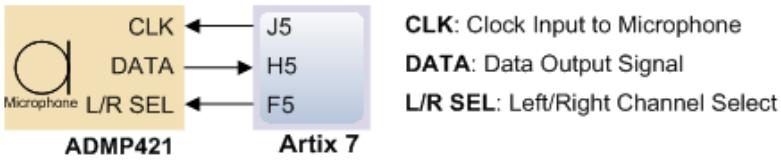

Figura 3.2. Esquema del micrófono ADMP421 [10].

3.1.2.2. SALIDA DE AUDIO

Consiste en un filtro de Butterworth paso bajo de cuarto orden implementado mediante dos topologías Sallen-Key en serie, generando una frecuencia de corte de 13 kHz. Este filtro permite transformar una señal digital PDM o PWM a una señal analógica, conectada a una salida jack de 3.5 mm para su reproducción en altavoz o auriculares. En este trabajo, la salida se ha codificado en PDM, el motivo se explicará en la sección 3.4, *Decodificación de PCM y codificación a PDM*.

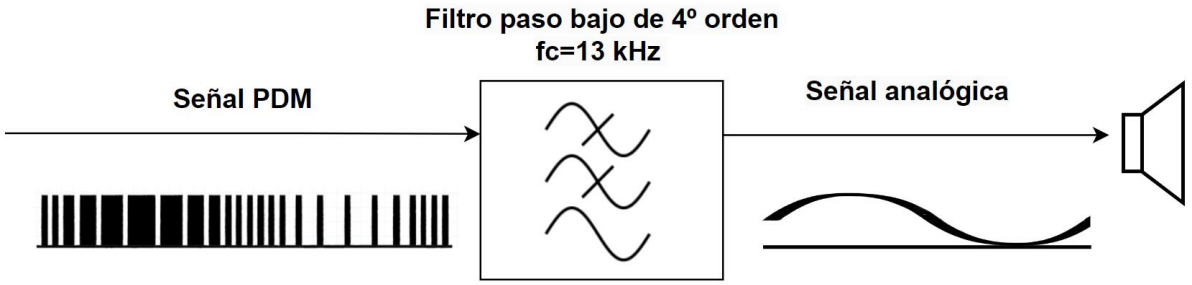

Figura 3.3. Funcionamiento del filtro a la salida de audio de la Nexys A7.

Además de los elementos mencionados, se ha hecho uso de los interruptores para implementar una interfaz hombre-máquina que permita al usuario seleccionar el rango de frecuencias a ecualizar y reiniciar el sistema.

3.2.MUESTREO DEL MICRÓFONO

La primera decisión tomada en este proyecto fue determinar la forma en que se muestrearían los datos del micrófono. En el manual de referencia de la Nexys A7 se indica un método para muestrear los datos PDM del micrófono y modularlos en PCM mediante contadores, explicado con el ejemplo de la Figura 3.4. En ella, el reloj del micrófono se fija a 2.4 MHz y se muestrea el audio para obtener una señal PCM de 7 bits a 24 kHz (esa sería la frecuencia de muestreo f_s, cuyo periodo es $T_s = 41.6\,\mu s$). Esto se consigue mediante dos contadores que cuentan hasta $2^7 = 128$ muestras durante dos periodos de muestreo ($2T_s = 83.2\,\mu s$) con un desfase de T_s para empezar a contar.

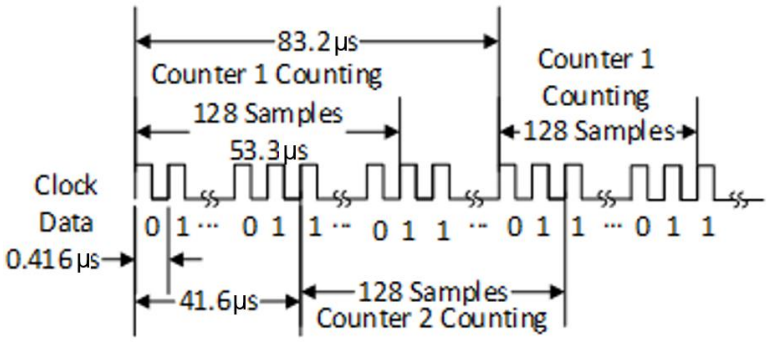

Figura 3.4. Muestreo del micrófono recomendado [10].

El problema de este método es que es complejo de escalar si se requieren mejores prestaciones. Por ejemplo, si se desea obtener una señal de 16 bits muestreada a 48 kHz con un reloj del micrófono a 3 MHz (lo máximo que soporta), en cada periodo de muestreo el número máximo de pulsos que se podría contar se puede calcular con la siguiente expresión:

$$\left\lfloor \frac{T_s}{T_{mic}} \right\rfloor = \left\lfloor \frac{f_{mic}}{f_s} \right\rfloor = \left\lfloor \frac{3\ MHz}{48\ kHz} \right\rfloor = 62\ pulsos \qquad (20)$$

Donde T_s es el periodo de muestreo y T_{mic} es el periodo de reloj del micrófono.

En este caso, para tener una señal de 16 bits se necesitarían contadores que contasen hasta $2^{16} = 65536$ pulsos. Si el número máximo de pulsos que se pueden contar en cada periodo de muestreo es 62, entonces el número de periodos de muestreo para llegar a contar 65536 pulsos se obtiene con la siguiente expresión:

$$\left\lceil \frac{65536}{62} \right\rceil = 1058\ periodos\ de\ muestreo \qquad (21)$$

Esto quiere decir que el muestreador devolvería el primer dato 1058 periodos de muestreo después de comenzar y se requerirían 1058 contadores, cada uno desfasado un periodo de muestreo, para devolver un dato cada T_s. A pesar de que este método podría implementarse, es una forma poco común y compleja de muestrear señales PDM.

Por otro lado, la forma más común y sencilla de muestrear señales PDM para obtener señales PCM es mediante un filtro diezmador, tal y como se explicó en la sección 2.3.4.1, *Filtro diezmador*. Se ha optado por esta segunda opción debido a la cantidad de información al respecto y su mayor facilidad de implementación mediante IP cores, especialmente útiles dadas las restricciones de tiempo.

3.3. DECODIFICACIÓN DE LA SEÑAL PDM PROVENIENTE DEL MICRÓFONO

Como se mencionó en la sección 2.3.4.1, *Filtro diezmador*, en este proyecto se ha llevado a cabo un procesamiento de datos basado en FFTs, cuya entrada ha de estar modulada en PCM, por lo que es imprescindible codificar la señal PDM generada por el micrófono a PCM. Para ello, se ha diseñado el módulo de diezmado, cuyo objetivo es transformar la señal PDM a 2.5 MHz a una señal PCM de menor frecuencia.

Antes de diseñar el filtro es necesario especificar las características de la señal PCM que se introducirá en la FFT. Tras investigar los anchos de palabra y frecuencias de muestreo más comúnmente utilizados se ha decidido utilizar 16 bits para representar la señal. Son tres los motivos por los que se ha elegido este ancho de palabra: en primer lugar, proporciona un SNR alto, concretamente de 98.8 dB para una señal senoidal según la ecuación (3), un valor apto para la mayoría de aplicaciones profesionales; en segundo lugar, el número de bits es relativamente bajo, manteniendo los archivos ligeros y el tiempo de procesamiento bajo; en tercer lugar, es un ancho de palabra estandarizado utilizado en gran variedad de aplicaciones.

Respecto a la frecuencia de muestreo, las más comunes son 44.1 kHz y 48 kHz, de las cuales se ha optado por la segunda. A primera vista, se podría pensar que no tiene sentido utilizar una frecuencia de muestreo mayor de 44.1 kHz, ya que, según el teorema de Nyquist, está permitiría muestrear señales con un ancho de banda de hasta 22.05 kHz, algo más del espectro audible por el ser humano. No obstante, es recomendable utilizar una frecuencia mayor debido al aliasing. El motivo es que, para evitar el aliasing, es necesario aplicar un filtro paso bajo a la señal de entrada que limite el ancho de banda a la mitad de la frecuencia de muestreo, pero cuanto menor sea la frecuencia de muestreo más abrupto ha de ser el filtro anti-aliasing para cumplir con el teorema de Nyquist sin eliminar parte de las frecuencias audibles. Usando una frecuencia de muestreo más alta, el filtro no tendrá que ser tan abrupto, que se traducirá en una reducción de área y consumo.

Para el diseño del filtro diezmador, ya sea implementado mediante filtros CIC o FIR, se han seguido las siguientes pautas de diseño:

- La entradas del filtro se han fijado a 8 bits, el menor valor posible permitido en los IP cores de Vivado, por lo que se usan los 2 LSBs para representar el valor PDM en complemento a 2 y el resto de bits se ponen a 0.
- Para la salida y los registros internos del filtro se ha utilizado precisión completa, pero el resultado final se ha truncado a 16 bits.
- Para pasar de los 2.5 MHz de la señal PDM a 48 kHz se requiere un factor de diezmado de 52. Como este factor ha de ser un número entero, la frecuencia de muestreo será realmente de 48076.92 Hz, pero a lo largo de este trabajo solo se referirá a ella como 48 kHz por simplicidad.
- La frecuencia de corte es próxima a los 13 kHz, ya que esa es la frecuencia de corte del micrófono. A partir de los 13 kHz la señal captada por el micrófono será muy débil y apenas audible por lo que no se pierde información descartando las frecuencias que están por encima.

Para elegir el tipo de filtro que mejor se adapta al sistema y seleccionar sus parámetros de implementación a nivel hardware es necesario realizar experimentos y comparaciones entre las prestaciones de ambos, lo cual se ha realizado en la sección 4.2, *Análisis de filtros CIC y FIR*.

Además de los parámetros del filtro, es necesario diseñar una interfaz para que el filtro pueda transmitir y recibir información del resto de módulos del circuito. Para ello, se ha reutilizaran las señales de los IP Cores para filtros CIC y FIR de Vivado, que son las mismas en ambos. Este IP Core utiliza una interfaz síncrona formada por dos señales: una señal de validación de entrada *valid_in*, para la cual se reutiliza la señal de reloj de 2.5 MHz usada para el reloj del micrófono, y una señal *valid_out*, que indica el procesamiento de un nuevo dato, la cuál será reutilizada como reloj de la FFT, IFFT y ecualizador, ya que tiene una frecuencia de 48 kHz.

3.4.DECODIFICACIÓN DE PCM Y CODIFICACIÓN A PDM

Mediante el filtro diezmador, se ha obtenido la señal PCM de 16 bits que se introducirá en la FFT para ser procesada. No obstante, tras su procesamiento es necesario convertirla de nuevo a PDM o PWM para su reproducción por la salida de audio. Antes de proceder es necesario decidir si se modulará la señal de salida en PDM o PWM. En primer lugar, hay que tener en cuenta que el procedimiento para convertir ambos tipos de señales moduladas a señales analógicas es el mismo: utilizar un filtro paso bajo analógico, el que está incluido en la placa, por lo que este factor no tiene importancia a la hora de elegir el tipo de modulación. Los factores relevantes en esta decisión son los métodos y requisitos para modular la señal y las limitaciones de cada tipo de modulación, los cuales se describen a continuación:

- PWM: este tipo de modulación consiste en la generación de una serie de pulsos a frecuencia constante que varían su anchura en proporción a la amplitud de la señal de entrada. La forma más común de generar una señal PWM es comparando la señal de entrada con una señal triangular. Para modular una señal PCM de 48 kHz y 16 bits se requerirían 2^{16}-1 divisiones de tiempo cada 48 kHz, por lo que se requeriría un reloj cuya frecuencia debería ser de como mínimo $65535 \cdot 48000 = 3.145\ \text{GHz}$, una frecuencia imposible de implementar en la FPGA, ya que su frecuencia máxima es de 100 MHz. Para 100 MHz, la anchura de palabra máxima con la que se podría generar un PWM se puede calcular como:

$$\#\ niveles\ discretos = \left\lfloor \frac{100\ MHz}{48\ kHz} \right\rfloor = 2083 \quad (22)$$

$$anchura\ de\ palabra\ máxima = \lceil \log_2(2083) \rceil = 12\ bits \quad (23)$$

  La anchura de palabra máxima que permite la modulación PWM para este caso es de 12 bits, insuficiente para esta aplicación.
- PDM: para transformar una señal PCM a PDM es necesario interpolar la señal PCM de 48 kHz a 2.5 MHz para obtener la señal sobremuestreada, y utilizar un modulador Sigma-Delta digital, para modular la señal en PDM y desplazar el ruido de cuantificación a las frecuencias no audibles. Esta opción no requiere una frecuencia exageradamente alta, basta con utilizar la misma frecuencia de sobremuestreo empleada por el micrófono, cuyo rango está comprendido entre 1 y 3.3 MHz, y permite un amplio rango para el ancho de palabra, que estará limitado por el filtro interpolador. En el caso de un FIR generado con el FIR Compiler de Vivado, este ancho puede estar comprendido entre 1 y 59 bits.

Es evidente que la modulación PWM no sirve en esta aplicación, por lo que se modulará en PDM mediante un interpolador y un convertidor Sigma-Delta digital.

3.5.INTERPOLACIÓN

El proceso de interpolación es muy similar al de diezmado, pero a la inversa. En la interpolación se incrementa la frecuencia de muestreo, apareciendo imágenes que han de ser atenuadas para recuperar la señal original. Para recuperar la señal original se aplica un filtro paso bajo tras la interpolación que atenúe las frecuencias superiores al ancho de banda al que habíamos limitado la señal original en el diezmado, por lo que la respuesta en frecuencia del filtro interpolador ha de ser la misma que la del filtro diezmador. Por este motivo, los parámetros de diseño del diezmador pueden ser reutilizados para este filtro, la única diferencia es que en este caso la entrada del filtro es de 16 bits, puesto que esa es la longitud de palabra que se ha fijado para la FFT. En la sección 4.2, *Análisis de filtros CIC y FIR*, se detallarán los parámetros del filtro implementado y sus resultados.

Respecto a la interfaz, es la misma que para el diezmador, pero en este caso se reutiliza la señal *valid_out* del diezmador para la señal *valid_in* del interpolador, cuya frecuencia es de 2.5 MHz. Otra consideración que se ha tenido en cuenta es que la señal de *reset* estará activa hasta que se procesen los primeros 1024 datos para no tener ruido hasta entonces en la salida. Esto se hará mediante la señal *IFFT_finish* (posteriormente se explicará), que indica que se han procesado los primeros 1024 datos.

3.6.MODULADOR SIGMA-DELTA

Tal y como se explicó anteriormente, los moduladores Sigma-Delta pueden ser utilizados como ADCs, capaces de transformar una señal analógica sobremuestreada a una señal digital PDM. De la misma forma, podemos reemplazar los componentes analógicos de este tipo de convertidores por sus equivalentes digitales para así transformar señales digitales PCM sobremuestreadas a PDM. Antes de implementar el modulador, es necesario decidir su orden, para ello hay que tener en cuenta ciertas consideraciones:

- Normalmente se utiliza el ENOB como medida de la precisión y fidelidad del convertidor, indicando la resolución efectiva del convertidor para el ancho de banda de la señal que se desea modular (f_B).
- El ENOB se calcula a partir de la fórmula de la potencia de ruido q_{rms}^2 dentro de f_B. Esta expresión es una estimación que se cumple si $OSR \gg 1$, en este caso el OSR es de 52. Para un modulador de orden L, la potencia de ruido se aproxima según la ecuación (24), donde e_{rms}^2 corresponde al ruido de cuantificación introducido por el ADC interno, que puede aproximarse como ruido blanco de valor cuadrático medio $\Delta^2/12$, siendo Δ el paso de cuantificación [12].

$$q_{rms}^2 = \frac{\pi^{2L} e_{rms}^2}{(2L+1)(OSR)^{2L+1}} \qquad (24)$$

El ENOB dentro de f_B se obtiene mediante la siguiente expresión:

$$ENOB = \log_2 q_{rms} \qquad (25)$$

Idealmente, el orden del modulador ha de ser lo mayor posible, ya que implica menos ruido en la banda de la señal, y por tanto mayor ENOB. No obstante, conforme aumenta el orden del modulador, también aumenta considerablemente su complejidad y cobran mayor importancia las consideraciones de estabilidad, que limitan las prestaciones realmente alcanzables. Por este motivo, se ha empleado el mínimo orden del modulador cuyo ENOB se corresponda con los bits de entrada, ya que esto asegura que el sistema aprovecha toda la resolución de la señal de entrada, evitando sobredimensionamientos innecesarios.

Si se evalúan las ecuaciones (24) y (25), se obtiene que un ENOB de 12 bits para un modulador de primer orden, un valor escaso para nuestra aplicación, y 19 bits para uno de segundo orden, un valor apto para nuestra aplicación ya que supera los 16 bits de la entrada. Por estos motivos, se ha empleado un modulador de segundo orden.

Existen varias formas de implementar un modulador de segundo orden, pero la más sencilla consiste en sustituir el ADC del modulador de primer orden de la Figura 3.5 por otro modulador de primer orden, resultando en el siguiente circuito:

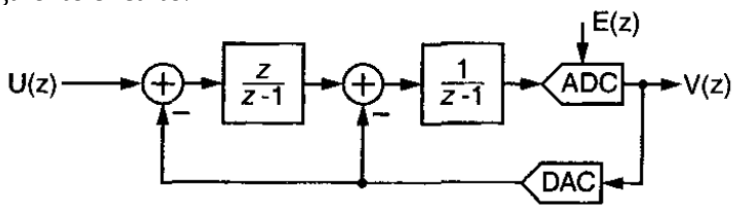

Figura 3.5. Modulador de segundo orden [12].

Si sustituimos el ADC y el DAC por su modelo lineal se obtiene:

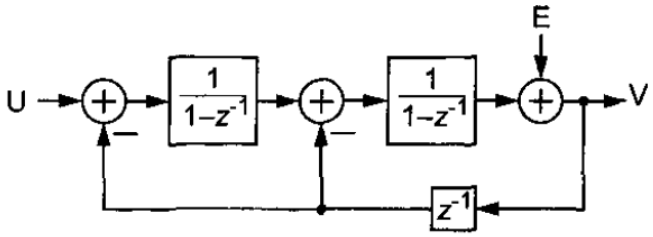

Figura 3.6. Modelo lineal del modulador de segundo orden [12].

A partir de este circuito, se puede extraer la siguiente ecuación, en la que se comprueba que el orden del filtro ha sido incrementado:

$$V(z) = U(z) + E(z)(1 - z^{-1})^2 \qquad (26)$$

Al implementar el modulador, es importante diseñar correctamente el DAC para que los valores PCM que genere coincidan con los valores máximos y mínimos de la entrada del modulador, que son inferiores a los máximos y mínimos representables con 16 bits en complemento a 2. Para obtener estos valores es necesario conocer el tipo de filtros interpoladores y diezmadores usados puesto que la entrada del modulador es la salida del filtro interpolador, por lo que estos valores se obtendrán en la sección 4.2.4, *Valores máximos de los filtros CIC*.

3.7.FFT E IFFT

El algoritmo de la FFT permite una enorme variedad de implementaciones, sin embargo, debido a las limitaciones de tiempo y a la complejidad que supone implementar una de las arquitecturas de FFTs desde cero, se incorporará una arquitectura SDF previamente diseñada y validada. No obstante, al tratarse de una implementación genérica, ha sido necesario realizar algunas modificaciones para adaptarla a las necesidades de nuestro circuito.

Antes de comenzar a diseñar, se ha estudiado el diseño de la FFT proporcionada, cuyas características se detallan a continuación. Esta emplea un ancho de palabra (WL) ajustable y sus entradas y salidas de datos son de 2WL, correspondiendo la parte imaginaria a los MSBs y la parte real a los LSBs. En cuanto a la interfaz, presenta una entrada de reloj, una entrada para reset asíncrono, una señal de inicio para indicar el comienzo de la recepción de datos en serie y una señal de finalización para indicar el comienzo de la salida de los resultados en serie.

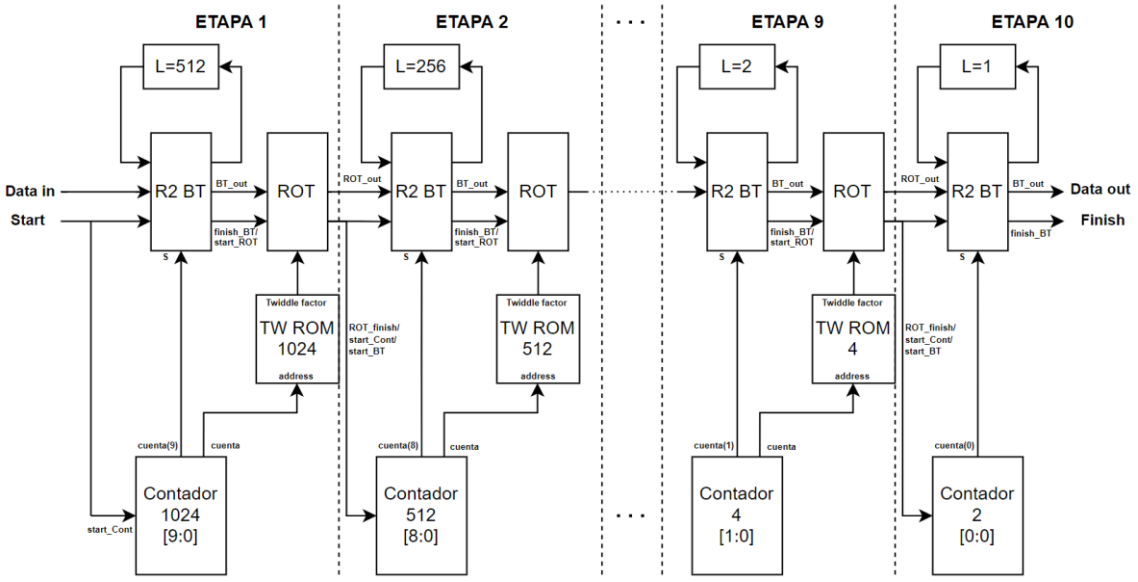

Figura 3.7. Estructura global de la implementación de la FFT.

Esta implementación utiliza una estructura modular parametrizada de 10 etapas, compuesta por 10 mariposas, 9 rotadores, 9 memorias ROM diferentes y delays, todos instanciados desde una entidad principal que contiene el controlador, representado en la Figura 3.7. A continuación, se explica de forma resumida la estructura de cada módulo:

- Mariposa (R2_BT_SDF): La estructura general de cada mariposa se muestra en la Figura 3.8, y está controlada por la señal "S". Inicialmente S es 0, por lo que los datos de entrada pasan al buffer (cuyo tamaño L dependerá de la etapa de la FFT) y a la entrada de la mariposa (R2). En estas condiciones, el resultado de la mariposa es descartado ya que el buffer aún no está lleno. Cuando el buffer se llena, S comienza a valer 1 y los resultados de la mariposa son válidos. La mariposa devuelve dos resultados, pero solo uno de ellos pasará al rotador, ya que el otro ya que el otro, que ha de salir L ciclos más tarde, se vuelve a introducir al buffer. Por ello, tras L ciclos, S vuelve a valer 0 y los resultados del rotador almacenados en el buffer pasan al rotador.

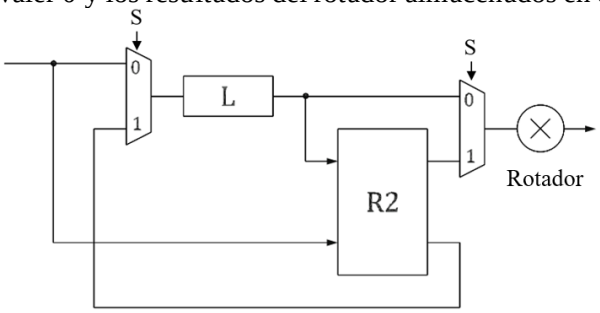

Figura 3.8. Estructura general de las mariposas [8].

  Cabe destacar que las operaciones que lleva a cabo R2 son dos sumas de palabras de WL bits, por lo que los resultados deberían ser de WL+1 bits, lo cual se conoce como crecimiento de bits. Sin embargo, en este caso se han limitado los resultados de R2 a WL bits, quedándose con los MSBs, lo cual equivale a dividir entre 2. Esto quiere decir, que el resultado de la FFT completa estará dividido entre 2^n, donde n es el número de etapas.
  Además de esto, las señales de entrada están registradas para segmentar el circuito en cada etapa, por lo que la latencia de cada mariposa es de L+1 ciclos.
- Rotador (rotator): está basado en 3 secciones DSP, a las cuales se les asignan parte de las operaciones necesarias. Para una entrada $x + jy$ y unos coeficientes $C + jS$, el DSP1 obtiene:

$$(x - y)S \tag{27}$$

  El DSP2 obtiene la parte real del real del resultado:

$$(C - S)x + (x - y)S = x \cdot C - y \cdot S = u \tag{28}$$

  Y el DSP3 obtiene la parte imaginaria del resultado:

$$(C + S)y + (x - y)S = x \cdot S + y \cdot C = v \tag{29}$$

  Ya que algunas operaciones dependen de los resultados de las anteriores, se requerirían por lo menos 5 ciclos de reloj para obtener cada resultado (teniendo en cuenta que el DSP2 y DSP3 trabajan en paralelo). Para mejorar las prestaciones, se ha segmentado el circuito en 6 etapas separadas por registros de forma que presenta una latencia de 6 ciclos, sin embargo, a partir de ahí extraerá un nuevo resultado cada ciclo de reloj. Cabe mencionar además que los rotadores y mariposas interactúan entre ellos mediante una interfaz síncrona, con señales de inicio y finalización como las anteriormente descritas.
- Memorias ROM (TW_ROM): hay una memoria diferente por cada etapa, que almacenará las rotaciones correspondientes al algoritmo DIF con radix-2. El controlador genera diferentes contadores con los que se accede a la posición de memoria adecuada para cada entrada del

rotador, no obstante, esta señal no tiene en cuenta la latencia de la mariposa anterior al rotador, por lo que a la hora de generar los coeficientes de la memoria estos han de ser desplazados circularmente un número de posiciones L+1.

- Controlador: se ha implementado mediante contadores, uno para cada etapa, que permite contar hasta el valor 2L. El contador de la primera etapa comienza a contar cuando se activa la señal de inicio de la FFT y el resto comienzan cuando se activa la señal de finalización del rotador de la etapa anterior. Estos contadores indican la dirección de memoria de las ROM a las que se debe acceder en cada momento y el MSB sirve como señal de control "S" para las mariposas, ya que se mantiene a 0 durante L ciclos, a 1 durante los L ciclos restantes y así sucesivamente.

La latencia del circuito completo se puede calcular como la suma de la latencia de cada mariposa más la suma de la latencia de los rotadores. Al tener 10 etapas, la latencia producida por las mariposas es:

$$Lat_{marip} = \sum_{i=1}^{10} (L_i + 1) = 513 + 257 + 129 + 65 + 33 + 17 + 9 + 5 + 3 + 2 = 1033\ ciclos \quad (30)$$

La latencia producida por los rotadores es:

$$Lat_{rot} = 6 \cdot 9 = 54\ ciclos \quad (31)$$

En total, la latencia de la FFT es:

$$Lat_{FFT} = Lat_{marip} + Lat_{rot} = 1087\ ciclos \quad (32)$$

Además, se ha estimado el SNR de la FFT para WL=16 en la sección 4.4, *Validación de la FFT e IFFT*, resultando en 56 dB, una calidad apta para la mayoría de aplicaciones de audio.

Una vez familiarizados con la implementación de la FFT se ha comenzado a diseñar la IFFT, la cual se ha de realizar tras el procesado de la señal en el dominio de la frecuencia para convertirla de nuevo al dominio del tiempo. No obstante, recordemos que la salida de la FFT viene dada en BR, por lo que existen dos opciones para procesar la señal e implementar la IFFT:

- La primera es utilizar un módulo que reordene las salidas de la FFT para pasar de BR a orden natural y otro módulo igual a la salida de la IFFT para que las salidas estén en el mismo orden que la entrada. Esta opción sería más simple desde un punto de vista conceptual ya que no habría que hacer cambios en la IFFT, sin embargo, eso supondría incluir dos módulos adicionales que incrementarían el área y la latencia del circuito.
- La segunda opción es adaptar el diseño de la IFFT para que acepte como entrada la señal en BR proveniente de la FFT. Esta opción es más compleja desde el punto de vista de diseño ya que requiere un nivel de comprensión más profundo tanto del algoritmo como de su arquitectura una vez implementado, sin embargo, minimiza el área y latencia. Por este motivo, se ha optado por esta alternativa.

El gráfico de flujo de una FFT con entrada en BR, se muestra en la Figura 3.9, para la IFFT será el mismo solo que las rotaciones están conjugadas y la salida se divide entre 2^n, siendo n el número de etapas. En este caso se divide entre 1024.

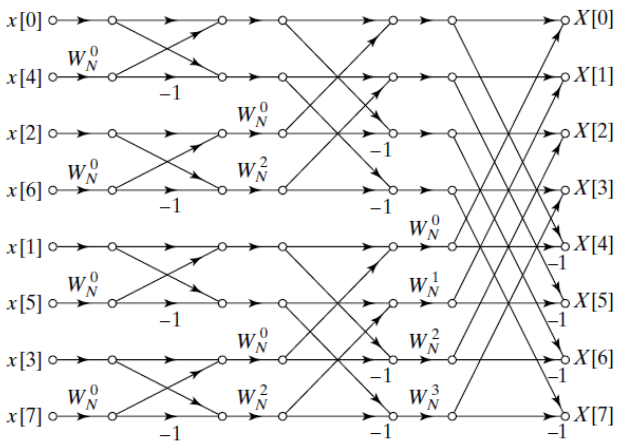

Figura 3.9. FFT de 8 puntos con entrada en BR [22].

Sin embargo, para llevar un proceso de diseño más gradual que permita depurar con mayor facilidad los posibles errores que surjan, antes de diseñar la IFFT con entrada BR, se ha diseñado una IFFT con entrada en orden natural. Para ello, basta con generar nuevas memorias ROM del mismo tamaño que las anteriores, pero con las rotaciones conjugadas. No hace falta implementar una división entre 1024 ya que esto ya sucedía debido a la limitación del crecimiento de bits. En estas condiciones, se ha estimado el SNR de la IFFT en la sección 4.4, obteniéndose 54 dB, por lo que la implementación es correcta.

Para la IFFT con entrada BR se han realizado los siguientes cambios:

- Como se muestra en la Figura 3.9, los rotadores están situados antes de las mariposas. Esto es problemático porque los contadores han de empezar la cuenta en el momento en el que las mariposas reciben el primer dato para que la señal "S" valga 1 durante L ciclos y 0 durante los L siguientes, pero esto no es posible porque los rotadores están colocados antes de la mariposa y necesitan que la cuenta comience antes de que las mariposas reciban el dato. Para solucionar este conflicto se han duplicado cada uno de los contadores, de forma que unos gestionan la señal "S" de las mariposas y los otros gestionan las direcciones de memoria de la ROM.
  Esta modificación afecta también a los coeficientes de la ROM, ya que ahora no le influye la latencia de la mariposa anterior, por lo que no es necesario desplazar las rotaciones L+1 posiciones.
- Los buffers de las mariposas están invertidos, ahora la mariposa de la etapa 1 tiene un tamaño de buffer 1, mientras que la de la última etapa tiene un buffer de 512 posiciones. Esto también afecta a los contadores del controlador, ya que hay que invertir el orden en el cada uno comienza a contar para que la señal de control "S" se adapte al tamaño de buffer de cada mariposa.
- El orden en el que se colocan las memorias ROM está invertido, empezando por la de 4 direcciones y finalizando por la de 1024.

En la Figura 3.10 se muestra un esquema general de la implementación de la IFFT con entrada en BR.

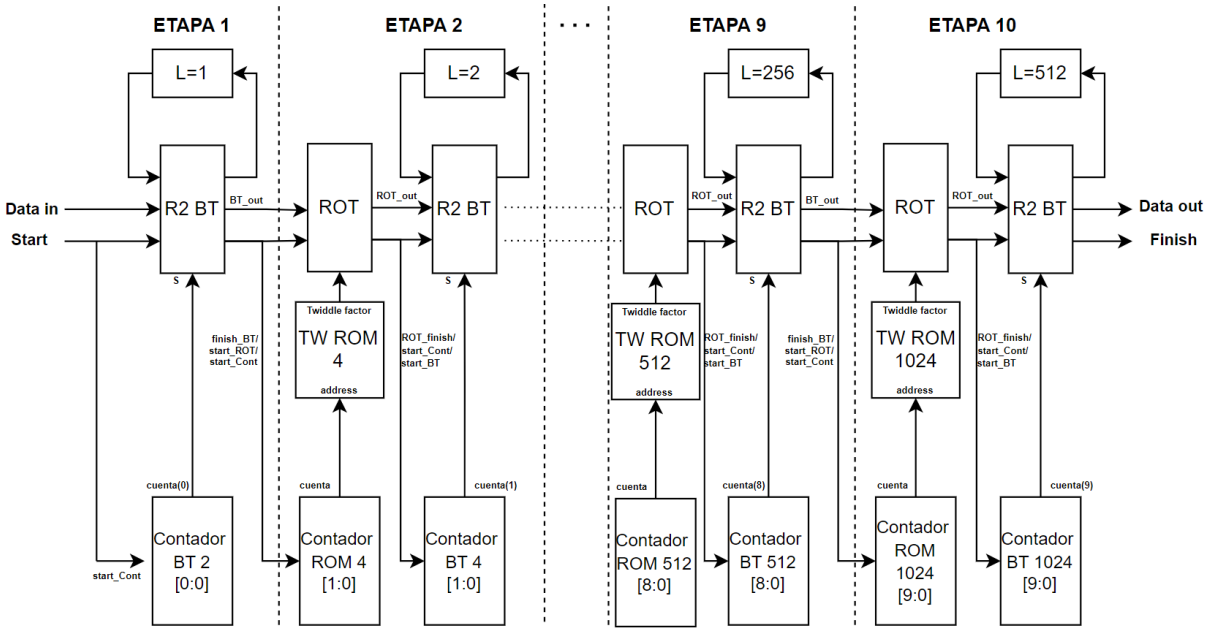

Figura 3.10. Estructura global de la implementación de la IFFT.

En la sección 4.4 se ha comprobado el correcto funcionamiento del circuito, obteniendo un SNR de 54 dB. Si se conectan en serie la FFT y la IFFT, el SNR disminuye a 25 dB. Esto sucede porque, al conectar los dos circuitos, el ruido de cuantificación de ambos se acumula y el SNR disminuye un 53.7%. Este SNR no es lo suficientemente elevado para que el ruido no sea perceptible. Para mejorarlo, se ha decidido no limitar el crecimiento de bits en todas las etapas de la FFT. Existen varias opciones, entre las cuales se ha elegido incrementar el ancho de palabra un bit cada dos etapas. Esta decisión se basa en la comparación entre los SNRs y el área del circuito FFT-IFFT según el número de bits del resultado final, explicada en más detalle en la sección 4.5, *Crecimiento de bits en la FFT*. Este problema no sucede en la IFFT, ya que esa limitación del crecimiento de bits genera la división entre N especificada en la ecuación (2).

Este error de cuantización solo se produce en las mariposas, donde se descarta el LSB, produciendo una división entre 2 por cada etapa. En los rotadores no hay cuantización ya que todas las rotaciones están dentro del círculo unitario. Por lo tanto, en el nuevo circuito la señal crece un bit cada dos etapas, obteniendo una salida de $2(WL + n/2)$, en este caso 42 bits. El SNR de este circuito es de 54 dB, adecuado para aplicaciones de audio.

3.8.ECUALIZADOR E INTERFAZ HOMBRE-MÁQUINA

Finalmente, se ha procedido al diseño del ecualizador. Este módulo permite la manipulación del audio por parte del usuario, por lo que está muy ligado a la interfaz hombre-máquina. Ya que la Nexys A7 tan solo cuenta con interruptores y pulsadores para que el usuario pueda interactuar con el circuito, no se puede establecer una interfaz hombre-máquina similar a la de los ecualizadores convencionales, que permiten modificar la amplitud de cada banda de frecuencia mediante controles deslizables. Por ese motivo, se ha optado por simplificar el diseño, de forma que cada interruptor corresponde a una banda de frecuencia, la cual puede estar activa o silenciada.

Antes de comenzar a diseñar, se han fijado las bandas de frecuencia asociadas a cada interruptor. El ancho de cada banda está limitado por la resolución de la FFT, que, al ser de 1024 puntos, proporciona una resolución en frecuencia de 46.92 Hz, como se demuestra en la ecuación (33).

$$\frac{f_{s\ FFT}}{N-1} \cong \frac{48\ kHz}{1023} = 46.92\ Hz \qquad (33)$$

Antes de comenzar con el desarrollo del ecualizador, es importante interpretar correctamente la salida de la FFT para saber a qué frecuencia corresponde cada dato de salida. Como la FFT es de 1024 puntos, por cada 1024 datos de entrada, devolverá 1024 datos de salida, cada uno asociado a una frecuencia. En este caso, la señal introducida en la FFT tiene un ancho de banda de 24 kHz, limitado por el filtro diezmador, y es muestreada y procesada a 48 kHz, de forma que el ancho de banda del sistema está limitado a esos 48 kHz. Por tanto, si la salida estuviese en orden natural, de esos 1024 datos de salida el primero correspondería a 0 Hz, el segundo a 46.92 Hz y el último a 48 kHz. Sin embargo, la salida de la FFT está en BR, por lo que el primer dato de salida está asociado a 0 Hz, el segundo a 23953.08 Hz, el tercero a 12000 Hz, etc. Este rango de frecuencias se ha dividido en las siguientes bandas de frecuencia:

- SW0: 0 - 1031.25 Hz y 46968.75 - 48000 Hz.
- SW1: 1078.13 - 2015.63 Hz y 45982.40 – 46920.82 Hz
- SW2: 2062.50 - 3000.00 Hz y 44997.06 - 45935.48 Hz
- SW3: 3046.88 - 4031.25 Hz y 43964.81 – 44959.15 Hz
- SW4: 4078.13 - 5015.63 Hz y 42979.47- 43917.88 Hz
- SW5: 5062.50 - 6000.00 Hz y 41994.14 - 42932.55 Hz
- SW6: 6046.88 - 7031.25 Hz y 40961.87 - 41947.21 Hz
- SW7: 7078.13 - 8015.63 Hz y 39976.54 - 40914.95 Hz
- SW8: 8062.50 - 9000.00 Hz y 38991.20 - 39929.62 Hz
- SW9: 9046.87 - 10031.25 Hz y 37958.94 - 38944.28 Hz
- SW10: 10078.13 - 11015.63 Hz y 36963.70 - 37912.02 Hz
- SW11: 11062.50 - 12000.00 Hz y 35988.27 - 36926.69 Hz
- SW12: 12046.88 - 13031.25 Hz y 34956.01 - 35941.35 Hz
- SW13: 13078.13 - 14015.63 Hz y 33970.67 - 34909.09 Hz
- SW14: 14062.50 - 24000.00 Hz y 24000.00 - 33923.75 Hz

En un principio se decidió realizar una división no uniforme de las bandas de frecuencias, utilizando anchos de banda más pequeños para modificar con mayor precisión las frecuencias comprendidas entre 100 y 5000 Hz, aproximadamente el rango de frecuencias de la voz humana. Sin embargo, tras probar el sistema de esta forma, se detectó que la interfaz no era intuitiva puesto que era difícil saber con certeza el rango de frecuencias que se estaba modificando. Por este motivo, se ha dividido el espectro en bandas de 1 kHz hasta los 14 kHz, cuya banda alcanza los 24 kHz.

Además, a cada interruptor le corresponden dos bandas de frecuencia simétricas respecto a la mitad de la frecuencia de muestreo. Como la señal que se introduce a la FFT es audio, se trata de una señal puramente real, las cuáles se caracterizan por tener un espectro simétrico (omitiendo la formulación matemática). Si se quiere mantener real es necesario modificar la salida de la FFT simétricamente, por eso a cada interruptor le corresponden dos bandas simétricas respecto a la frecuencia de Nyquist, en este caso 24 kHz.

Una vez definidas las bandas de frecuencia, se ha procedido al diseño de la arquitectura. El ecualizador está controlado por un contador de 1024 valores, el cual se reinicia cuando se recibe la señal de finalización de la FFT o cuando llega al final de la cuenta. Cada uno de estos valores del contador está asociado a dato de salida de la FFT en BR, por ejemplo, el valor 0 está asociado al dato de 0 Hz, el 1 a 23953.12 Hz, el 2 a 12000 Hz, etc. De esta forma, cada valor del contador lleva asociada una frecuencia, y cada frecuencia lleva asociado un interruptor. En base a esto, se ha implementado un circuito que transmite cada dato de entrada del ecualizador a la salida si su interruptor correspondiente vale 1. Si el interruptor vale 0, entonces se transmite un valor nulo a la salida.

En la Figura 3.11 se muestra un diagrama del sistema completo con el ancho de palabra de cada señal y su frecuencia de reloj.

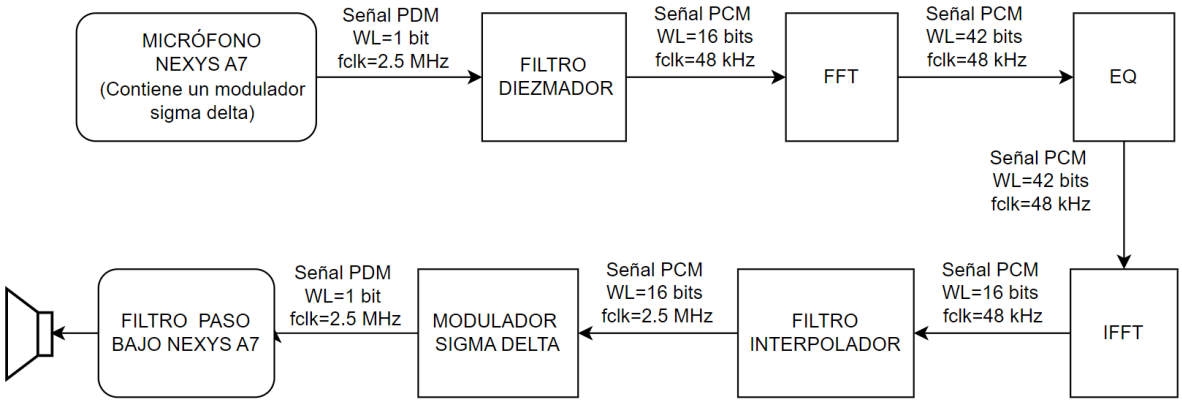

Figura 3.11. Diagrama del sistema completo.

3.9.INTERFACES ENTRE MÓDULOS

Una vez diseñado y comprobado el funcionamiento de cada uno de los módulos del circuito, se ha procedido a su integración en un único sistema, representado en la Figura 3.11. Para ello, cada módulo requiere una interfaz síncrona compuesta por un reloj y señales de validación, de inicio y finalización, aunque no en todos los módulos tienen las mismas señales ni se implementan de la misma forma:

- Diezmador: contiene una señal *valid_in* que indica que un dato disponible a la entrada y una señal *valid_out* que indica la salida de un nuevo resultado. A partir de esta, se genera una señal que sirva como señal de inicio de la FFT. Ya que la FFT acepta un dato cada ciclo de reloj y es de 1024 puntos, su señal de inicio debe activarse cada vez que hayan detectado 1024 flancos positivos de la señal *valid_out* del diezmador. Esto se puede implementar fácilmente mediante un contador de 1024 valores que incremente con cada flanco positivo de *valid_out* y que active la señal de inicio de la FFT cada vez que el contador está a 0.
- FFT e IFFT: tal y como se mencionó en la sección 3.7, *FFT e IFFT*, contienen una señal de inicio, una de finalización y su reloj será la señal *valid out* del diezmador, de 48 kHz. Cada mariposa tiene una señal de finalización que indica el comienzo de la salida de sus resultados en serie, por lo que la señal de finalización global de la FFT es la de la última mariposa.
- Ecualizador: este módulo procesa un dato cada ciclo de reloj con una latencia de un ciclo de reloj. Por lo que su señal de finalización es la señal de entrada retrasada un ciclo.
- Interpolador: El interpolador tiene la misma interfaz que el diezmador, pero su señal *valid_in* es el reloj de 48 kHz de la IFFT y su señal de *reset* está activa hasta que finalice la IFFT de los primeros 1024 datos. Para lograr esto, se ha generado la señal *IFFT_ready*, que se activa cuando ocurre el primer flanco positivo en la señal *finish* de la FFT y se desactiva cuando hay un *reset* global.
- El circuito cuenta con un *reset* global asíncrono activo a nivel bajo, que está conectado a los *resets* de los distintos módulos.

4. EXPERIMENTOS Y RESULTADOS

En este capítulo se explicarán los experimentos llevados a cabo para validar el comportamiento del sistema, analizar decisiones de diseño y estimar el efecto de estas sobre las prestaciones del circuito. Además, se discutirán y valorarán los resultados obtenidos. En concreto, el capítulo está dividido en 11 secciones: la sección 4.1, *Comprobación del micrófono y salida de audio*, se centra en la verificación de los componentes de la placa; la sección 4.2, *Análisis de filtros CIC y FIR*, trata el diseño, comparación y validación de filtros diezmadores e interpoladores CIC y FIR; las secciones 4.3, *Validación del modulador Sigma-Delta*, 4.4, *Validación de la FFT e IFFT* y 4.7, *Validación del ecualizador*, se centran en la validación de los distintos módulos del sistema; la sección 4.5, *Crecimiento de bits en la FFT*, analiza las distintas opciones de cuantificación en la FFT; la sección 4.6, *Módulo de reordenación en BR*, estudia el efecto en área y latencia que supone añadir módulos de reordenación de datos a las salidas de la FFT e IFFT; la sección 4.8, *Análisis del sistema completo*, analiza las señales más relevantes del sistema y verifican su correcto funcionamiento; la sección 4.9, *Prueba del sistema en placa*, donde se implementa el sistema completo en la placa de desarrollo para verificarlo desde el punto de vista del usuario, centrándose en la sensación auditiva; la sección 4.10, *Latencia del sistema*, donde se obtiene la latencia global del sistema y se proponen opciones para reducirla ; y finalmente la sección 4.11, *Impacto en el área y potencia*, donde se estima el impacto de las decisiones de diseño centradas en la optimización de área y potencia.

4.1. COMPROBACIÓN DEL MICRÓFONO Y SALIDA DE AUDIO

Una forma sencilla de validar el correcto funcionamiento del micrófono y de la salida de audio es conectar la señal PDM del micrófono directamente a la entrada del filtro paso bajo de la salida de audio y comprobar que se escucha correctamente. En la Figura 4.1 se muestra un esquema del circuito:

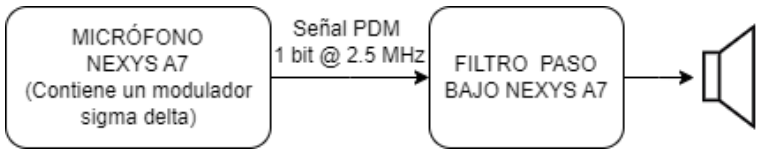

Figura 4.1. Circuito para la prueba del micrófono y la salida de audio.

Al llevar a cabo esta prueba, aparentemente trivial, se detectó que no se reproducía ningún sonido por la salida de audio. A partir de ahí se formularon dos hipótesis: la primera, que existiese algún problema en la interfaz entre micrófono y salida de audio, la cual fue descartada tras revisar el diseño, que era muy simple; la segunda, que la entrada o salida de audio estuviesen deshabilitadas mediante algún pin. Para comprobar esta hipótesis se revisó de nuevo la documentación relativa a la placa, sin encontrar ninguna información al respecto. Debido a esto se procedió con otro método: comparar nuestra implementación del diseño con la demo de audio para la Nexys A7 ofrecida por Xilinx [23]. Tras analizar la implementación se observó que se había declarado una salida llamada "PWM_AUDIO_0_en" conectada al pin D12, del que no se menciona nada en la documentación y que activa o desactiva la salida de audio. Al añadir esa salida en nuestro sistema se comprobó que funcionaba correctamente, reproduciendo a la salida el sonido captado por el micrófono.

4.2. ANÁLISIS DE FILTROS CIC Y FIR

Como se mencionó en la sección 2.3.4.1, Filtro diezmado, en Vivado se disponen de IP cores para filtros FIR y CIC, ambos capaces de implementar filtros diezmadores e interpoladores. Aunque se conocen las características generales de cada tipo de filtro, estas pueden variar en función de factores específicos de la aplicación, como la precisión requerida, el factor de diezmado e interpolación, los coeficientes del filtro FIR, etc. Para tomar una decisión más fundamentada entre estos dos tipos de filtros, se han implementado ambos y se han analizado sus prestaciones mediante los informes proporcionados por Vivado.

4.2.1.DISEÑO Y RESPUESTA DE LOS FILTROS CIC Y FIR

Para elegir entre los dos tipos de filtros, primero se han diseñado un CIC y un FIR adecuados al sistema, y luego se han comparado en términos de uso de recursos (área), consumo de potencia y respuesta transitoria.

En primer lugar, se ha diseñado el filtro CIC diezmador, cuyos parámetros de implementación a nivel hardware se han especificado mediante el CIC Compiler de Vivado. Además de los parámetros de diseño enumerados en la sección 3.3, *Decodificación de la señal PDM proveniente del micrófono*, solo hay 3 parámetros de diseño para un filtro CIC que permitan modelar su respuesta en frecuencia. Estos son el factor de diezmado R, el número de etapas N y el delay diferencial del filtro de peine M. El factor de diezmado está fijo a 52, por lo que solo se pueden alterar los otros dos parámetros. Normalmente, el delay diferencial se fija a 1 o 2, por lo que esas son las únicas opciones seleccionables en el compilador, mientras que el número de etapas puede ser de 3 a 6.

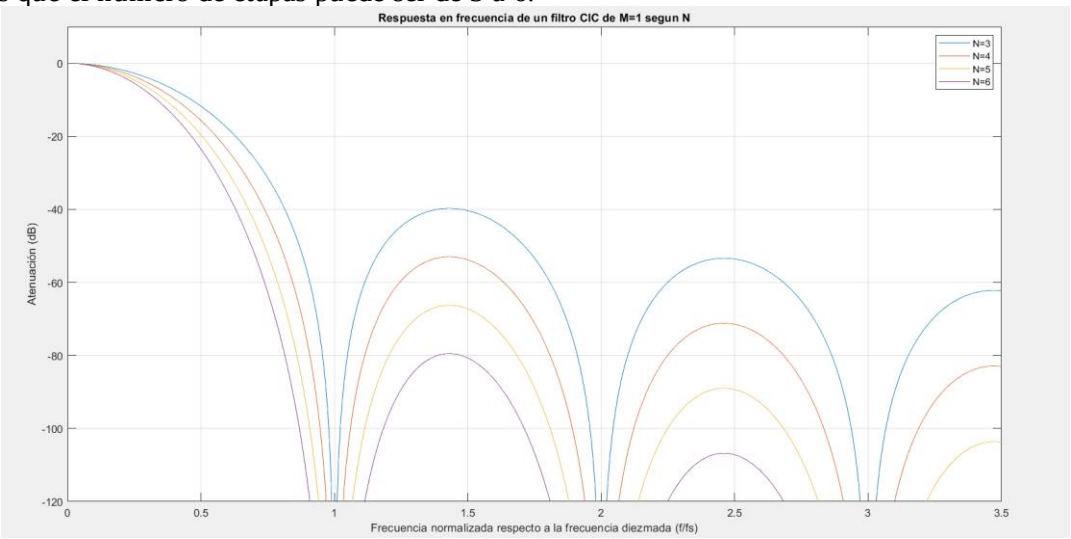

Figura 4.2. Comparación entre las respuestas en frecuencia de un CIC según su orden.

En la Figura 4.2 se ha representado a partir de la ecuación (16) la respuesta en frecuencia del CIC en función del número de etapas, manteniendo M=1. Se observa que, a medida que el número de etapas incrementa, disminuye la frecuencia de corte del filtro y aumenta la atenuación en la banda de rechazo, que comienza en 38976 Hz en todos los casos:

- Para N=3, la frecuencia de corte es de 12768 Hz y la banda rechazo presenta una atenuación mínima de 40 dB.
- Para N=4, la frecuencia de corte es de 10896 Hz y la banda rechazo presenta una atenuación mínima de 53 dB.
- Para N=5, la frecuencia de corte es de 9888 Hz y la banda rechazo presenta una atenuación mínima de 66 dB.
- Para N=6, la frecuencia de corte es de 8880 Hz y la banda rechazo presenta una atenuación mínima de 79 dB.

En vista de los resultados, la mejor opción es un filtro de orden 3, ya que la frecuencia de corte de los demás es demasiado baja. En cuanto a la atenuación en los 24 kHz, esta es de 11.81 dB, una atenuación baja pero suficiente teniendo en cuenta que a esas frecuencias solo hay ruido de baja potencia proveniente del modulador Sigma-Delta, ya que el micrófono no capta frecuencias tan elevadas.

Una vez elegido el orden, se ha analizado el efecto del delay diferencial, obteniendo los resultados de la Figura 4.3. Se observa, que los mínimos aparecen en las frecuencias relativas $1/M$, pero la atenuación mínima de la banda de rechazo se mantiene igual. Para M=2, la frecuencia de corte disminuye hasta 6192 Hz, lo cual supondría un recorte demasiado grande del espectro de la señal, por lo queda descartado.

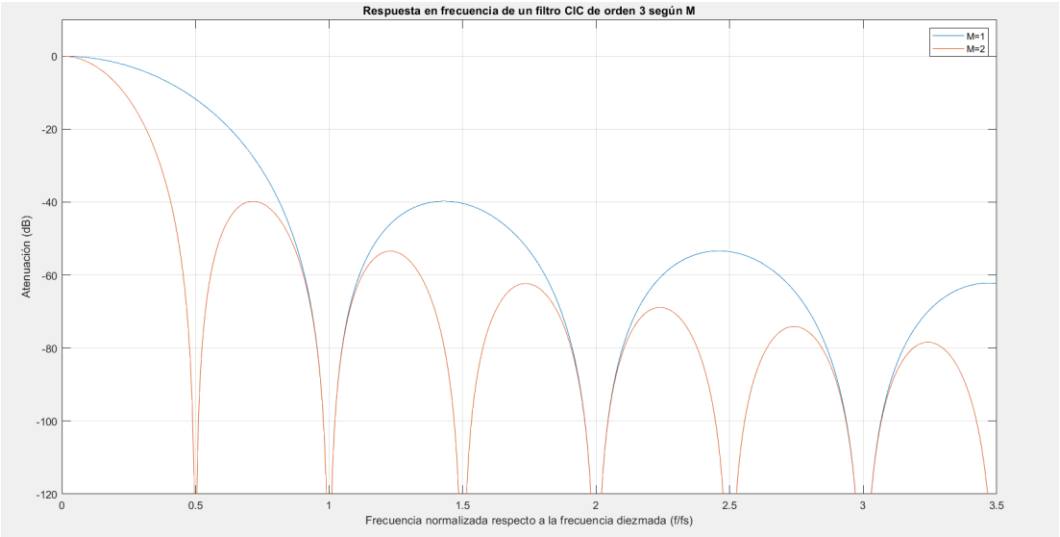

Figura 4.3. Comparación entre las respuestas en frecuencia de un CIC de orden 3 según el delay diferencial.

En resumen, los parámetros del filtro CIC diezmador diseñado son R=52, M=1 y N=3, con una frecuencia de corte de 12768 Hz y una atenuación de 11.81 dB a 24 kHz. Su respuesta en frecuencia se representa con la línea azul de la Figura 4.2 y la Figura 4.3.

A continuación, se ha procedido a diseñar un filtro FIR diezmador para compararlo con el CIC. Los parámetros de implementación de este filtro a nivel hardware se han especificado mediante el FIR Compiler de Vivado, mientras que los coeficientes que modelan la respuesta en frecuencia se han obtenido mediante la herramienta FilterDesigner de MATLAB. Las decisiones de diseño más relevantes son:

- Se han generado los coeficientes del filtro paso para tener una banda de paso de 13 kHz, ya que el micrófono apenas capta señal a partir de esta frecuencia, y una banda de rechazo de 60 dB a partir de 24 kHz, la mitad de la frecuencia de muestro. Estos parámetros de diseño resultan en un filtro de orden 549 y 550 coeficientes simétricos, cuya respuesta en amplitud se muestra en la Figura 4.4. Aunque estos valores parecen elevados, son asumibles por una FPGA de la serie Artix-7. Además, al tener coeficientes simétricos, el número de multiplicaciones y de coeficientes almacenados en la RAM se reducen a la mitad.

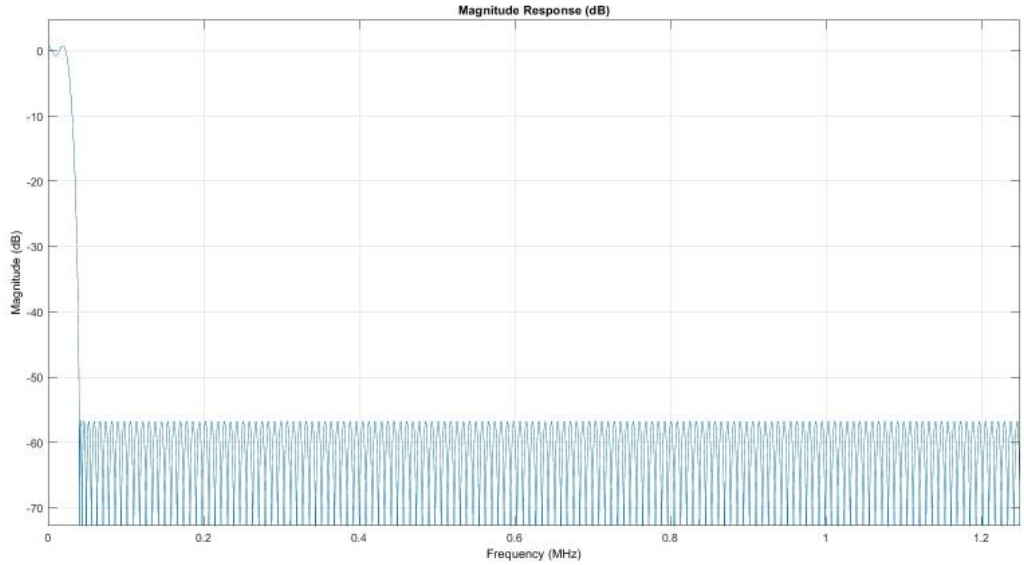

Figura 4.4. Respuesta en amplitud del filtro FIR diezmador e interpolador.

- Los coeficientes se han truncado a 16 bits fraccionarios, el mismo ancho de palabra que la señal PCM de salida, que también se cuantifica mediante truncamiento.
- El FIR Compiler ofrece la opción de optimizar el circuito para minimizar el área o para aumentar la velocidad del circuito, lo cual se consigue generalmente a costa de añadir etapas de segmentación en los caminos críticos. Se ha optado por la optimización de área porque la velocidad de procesamiento no es crítica en esta parte del circuito, ya que está limitada por la frecuencia del muestreo del micrófono.

Recapitulando, además del filtro CIC diezmador, se ha diseñado un filtro FIR de orden 559 con una frecuencia de corte de 13 kHz y una atenuación de 60 dB a 24 kHz, 550 coeficientes simétricos codificados con 16 bits y cuya respuesta en frecuencia se muestra en la Figura 4.4.

Finalmente, se han diseñado los filtros CIC y FIR interpoladores, cuyos parámetros de diseño son los mismos que los diezmadores, ya que requieren la misma respuesta en frecuencia que para el diezmado. La única diferencia es que el ancho de palabra de la entrada en este caso es de 16 bits, que provienen de la IFFT.

4.2.2.COMPARACIÓN ENTRE CIC Y FIR

Una vez diseñados ambos filtros, se ha procedido a la comparación de sus prestaciones. Para comparar los dos filtros, se han implementado dos circuitos iguales, pero con distinto tipo de filtro, formados por un diezmador en serie con un interpolador, representado en la Figura 4.5:

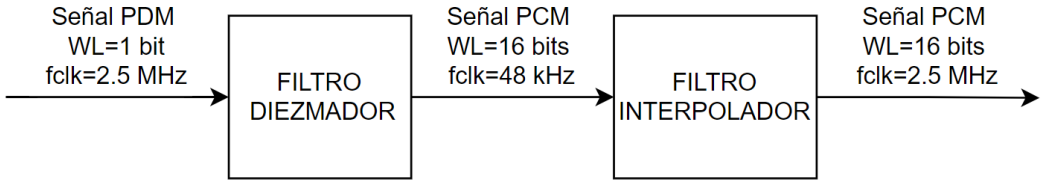

Figura 4.5. Circuito de prueba del filtro diezmador e interpolador.

En estas condiciones, los resultados del informe de implementación de Vivado son los siguientes:

- Área: los resultados se muestran en la Tabla 4.1. Como era de esperar, se observa un mayor porcentaje de utilización de los recursos disponibles por parte del sistema basado en el filtro FIR, aunque la diferencia no es tan grande como cabría esperar, ya que representa tan solo el 0.031% del total de componentes disponibles en la FPGA.

Bloque	CIC		FIR		Bloques disponibles
	Usados	Porcentaje	Usados	Porcentaje	
LUT	99	0.16%	318	0.50%	63400
LUTRAM	19	0.10%	148	0.78%	19000
FF	220	0.17%	418	0.33%	126800
DSP	2	0.83%	3	1.25%	240
BRAM	0	0%	1.5	1.11%	135
Porcentaje total usado	0.1622%		0.1937%		

Tabla 4.1. Comparación de área entre filtros CIC y FIR.

- Potencia: para estimar el consumo de potencia en Vivado es necesario aproximar las opciones de simulación lo más posible a las condiciones típicas de funcionamiento de la placa. Para la Artix-7 integrada en la Nexys A7, se han seleccionado las siguientes condiciones, que son las que se han usado para el resto de simulaciones de potencia de este capítulo:
  - La placa no incluye disipador de calor.
  - El flujo de aire se ha considerado 250 LFM (lineal feet per minute), que representa entorno sin ventiladores, donde la disipación de calor depende de la convección natural.

- Se han aproximado las dimensione de la placa a 4”x 4”, ya que es lo más similar a las dimensiones de la Nexys, que son 4.3”x 4.8”.
- La placa tiene 6 capas, por lo que se ha seleccionado la opción 4to7 layers.
- La resistencia térmica de la FPGA es de $\theta_{JA} = 15.4\ ºC/W$, para un encapsulado CSG324 y un flujo de aire de 250 LFM [24].

Los resultados muestran un consumo de 0.113 W para el CIC y 0.121 W para el CIC. Como se observa en la Figura 4.6, el consumo de este circuito es predominantemente estático, debido a las pequeñas corrientes de fuga que circulan por las múltiples interconexiones del circuito. Se ha estimado que esta componente del consumo es la misma para ambos, aunque el consumo dinámico es el doble en el FIR. Esto es un motivo más para decantarse por el CIC, aunque tampoco es crítico, ya que este consumo es mucho menor que el estático. En la Tabla 4.2 se muestra el reparto de consumo dinámico entre los distintos bloques del circuito.

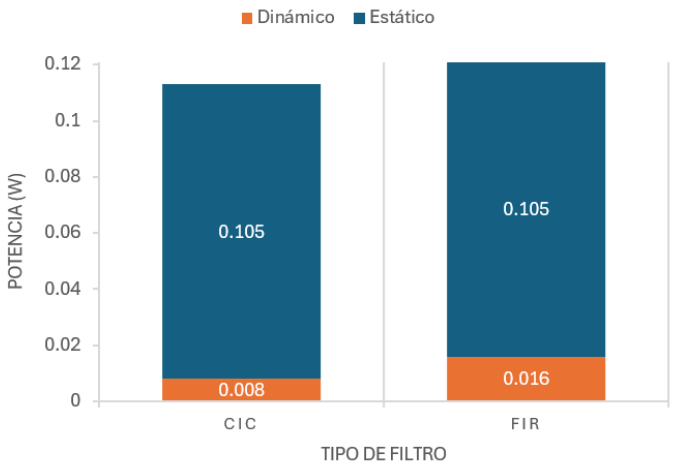

Figura 4.6. Comparación de consumo estático y dinámico entre FIR y CIC.

Bloque	CIC		FIR	
	Potencia (W)	Porcentaje	Potencia (W)	Porcentaje
Relojes	0.002	22%	0.006	36%
Señales	0.001	12%	0.002	10%
Lógica	0.001	9%	0.002	10%
DSP	0.001	15%	0.002	18%
BRAM	0.003	38%	0.004	26%

Tabla 4.2. Comparación de consumo dinámico entre FIR y CIC.

- Respuesta transitoria: cuando el filtro comienza a recibir una señal de entrada, no tiene suficientes muestras como para proporcionar una salida estable, por lo que se completan las muestras inexistentes con valores nulos. Esto provoca que la salida tarde un tiempo, conocido como tiempo de establecimiento, en alcanzar el estado estacionario. Su valor se suele estimar introduciendo una entrada de tipo escalón y midiendo el tiempo necesario para que la salida alcance un 95% del valor final.

  En la Figura 4.7 y Figura 4.8 se muestran las respuestas transitorias de un CIC y un FIR ante una entrada escalón, donde se ha medido el tiempo de establecimiento. Este tiempo es de 83.2 µs para el CIC y de 124.8 µs para el FIR, otro motivo más decantarse por el CIC.

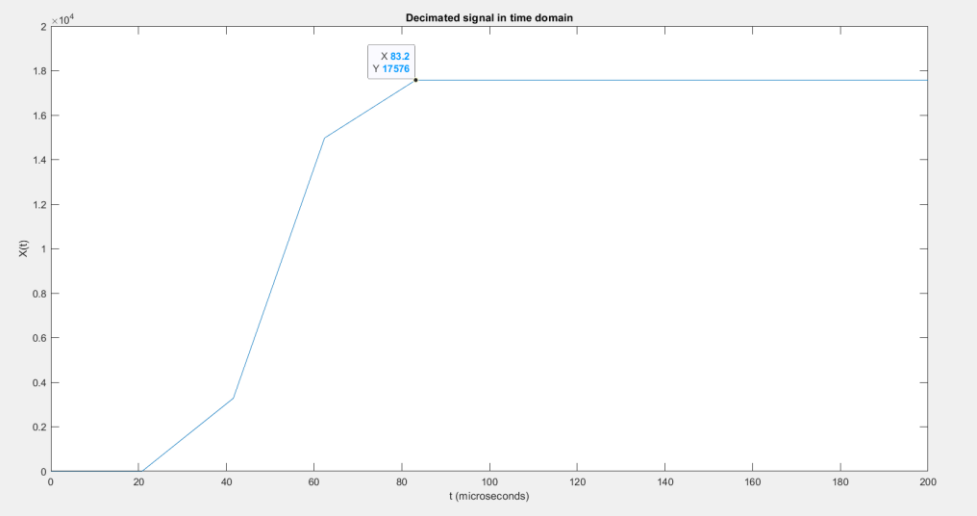

Figura 4.7. Respuesta transitoria del CIC diezmador ante impulso escalón.

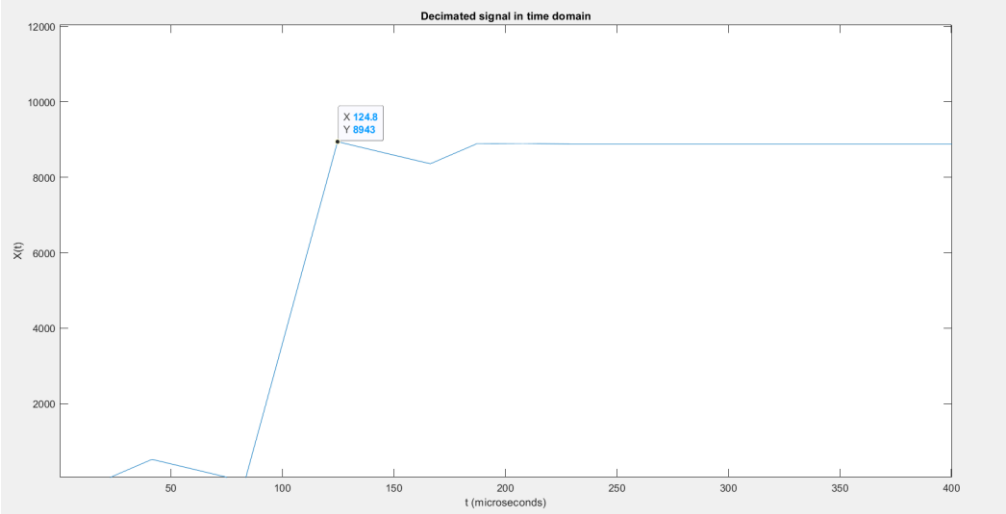

Figura 4.8. Respuesta transitoria del FIR diezmador ante impulso escalón.

Gracias a estas pruebas, se han verificado las ventajas de los filtros CIC frente a los FIR para el diezmado e interpolación, presentando menor área, consumo y tiempo de establecimiento. Por este motivo, se ha llegado a la conclusión de que los filtros CIC son más convenientes para este proyecto.

4.2.3. VALIDACIÓN DEL FILTRO CIC DIEZMADOR E INTERPOLADOR

Para comprobar que el funcionamiento del filtro CIC diezmador e interpolador es correcto se ha simulado el sistema descrito en la Figura 4.5. Para esta simulación, se ha introducido una señal senoidal ideal de 10 kHz modulada en PDM, y se han analizado los resultados en el dominio de la frecuencia, obteniéndose la Figura 4.9.

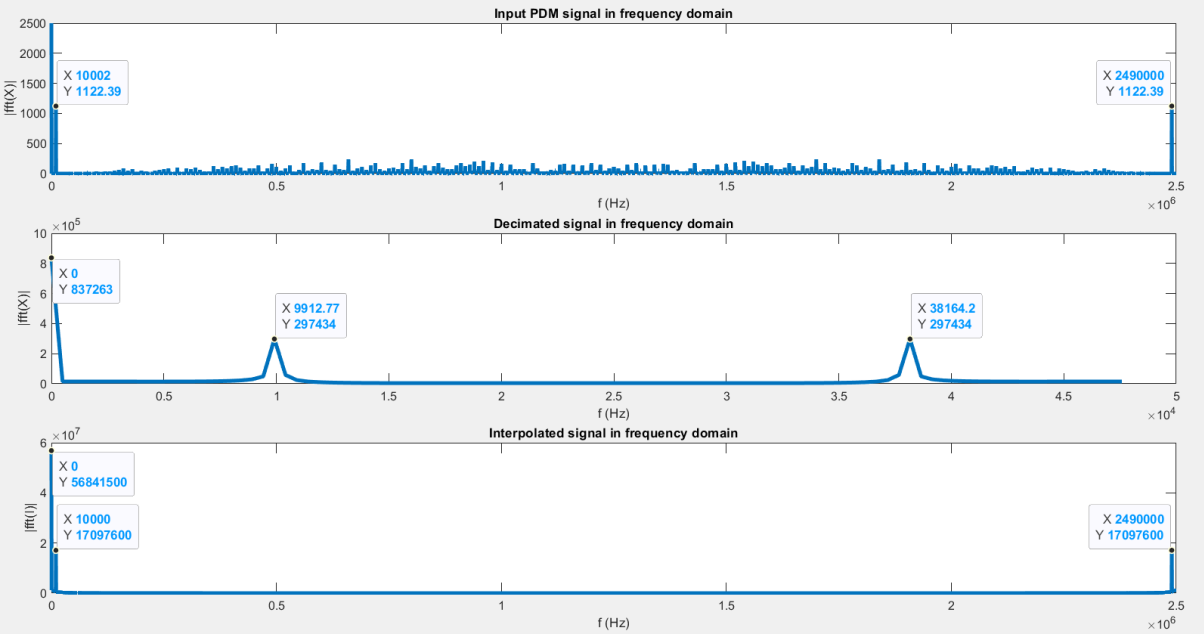

Figura 4.9. Señal PDM de entrada, señal tras el filtro diezmador y tras el filtro interpolador en el dominio de la frecuencia.

Se observa que la señal PCM obtenida tras el filtrado es correcta puesto que su densidad espectral está concentrada en 10 kHz y su frecuencia simétrica respecto a la frecuencia de muestreo. Sin embargo, tanto la señal PDM como las señales PCM obtenidas con los filtros contienen una componente continua no deseada. Este problema retrasó durante unas semanas el avance de este trabajo, llegando a considerar la opción de implementar un filtro paso banda que eliminase esa componente continua. Finalmente, se llegó a la conclusión de que esto era simplemente el comportamiento normal del filtro, ya que la señal PDM que se estaba introduciendo no estaba centrada en torno a 0, generando una componente continua. Para solucionar esto, simplemente hay que añadir un multiplexor a la entrada del filtro que introduzca el valor 1 si la señal PDM vale 1 y el valor -1 si la señal PDM vale 0. De esta forma, el resultado de la simulación es el deseado, tal y como se muestra en la Figura 4.10:

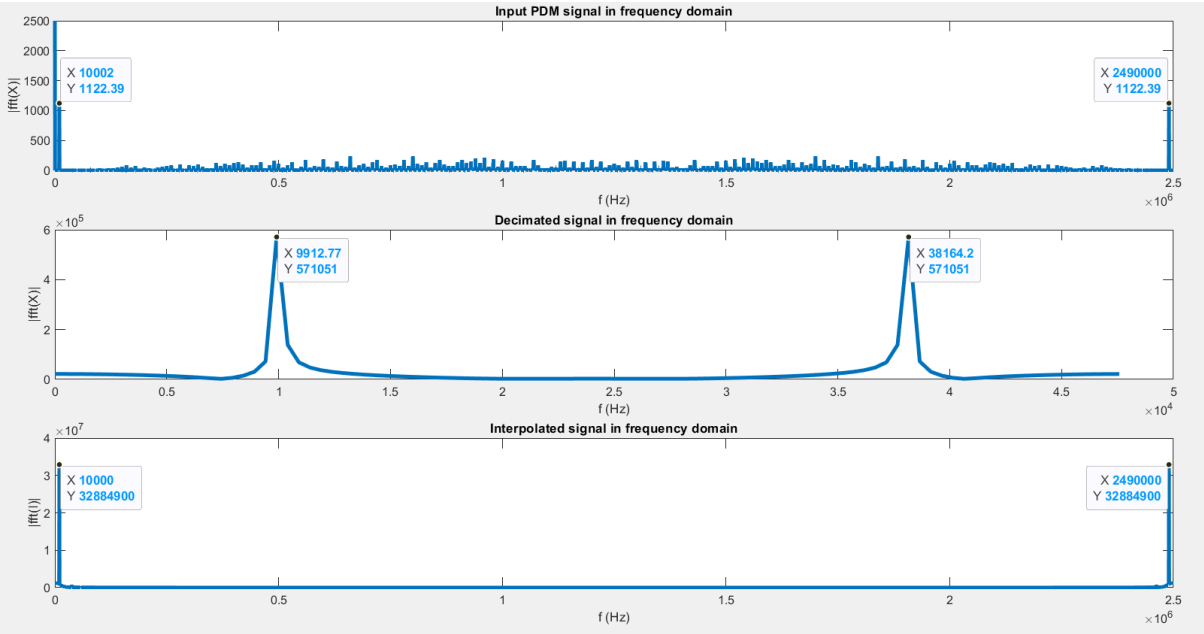

Figura 4.10. Señal PDM de entrada, señal tras el filtro diezmador y tras el filtro interpolador en el dominio de la frecuencia corregidas.

4.2.4. VALORES MÁXIMOS DE LOS FILTROS CIC

Finalmente, para diseñar el DAC del modulador Sigma Delta y aprovechar al máximo el rango dinámico del micrófono sin saturar la salida, es esencial conocer las amplitudes máximas y mínimas de la señal tras el diezmado y la interpolación. Para determinar estas amplitudes, se ha utilizado la expresión de la ganancia máxima del CIC diezmador, obtenida a partir de la ecuación (14) [19].

$$G_{max\ diezmador} = (RM)^N \quad (34)$$

Esta expresión es válida si se evalúa con frecuencia nula, por lo que se han considerado los siguientes casos:

- La salida del micrófono es 1 constantemente, en ese caso la amplitud tras el diezmado será máxima. El resultado es:

$$1 \cdot G_{max} = 1 \cdot (RM)^N = 140608 \quad (35)$$

  Se requieren 19 bits como mínimo para representar este valor en complemento a 2, por lo que, si se trunca la señal a 16 bits, el valor de amplitud máximo será 17576.
- La salida del micrófono es 0 constantemente, de forma que el filtro diezmador recibe -1. En ese caso la amplitud tras el diezmado será mínima. El resultado es:

$$-1 \cdot G_{max} = -1 \cdot (RM)^N = -140608 \quad (36)$$

  Al igual que en el caso anterior, se requieren 19 bits como mínimo para representar este valor en complemento a 2, por lo que, si se trunca la señal a 16 bits, el valor de amplitud mínimo será -17576.

Para determinar las amplitudes máximas de la salida del interpolador se ha llevado a cabo el mismo procedimiento que para el diezmador, pero en este caso la entrada del interpolador es la salida del diezmador. Se ha utilizado la expresión de la ganancia máxima del CIC interpolador, obtenida a partir de la ecuación (14) [19].

$$G_{max\ interpolador} = \frac{(RM)^N}{RM} \quad (37)$$

Esta expresión es válida si se evalúa con frecuencia nula, por lo que se han considerado los siguientes casos:

- La salida del diezmador es máxima constantemente, es decir, la entrada del interpolador es siempre 17576. En ese caso, la amplitud tras el diezmado será máxima. El resultado es:

$$17576 \cdot G_{max} = 17576 \cdot \frac{(RM)^N}{RM} = 47525504 \quad (38)$$

  Se requieren 27 bits como mínimo para representar este valor en complemento a 2, por lo que, si se trunca la señal a 16 bits, el valor de amplitud máximo será 23205.
- La salida del diezmador es mínima constantemente, es decir, la entrada del interpolador es siempre -17576. En ese caso, la amplitud tras el diezmado será máxima. El resultado es:

$$-17576 \cdot G_{max} = -17576 \cdot \frac{(RM)^N}{RM} = -47525504 \quad (39)$$

  Al igual que en el caso anterior, se requieren 27 bits como mínimo para representar este valor, por lo que, si se trunca la señal a 16 bits, el valor de amplitud máximo será -23205.

Mediante Vivado, se han simulado los dos casos y se ha comprobado que los valores máximos y mínimos calculados son correctos.

4.3.VALIDACIÓN DEL MODULADOR SIGMA-DELTA

Para comprobar el correcto funcionamiento del modulador Sigma-Delta de segundo orden diseñado, se le ha introducido una señal senoidal de 10 kHz con la amplitud 23205 calculada en el apartado anterior. En la Figura 4.11, se muestran los resultados, se observa que la señal de salida tiene un pico en 0 Hz, debido a la componente continua de la señal PDM representada mediante 1s y 0s, otro pico en 10 kHz, correspondiente a la señal senoidal, y una serie de picos de menor amplitud repartidos por las frecuencias fuera del ancho de banda de la señal, correspondientes al ruido modelado por el modulador.

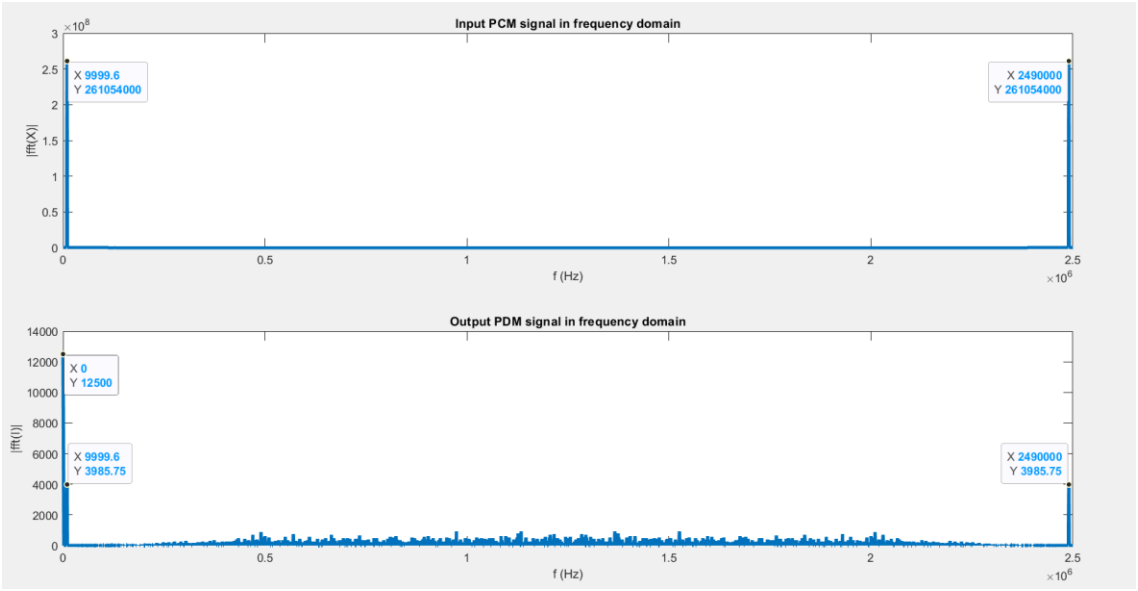

Figura 4.11. Entrada y salida del modulador Sigma-Delta de segundo orden en el dominio de la frecuencia.

Además, para comprobar el correcto funcionamiento del conjunto de los módulos diseñados hasta ahora se ha integrado e implementado en la placa el sistema de la Figura 4.12. Gracias a este circuito, se ha comprobado que la señal captada por el micrófono es reproducida a la salida con un volumen adecuado y un nivel de ruido aceptable, por lo que el diseño hasta este punto es correcto.

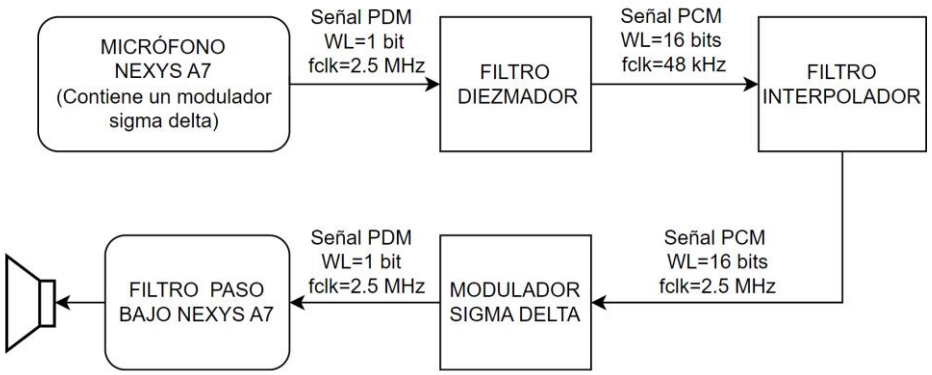

Figura 4.12. Sistema para la validación de los módulos de diezmado, interpolación y modulador Sigma-Delta.

4.4.VALIDACIÓN DE LA FFT E IFFT

Una vez implementados los circuitos para la FFT e IFFT, es necesario validar su funcionamiento y determinar su precisión, para ello se ha utilizado el SNR. Para llevar a cabo la validación, se simulan los distintos circuitos en Vivado mediante un *testbench* que lee las entradas de un fichero de texto y escribe las salidas en otro, sobre los datos de entrada se realiza en MATLAB la misma operación que hace el circuito y se compara la salida de MATLAB, que se supone ideal, con la salida de Vivado.

Respecto a los datos de entrada, se ha generado una distribución aleatoria de 1024 valores dentro del círculo unitario, que posteriormente es escalada a la amplitud máxima permitida por el ancho de palabra del sistema. Para ello, simplemente se multiplican los valores del círculo unitario por $2^{WL-1}-1$, que equivale a 32767 en este caso.

En cuanto a los datos de salida, para que los datos de la simulación tengan el mismo rango de valores que los de MATLAB, es necesario desescalar el valor devuelto por la simulación, dividiendo entre $2^{WL-1}-1$, y compensar la división producida por la limitación del crecimiento de bits. Además, en función del circuito que se esté probando, puede ser necesario reordenar los datos de entrada o de salida en BR para comparar los resultados gráficamente, ya que la salida de las funciones *fft(x)* e *ifft(x)* de MATLAB se devuelve en orden natural.

Una vez ajustadas las salidas de la simulación y de MATLAB, se ha estimado el SNR de la salida de la simulación dividiendo la energía de la señal entre la energía de ruido. La energía de una señal discreta es de N muestras es:

$$E[x[n]] = \sum_{n=0}^{N-1} |x[n]|^2 \qquad (40)$$

Si se asume que la salida de MATLAB es ideal, el ruido se puede estimar restando la salida de la función de MATLAB menos la salida de la simulación. En ese caso, el SNR en dB se obtiene según la ecuación:

$$SNR = 10 \cdot \log_{10}\left(\frac{E[x[n]]}{E[e[n]]}\right) \qquad (41)$$

Donde $x[n]$ es la salida de la simulación y $e[n]$ es el ruido:

Se han verificado los siguientes casos:

- FFT original: es la FFT que nos ha sido proporcionada, sin ningún tipo de modificación. Es importante comprobar su correcto funcionamiento y su precisión, ya que sirve como punto de partida para el resto de circuitos. En este caso se introduce la entrada de la simulación en orden natural y la salida, al devolverse en BR, ha de ser reordenada para compararla gráficamente con la salida de la función *fft(x)*. Además, al limitar el crecimiento de bits, el resultado de la simulación está divido entre 1024, por lo que hay que multiplicar por este valor para compensarlo en la comparación. El resultado es un SNR de 56 dB.
- IFFT con entrada en orden natural: es muy similar FFT original, ya que solo se han alterado las rotaciones de las memorias ROM. Al igual que en el caso anterior, la entrada se introduce en la simulación en orden natural y la salida se reordena para compararla gráficamente con la salida de la función *ifft(x)*. En este caso, como la IFFT divide la salida entre 1024, no hace falta compensar la salida de la simulación, ya viene dividida entre 1024 debido a la limitación del crecimiento de bits. El resultado es un SNR de 56 dB.
- IFFT con entrada en BR: en este caso, la entrada de la simulación se introduce en BR y no hace falta dividir la salida entre 1024 por el mismo motivo que en el caso anterior. El resultado es un SNR de 54 dB.
- FFT original en serie con IFFT con entrada en BR: la entrada y la salida de la simulación están en orden natural por lo que no es necesario reordenar los datos. En este caso se ha comparado la señal de entrada con el resultado de la simulación, el cual hay que multiplicar por 1024 debido a cuantificación de la FFT. El resultado es un SNR de 25 dB, casi la mitad del valor anterior, lo cual es insuficiente para proporcionar una buena calidad de audio.
  Para mejorar el SNR es necesario reducir la cuantificación. En la siguiente sección se valorarán las distintas opciones para lograrlo sin aumentar el área excesivamente.

4.5.CRECIMIENTO DE BITS EN LA FFT

En la sección 3.7, *FFT e IFFT*, se explicó la arquitectura de la FFT, cuyas mariposas realizan la suma de dos valores con el mismo ancho de palabra, por lo que ancho de palabra del valor resultante se incrementaría un bit. No obstante, cuando el número de etapas de la FFT es elevado, no es viable incrementar un bit en cada etapa, por lo que limita este crecimiento dividiendo el resultado de la mariposa entre 2. Esto supone otro problema, comprobado en la sección anterior, que es el aumento de ruido de cuantificación de la FFT. No obstante, esto no es un problema en la IFFT, ya su resultado debe ser divido entre N. Si se divide entre 2 en cada etapa y hay $\log_2 N$ etapas, la limitación del crecimiento de bits en cada etapa produce la división entre N sin añadir ruido de cuantificación.

En este caso, la FFT cuenta con solo diez etapas, por lo que el incremento de un bit por etapa es asumible, ya que el ancho de palabra resultante no es inusualmente grande en comparación con los valores típicos utilizados en circuitos digitales. Sin embargo, esto repercute negativamente en el área del circuito, que requiere una mayor cantidad de operadores, y en el consumo de potencia, que también aumenta con el área. En estos casos, es necesario llegar a un compromiso entre precisión y área. Para ello, se han analizado los siguientes casos:

- FFT con crecimiento de un bit cada etapa seguida de IFFT con entrada BR: el ancho de palabra del resultado de la FFT se incrementa 10 bits. En este caso, la amplitudes de las señales se corresponden con las amplitudes teóricas esperadas para la FFT e IFFT mostradas en la Figura 2.3. El resultado es un SNR de 79 dB, un valor muy elevado que hace este circuito apto para cualquier aplicación de audio. Si se compara con el circuito FFT-IFFT sin crecimiento de bits, se ha incrementado el SNR en un 216%.
- FFT con crecimiento de un bit cada dos etapas seguida de IFFT con entrada BR: al tener 10 etapas, el ancho de palabra del resultado de la FFT se incrementa 5 bits. En este caso, la amplitud de salida de la FFT es $A \cdot N/2^5$ donde A es la amplitud de entrada, mientras que la de la IFFT es $A/2^5$. El resultado es un SNR de 54 dB, casi el mismo valor que se obtenía solo con la FFT, un incremento del 116% respecto al circuito FFT-IFFT sin crecimiento de bits.

A partir de los informes de utilización al sintetizar los circuitos con Vivado, se ha estimado el área de los dos circuitos propuestos y del circuito FFT-IFFT original que limita el crecimiento de bits en cada etapa. Los resultados se muestran en la Tabla 4.3.

Bloque	FFT-IFFT sin crecimiento de bits		FFT-IFFT con crecimiento de 10 bits		FFT-IFFT con crecimiento de 5 bits		Bloques disponibles
	Usados	Porcentaje	Usados	Porcentaje	Usados	Porcentaje	
LUT	6539	10.31%	3230	5.09%	4455	7.03%	63400
LUTRAM	613	3.23%	210	1.11%	514	2.71%	19000
FF	5587	4.41%	3702	2.92%	3985	3.14%	126800
DSP	54	22.5%	54	22.5%	54	22.5%	240
BRAM	13.5	10%	9.5	7.04%	12.5	9.26%	135
Porcentaje total usado	3.4381%		6.1107%		4.3042%		

Tabla 4.3. Comparación de área entre FFT en serie con IFFT con y sin crecimiento de bits.

En vista de estos resultados, se ha optado por el circuito FFT-IFFT con crecimiento de 5 bits, ya que el aumento en precisión que proporciona el crecimiento de 10 bits es desproporcionado para la aplicación a la que se destina. Con un crecimiento de 5 bits se obtiene una buena precisión, con un decremento de área del 29.56% respecto al circuito con crecimiento de 10 bits y un incremento del 25.19% respecto al circuito FFT-IFFT con entrada BR original.

4.6.MÓDULO DE REORDENACIÓN EN BR

Tal y como se explicó en la sección 3.7, *FFT e IFFT*, existen dos opciones de diseño para la IFFT, teniendo en cuenta que la salida de la FFT está en BR: la primera es utilizar módulos que reordenen las salidas de la FFT e IFFT para pasar de BR a orden natural, y la segunda es adaptar el diseño de la IFFT para que acepte como entrada la señal en BR. En esta sección se han diseñado los módulos de reordenación BR necesarios para implementar la primera opción y se ha analizado el impacto en área y latencia que tendrían sobre el circuito.

Para implementar esta opción son necesarios 2 módulos de reordenación: uno para la salida de la FFT y otro para la salida de la IFFT, ambos con la misma arquitectura, pero con diferentes anchos de palabra: 26 bits tras la FFT y 16 bits tras la IFFT. Cada uno de estos módulos contiene 2 buffers de 1024 posiciones, de forma que uno de ellos almacena los datos que están entrando y el otro almacena los últimos 1024 datos que entraron, del cual también se extraen los datos en BR.

Este proceso es controlado mediante una máquina de Mealy de dos estados. En el primero, con cada dato de entrada se incrementa la dirección de escritura del buffer en el que se están almacenando los datos hasta completar las 1024 posiciones. A partir de esa dirección de escritura, se invierten los bits para obtener la dirección de lectura del buffer que contiene los 1024 datos anteriores. De este modo, se obtienen a la salida los datos en BR de la última FFT a la vez que se llena el buffer con los datos de la FFT actual. En el segundo estado, el buffer que anteriormente se utilizaba para escribir ahora se emplea para leer, y viceversa. En la Figura 4.13 se muestra un diagrama más detallado de la máquina de estados utilizada.

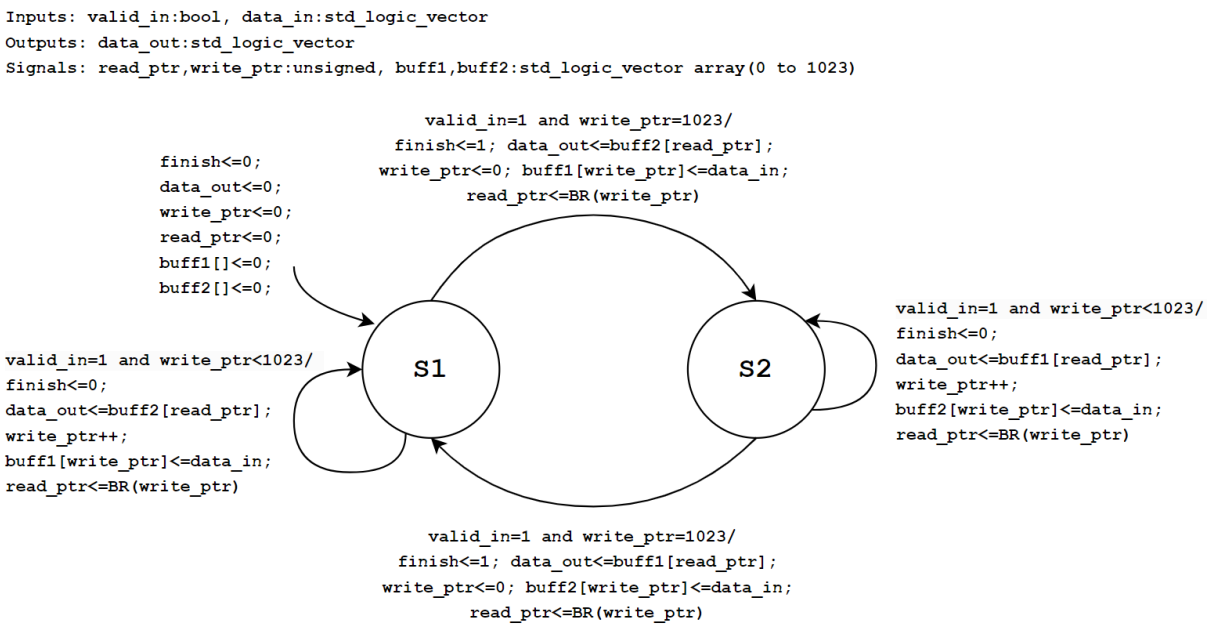

Figura 4.13. Máquina de estados del módulo de reordenación BR.

Ya que este módulo almacena los 1024 datos de la FFT en un buffer para luego reordenarlos, la latencia es de 1025 ciclos de reloj, y al utilizar dos, la latencia global del circuito se incrementa en 2050 ciclos de reloj. Teniendo en cuenta que la latencia de la FFT en serie con la IFFT es de $1087 \cdot 2 = 2174$ ciclos, añadir los dos módulos de BR casi duplica la latencia del circuito FFT-IFFT.

En cuanto al área adicional que suponen estos módulos, en la Tabla 4.4 se muestran los resultados de utilización de la síntesis de Vivado. Se observa que el área de los 2 módulos supone aproximadamente un tercio del área del circuito FFT-IFFT con entrada BR sin cuantificación.

Bloque	BR con WL=26 bits		BR con WL=16 bits		Bloques disponibles
	Usados	Porcentaje	Usados	Porcentaje	
LUT	1431	2.26%	893	1.41%	63400
LUTRAM	1152	6.06%	704	3.71%	19000
FF	122	0.10%	86	0.07%	126800
DSP	0	0%	0	0%	240
BRAM	0	0%	0	0%	135
Porcentaje total usado	1.2907%		0.8031%		Suma de ambos: 2.0937%

Tabla 4.4. Área de los módulos de reordenación BR con WL=26 y WL=16.

Estos resultados confirman el ahorro en área y latencia obtenido mediante la modificación de la IFFT para que procese la secuencia de entradas en BR.

4.7. VALIDACIÓN DEL ECUALIZADOR

Para comprobar el correcto funcionamiento del ecualizador, se ha realizado una simulación con una señal de entrada aleatoria y distintos valores para los interruptores. La Figura 4.14 muestra los resultados de la simulación con los interruptores 6, 10 y 14 a 0, donde se observa que, efectivamente, se han anulado las bandas de frecuencia correspondientes de forma simétrica.

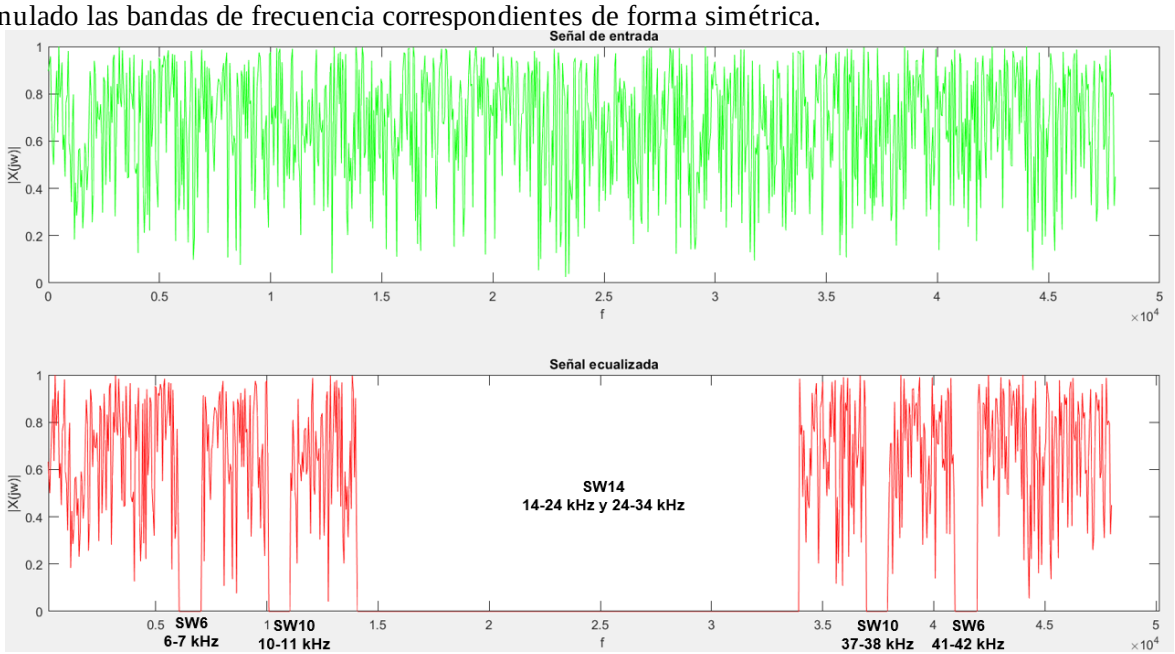

Figura 4.14. Simulación del ecualizador con SW6, SW10 y SW14 a 0.

4.8. ANÁLISIS DEL SISTEMA COMPLETO

En esta sección se ha simulado el sistema completo en condiciones normales para analizar y validar el comportamiento de las señales que componen los distintos módulos del circuito. Para este análisis, se ha generado mediante un *testbench* la siguiente señal:

$$data_{in}(t) = 32767 \cdot [0.5 \cdot \sin(2\pi t \cdot 10000) + 0.5 \cdot \sin(2\pi t \cdot 1000)] \quad (42)$$

Se ha escogido esta señal porque, al estar compuesta por dos senoidales de distinta frecuencia: una de 1 kHz y otra de 10 kHz, es sencilla de interpretar en el dominio de la frecuencia y nos permite comprobar fácilmente el funcionamiento del ecualizador eliminando uno de los senos. Como esta señal es generada mediante una variable real, se redondea su valor para convertirla a complemento a 2. Una vez generada, se modula a PDM mediante un Sigma Delta de primer orden para simular la entrada del micrófono.

Diezmador

En la simulación, el primer paso es resetear el sistema para comprobar que las salidas de todos los módulos son nulas y que se reinician todos los contadores y máquinas de estados. Una vez reseteado, el filtro diezmador comienza a recibir datos y procesarlos, lo cual se muestra en la Figura 4.15.

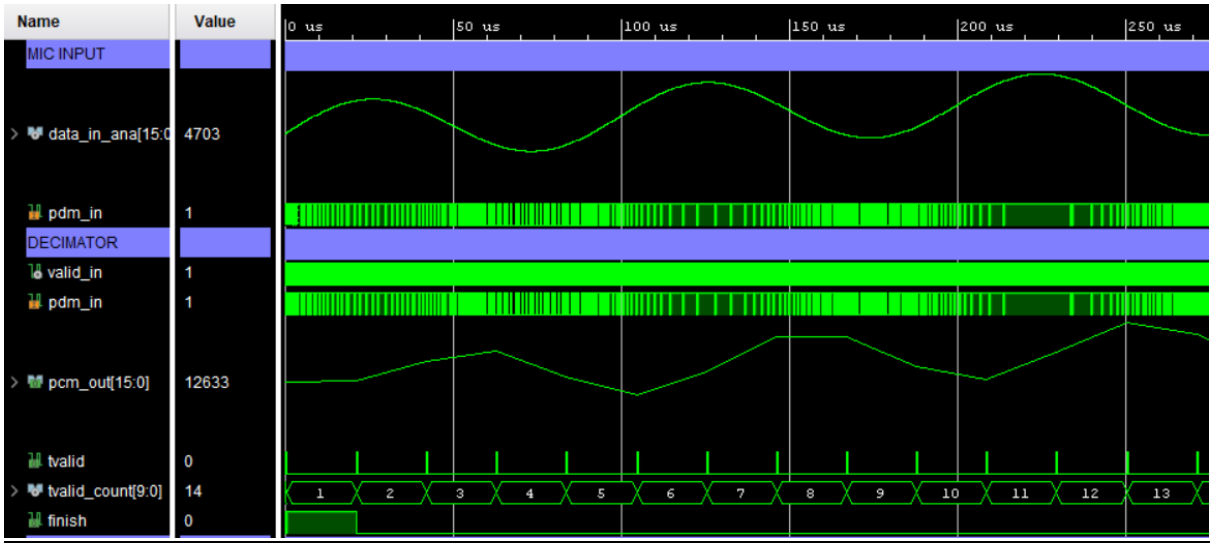

Figura 4.15. Simulación de las señales del entrada de micrófono y filtro diezmador.

Se observa que cada vez que hay un pulso en la señal *valid_in* entra un nuevo dato al filtro y que la señal *tvalid* genera un pulso positivo cada vez que un nuevo dato es procesad. El número de pulsos es contado por la señal *tvalid_count* y cada vez que su valor es 1 se activa la señal *finish*, que sirve como señal de inicio *start* de la FFT. Por otro lado, la señal *tvalid*, de 48 kHz generada por el CIC Compiler diezmador, sirve de reloj de los módulos de FFT, IFFT y ecualización. Finalmente, se ha comprobado que la amplitud máxima de la señal diezmada es de15746, dentro del rango calculado en la sección 3.3, *Decodificación de PCM y codificación a PDM*.

FFT

Cuando se activa la señal *start* del módulo de FFT, los datos recibidos son almacenados en los buffers de la mariposas y procesados como se explicó en la sección 3.7, *FFT e IFFT*, y tras 1087 ciclos se obtiene el primer dato de salida, indicado mediante la señal de finalización *finish*, que funciona como señal de inicio *start* del ecualizador. Estas señales se observan en la Figura 4.16.

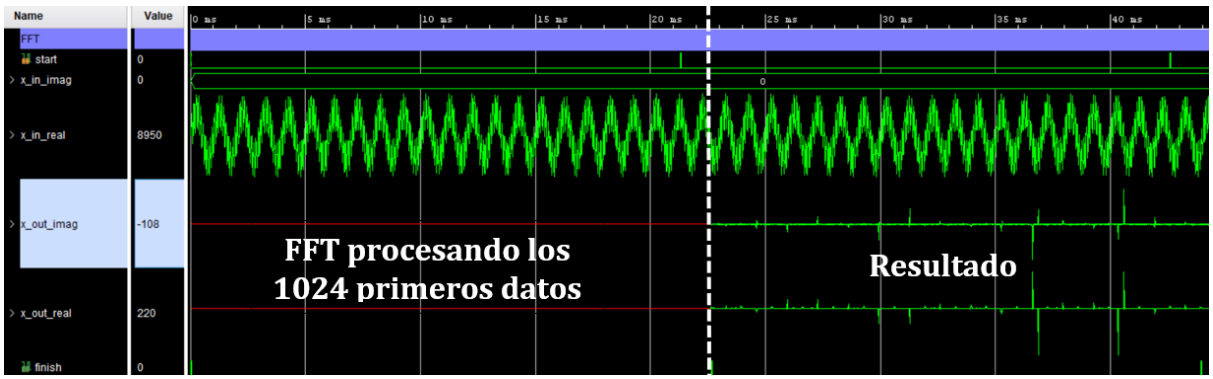

Figura 4.16. Simulación de las señales del entrada y salida de la FFT.

Ecualizador

Al activarse la señal *start* del ecualizador, su contador *mod_1024* se reinicia y comienza a incrementarse con cada nuevo dato de entrada. Si el valor del contador coincide con que el valor de su interruptor asociado es 0, la salida del ecualizador será 0. En este caso, se ha fijado a 0 el valor de los interruptores

9 y 10 (*SW9* y *SW10)*, al que le corresponden los rangos de frecuencias que van de 9 a 11 kHz y de 37 a 39 kHz. Se ha elegido este interruptor para eliminar la componente senoidal de 10 kHz.
En la Figura 4.17, se observa que el circuito solo tiene 1 ciclo de latencia.

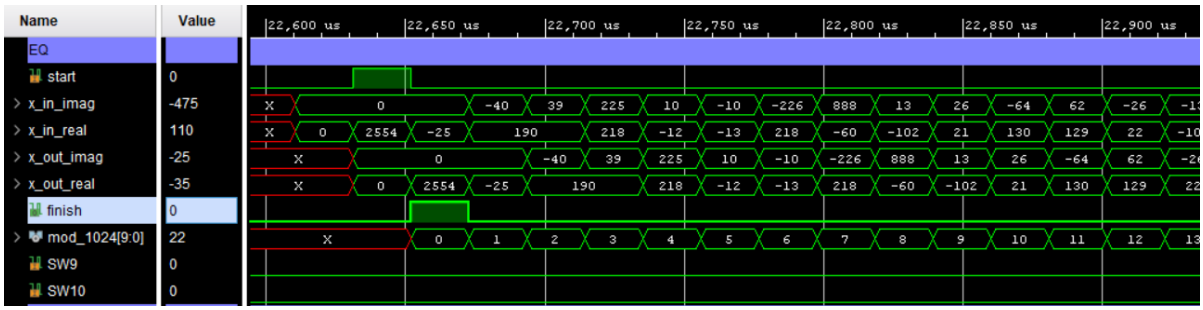

Figura 4.17. Simulación de las señales del ecualizador.

IFFT

La señal resultante tras la ecualización es introducida en el módulo IFFT. Sus señales y latencia son las mismas que las de la FFT. En la Figura 4.18 se nota a simple vista como la componente real de la salida tiene forma senoidal, ya que se ha eliminado la componente de10 kHz mediante el ecualizador. Además, se observa que la componente imaginaria de la señal no es nula como se espera teóricamente, aunque sus valores son despreciables, siendo -6 su valor máximo. Esto se debe al error de cuantización del circuito.
Como se explicó en la sección 3.9, *Interfaces entre módulos*, mediante la señal *finish* se genera la señal *IFFT_ready*, que indica que se ha realizado la IFFT de los primeros 1024 datos, sirviendo como *reset* activo a nivel bajo del interpolador.

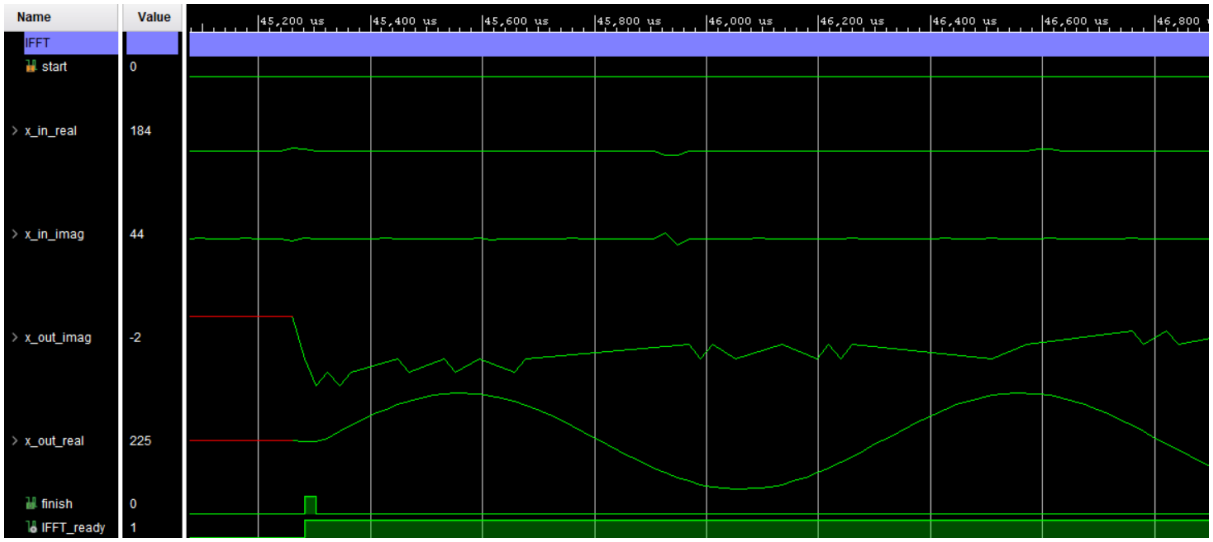

Figura 4.18. Simulación de las señales de la IFFT.

Interpolador

Una vez que la señal *IFFT_ready* comienza a valer 1, el filtro interpolador recibe y procesa los datos de la IFFT, de la cual se ha descartado su salida imaginaria puesto que la señal de audio ha de ser puramente real. Al igual que el diezmador, la señal *tvalid* genera un pulso positivo cada vez que un nuevo dato es procesado por el filtro. El número de pulsos es contado por la señal *tvalid_count* y cada vez que su valor sea 1 se activa la señal *finish*, que sirve como señal de inicio *start* de la FFT. Si se compara con la figura diezmador, se nota como la señal se ha convertido en una senoidal de 10 kHz como consecuencia del ecualizador. Además, tanto la amplitud de señal de entrada como de salida del interpolador están comprendidas dentro del rango de valores calculado en la sección 3.5, *Interpolación*, siendo 8672 en la entrada y 11406 en la salida.

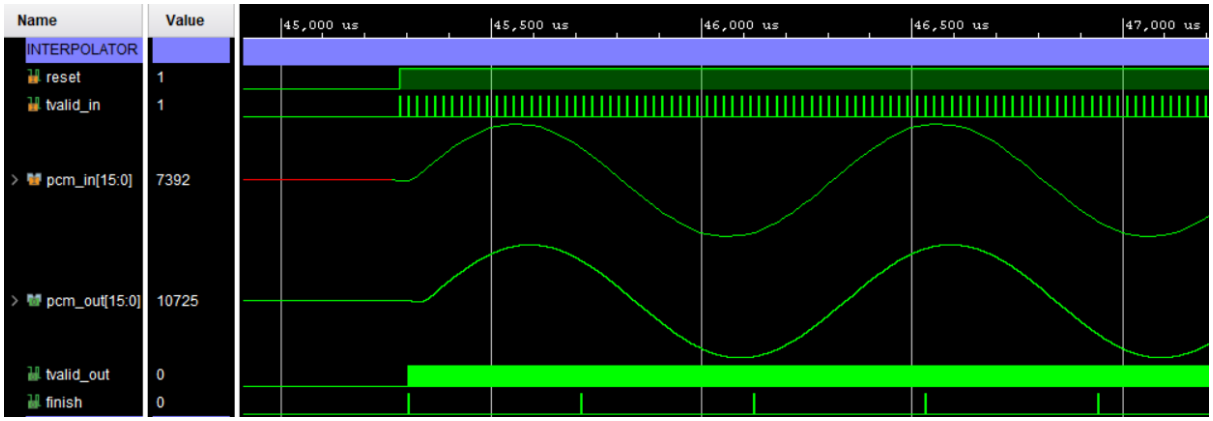

Figura 4.19. Simulación de las señales del interpolador.

Modulador Sigma-Delta

Una vez interpolada a 2.5 MHz, la señal es modulada a PDM por el Sigma-Delta. En la Figura 4.20 se muestran los resultados de esta modulación. Se aprecia como la densidad de pulsos es aumenta y disminuye conforme lo hace la amplitud de la señal de entrada.

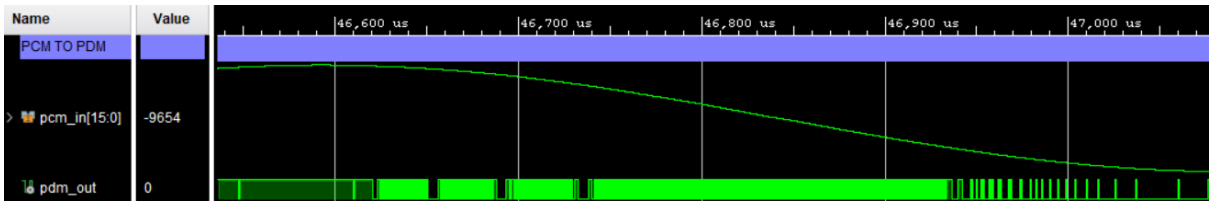

Figura 4.20. Modulación PDM de la señal interpolada.

4.9.PRUEBA DEL SISTEMA EN PLACA

Finalmente, una vez validados y corregidos todos los módulos, se han realizado varias pruebas con el sistema completo en la placa. Estas son necesarias para detectar posibles fallos no vistos en la simulación y para valorar la comodidad de la interfaz hombre-máquina.

En primer lugar, se ha comprobado que el nivel de ruido introducido por el procesamiento y cuantificación de la señal es aceptable. El volumen de este ruido es bajo, aunque lo suficientemente alto para escucharse cuando la señal captada por el micrófono es baja. A pesar de esto, es un nivel de ruido aceptable que pierde presencia cuando la señal grabada por el micrófono tiene cierta potencia.

Posteriormente, para comprobar las bandas de frecuencia, se han generado señales senoidales de distintas frecuencias mediante un ordenador, cuya salida de audio se ha aproximado al micrófono de la placa. Si se escucha el audio ecualizado mediante unos auriculares que aíslen el sonido exterior, se comprueba que la señales dejan de escucharse si se anula la banda de frecuencia en la que están contenidas.

Finalmente, se ha comprobado la señal de reset, controlada mediante el interruptor 15. Si este interruptor vale 0 el sistema deja de funcionar y no hay salida de audio, mientras que si vale 1 el sistema vuelve a funcionar correctamente.

4.10. LATENCIA DEL SISTEMA

Generalmente, la latencia es una de las características más relevantes en un sistema de grabación y procesamiento de audio en “tiempo real”, por lo que es esencial determinarla. Esta latencia se puede medir fácilmente en la simulación como la diferencia entre el instante en el que el filtro diezmador recibe el primer dato y el instante en el que se genera el primer dato de salida del modulador Sigma-Delta. El resultado obtenido es 45.3 ms, una latencia alta en comparación con las latencias típicas de una interfaz de audio, que suelen estar entre los 5 y 20 ms.

No obstante, es normal que este caso la latencia sea mayor, ya que este sistema no solo graba, sino que incluye la ecualización de audio. Además, el procesamiento de audio en el dominio de la frecuencia incrementa significativamente la latencia del sistema, puesto que cada FFT tiene una latencia de 1087 ciclos de 48 kHz (22.64 ms). Esta latencia aumenta en función del número de datos que se procesa por cada FFT, en este caso 1024, el cual es elevado si se tiene en cuenta que este valor suele estar entre 32 y 4096 [25], utilizándose solo potencias de dos. La ventaja de utilizar un número de puntos elevado es que proporciona más resolución para ecualizar frecuencias.

A raíz de estas consideraciones se ha observado una grave incongruencia en el diseño del sistema que ha de ser corregida en futuras versiones. El problema es el elevado número de puntos utilizados en la FFT, que, aunque proporcionan una muy buena resolución en el espectro de frecuencias para ecualizar la señal, no se aprovecha ya que se ha dividido este espectro en bandas de 1 kHz, debido al diseño de la interfaz hombre-máquina del ecualizador. Para solucionar este problema, hay varias opciones:

- Reducir considerablemente el número de puntos de la FFT y mantener el mismo diseño de ecualizador dividido en bandas de 1 kHz.
- Mantener el número de puntos de la FFT, por lo que la latencia seguiría siendo alta, pero diseñar un nuevo ecualizador basado en otra interfaz que permitiese una ecualización más precisa, en la que se pueda alterar la amplitud de cada frecuencia por separado usando los botones de la placa.
- Diseñar un sistema que mezcle las dos opciones anteriores, reduciendo la latencia a la vez que se aprovecha mejor la resolución del sistema con una interfaz cómoda pero capaz de ofrecer más posibilidades.

De las 3 opciones se ha optado por la tercera como línea de trabajo para futuras continuaciones del proyecto ya que es la más completa y la que puede ofrecer resultados que aprovechen plenamente la capacidad de procesamiento del circuito. En concreto, se ha decidido reducir el número de puntos de la FFT a 256, de forma que la latencia generada por las mariposas de la FFT ahora se obtiene como:

$$Lat_{marip} = \sum_{i=1}^{10}(L_i + 1) = 129 + 65 + 33 + 17 + 9 + 5 + 3 + 2 = 263\ ciclos \quad (43)$$

La latencia producida por los rotadores es:

$$Lat_{rot} = 6 \cdot 7 = 42\ ciclos \quad (44)$$

En total, la latencia de la FFT de 256 puntos es:

$$Lat_{FFT\ 256} = Lat_{marip} + Lat_{rot} = 305\ ciclos \quad (45)$$

Esto supone una gran diferencia respecto a la latencia de la FFT con 1024 puntos, que tenía 1087 ciclos de latencia. Como la latencia de la IFFT es la misma que la de la FFT, con esta modificación la latencia total del circuito FFT-IFFT es:

$$Lat_{FFT-IFFT\ 256} = 2Lat_{FFT\ 256} = 610\ ciclos \quad (46)$$

Con 1024 puntos la latencia del circuito FFT-IFFT, se calcula como:

$$Lat_{FFT-IFFT\ 1024} = 2Lat_{FFT\ 1024} = 2 \cdot 1087 = 2174\ ciclos \quad (47)$$

Entonces, con una FFT-IFFT de 256 puntos hay una reducción de $2174 - 610 = 1564\ ciclos$. Como la frecuencia de reloj de la FFT es de 48 kHz, la reducción será de 32.58 ms, por lo que la latencia global del sistema pasará de 45.3 ms a 12.72 ms, un valor de latencia relativamente bajo, que estaría dentro de los valores esperados para una interfaz de sonido. Esta reducción de latencia también conlleva

una reducción en la resolución en frecuencia, haciendo que el mínimo escalón en frecuencia pase de 46.92 Hz a 188.23 Hz, como se muestra en la ecuación (48).

$$\frac{f_{s\ FFT}}{N-1} \cong \frac{48\ kHz}{255} = 188.23\ Hz \qquad (48)$$

4.11. IMPACTO EN EL ÁREA Y POTENCIA

A lo largo del capítulo 3, se han explicado las decisiones de diseño tomadas para reducir el área y, consecuentemente, disminuir el consumo de potencia del circuito, mientras que en este capítulo se ha concretado el impacto de estas decisiones por separado. En esta sección, se ha estimado el impacto global que estas consideraciones han tenido sobre el sistema en su conjunto.

Para ello, se ha implementado un sistema formado por los módulos que han sido descartados: filtros FIR para el diezmado e interpolación, FFT con crecimiento de 1 bit cada etapa, IFFT sin entrada en BR y dos módulos de reordenación BR. Se ha comparado el área de este circuito con el circuito optimizado, obteniéndose los resultados mostrados en la Tabla 4.5.

Bloque	Sistema optimizado		Sistema sin optimizar		Bloques disponibles
	Usados	Porcentaje	Usados	Porcentaje	
LUT	4630	7.30%	12019	18.96%	63400
LUTRAM	550	2.89%	5128	26.99%	19000
FF	4496	3.55%	6444	5.08%	126800
DSP	58	24.17%	58	24.17%	240
BRAM	12.5	9.26%	13.5	10%	135
Porcentaje total usado	4.65%		11.291%		

Tabla 4.5. Comparación de área entre el sistema optimizado y sin optimizar.

Mediante los informes de implementación de Vivado, se ha estimado que, gracias a las optimizaciones, se ha logrado una reducción de área del 58,82% respecto al sistema sin optimizar. A pesar de que el porcentaje total utilizado no es muy elevado en ninguno de los casos, es conveniente optimizar siempre el uso de recursos en la medida de lo posible.

En cuanto a la utilización de los distintos bloques, se observa que el mayor porcentaje corresponde a los DSPs. Estos bloques son fundamentales para implementar sumadores y multiplicadores binarios de alta velocidad y bajo consumo, esenciales para los filtros CIC y las rotaciones de la FFT. En segundo lugar, destacan los bloques de memoria RAM, principalmente utilizados para almacenar los rotadores de la FFT e IFFT. Por último, se encuentran los bloques básicos de cualquier circuito digital: LUTs, que permiten implementar cualquier función lógica; LUTRAMs, que además permiten almacenar datos intermedios o temporales; y Flip-Flops, que almacenan el estado de las señales. Debido a su gran disponibilidad en la FPGA, los porcentajes de utilización no son altos.

Como se ha mencionado antes, esta reducción de área conlleva una reducción de potencia, que ha sido estimada comparando los informes de potencia de ambos circuitos. Los resultados muestran un consumo de 0.383 W para el sistema sin optimizar y 0.296 W para el optimizado, una reducción del 22.71%. Como se observa en la Figura 4.21Figura 4.6, el consumo de este circuito es predominantemente dinámico, con el mismo consumo estático en ambos. En la Tabla 4.2 se muestra el reparto de consumo dinámico entre los distintos bloques del circuito, donde se observa que la mayor parte del consumo es debido a las señales, seguidas de los DSPs y la lógica, mientras que los relojes y los bloques RAM presentan un consumo significativamente menor.

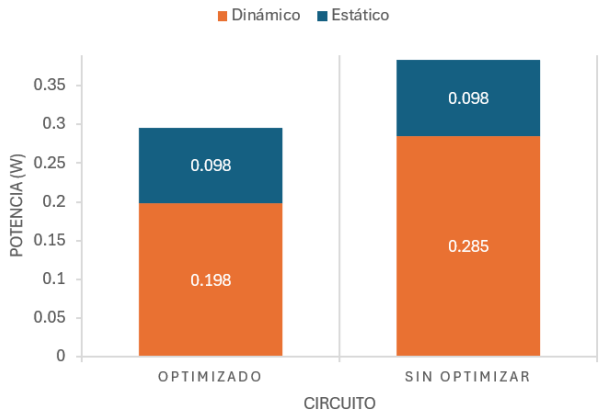

Figura 4.21. Comparación de consumo estático y dinámico entre el circuito optimizado y sin optimizar.

Bloque	Optimizado		Sin optimizar	
	Potencia (W)	Porcentaje	Potencia (W)	Porcentaje
Relojes	0.003	1%	0.004	1%
Señales	0.081	41%	0.128	45
Lógica	0.052	26%	0.093	33%
DSP	0.061	31%	0.060	21%
BRAM	0.001	<1%	0.001	<1%

Tabla 4.6. Comparación de consumo dinámico entre el circuito optimizado y sin optimizar.

5. CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se resumen las conclusiones principales del proyecto y se proponen posibles direcciones de trabajo para continuaciones del trabajo. Se evalúan los resultados obtenidos en relación con los objetivos iniciales y se identifican áreas para mejorar y expandir el trabajo realizado.

CONCLUSIONES

Los resultados del trabajo han sido evaluados en base a los objetivos y especificaciones establecidos en la sección 1.2, *Objetivos y especificaciones*, que pueden resumirse en la implementación de un ecualizador basado en FFTs optimizando el área y consumo, con una interfaz hombre-máquina intuitiva y cómoda, validando progresivamente los módulos diseñados antes de comenzar el diseño de los siguientes. A partir de este análisis, se han extraído las siguientes conclusiones:

- Se ha conseguido implementar el sistema completo y, tanto en las simulaciones como en la implementación en placa, se ha comprobado que el ecualizador funciona correctamente, permitiendo al usuario escuchar el resultado del procesamiento con una latencia lo suficientemente baja como para dar la sensación de que procesamiento en tiempo real. Sin embargo, esta latencia es aun considerablemente mayor que las de interfaces de audio actuales, lo que indica la necesidad de futuras mejoras.
- Se ha logrado una interfaz hombre-máquina cómoda e intuitiva que permite al usuario modificar en tiempo real las bandas de frecuencia a ecualizar. No obstante, existe margen para futuras mejoras que posibiliten una ecualización más compleja y precisa.
- Mediante las decisiones de diseño tomadas se ha reducido el área en un 58.82% y la potencia en un 22.71% respecto a opciones con mayor precisión, manteniendo una calidad y precisión de conversión adecuadas.
- El tiempo requerido para diferentes partes del proyecto no ha coincidido con lo planificado inicialmente. En particular, el tiempo dedicado al diseño ha sido menor del esperado, mientras que el estudio de métodos de procesamiento y modulación, así como las tareas de depuración y validación, han requerido más tiempo del esperado. Sin embargo, este tiempo ha adicional se ha traducido en un mejoras y optimizaciones muy significativas en la calidad del prototipo dentro de los plazos establecidos para la finalización del proyecto. El porcentaje de tiempo dedicado a la depuración y verificación se corresponde con la mayoría de las estimaciones realizadas para sistemas basados en FPGAs, que indican que supone entre el 50 y 70% del tiempo disponible [26] [27].

En conclusión, se ha logrado desarrollar un primer prototipo funcional del sistema, que, a pesar de requerir mejoras en la latencia y un rediseño de la interfaz hombre-máquina, cumple con los objetivos y especificaciones iniciales dentro del plazo de tiempo previsto.

LÍNEAS FUTURAS

A lo largo de este trabajo se han identificado varios errores de diseño que deberán ser corregidos en versiones futuras. Además, se han detectado áreas que, aunque funcionales, podrían beneficiarse de mejoras en sus prestaciones. A continuación, se detallan algunas tareas y líneas de trabajo propuestas para futuras continuaciones del proyecto.

Como se ha explicado en la sección 4.10, *Latencia del sistema*, la latencia del sistema es demasiado elevada, siendo el elevado número de puntos de la FFT la razón principal. Esto, combinado con una interfaz hombre-máquina sobre simplificada, conlleva un desaprovecho de la resolución de la FFT. Para solucionar este problema, se han barajado varias opciones, de las cuales se ha optado por reducir el número de puntos de la FFT y rediseñar el ecualizador y su interfaz hombre-máquina, haciendo uso de los botones disponibles en la placa. Concretamente, el número de puntos de la FFT se reducirá a 256, de forma que la latencia pasará de 45.3 ms a 12.72 ms, aunque esto conlleve una pérdida de resolución en frecuencia para el ecualizador, cuyo escalón mínimo de frecuencia pasará de 46.92 Hz a 188.23 Hz. Además de la selección de cada banda mediante los interruptores, se empleará el botón superior e

inferior para seleccionar la amplitud de cada banda, y el izquierdo y derecho para seleccionar el escalón de frecuencia deseado dentro de la banda de frecuencia marcada con el interruptor. La frecuencia concreta que se esté modificando se mostrará en los 4 displays de 7 segmentos que también contiene la placa.

Por otro lado, debido a las limitaciones de tiempo a las que está sujeto este trabajo, se han implementado una FFT e IFFT radix-2 DIF, ya que permite generar los coeficientes de manera más sencilla. Sin embargo, desde el punto de vista de las prestaciones, esta no es la opción óptima [28]. Una línea de mejora futura podría consistir en reemplazar la FFT e IFFT actuales por versiones de radix-2^2, que presentan un menor número de operaciones triviales y mejorarían significativamente el rendimiento del sistema. En este algoritmo, las etapas impares solo contienen rotaciones triviales, permitiendo simplificar los rotadores y reducir el área del circuito.

Finalmente, otra línea de trabajo futuro sería el diseño de una PCB dedicada que contenga únicamente la FPGA, el micrófono, la salida de audio y los elementos necesarios para la interfaz hombre-máquina. Esto permitiría una integración más compacta y específica, optimizando el rendimiento general del dispositivo.

6. REFERENCIAS Y BIBLIOGRAFÍA

[1] J. Bellamy, Digital Telephony, Tercera ed., Wiley-Interscience, 1941, p. 56.

[2] U. Zölzer, Digital Audio Signal Processing, Segunda ed., Hamburgo: Wiley, 2008, pp. 10-11.

[3] G. Slade, «The Fast Fourier Transform in Hardware: A Tutorial Based on an FPGA,» p. 1, 2013.

[4] Xilinx, «XCELL,» *The Quarterly Journal For Programmable Logic User,* nº 32, pp. 4-5, 1999.

[5] Polaris Market Research, «Field Programmable Gate Array (FPGA) Market Projected to Grow by USD 28.81 Billion, at 10.5% CAGR During Period 2024 - 2032: Polaris Market Research,» 23 Enero 2024. [En línea]. Available: https://www.globenewswire.com/news-release/2024/01/23/2813785/0/en/Field-Programmable-Gate-Array-FPGA-Market-Projected-to-Grow-by-USD-28-81-Billion-at-10-5-CAGR-During-Period-2024-2032-Polaris-Market-Research.html. [Último acceso: 10 Abril 2024].

[6] S. D. Brown, R. J. Francis, J. Rose y Z. G. Vranesic, «Introduction to FPGAs,» de *Field-Programmable Gate Arrays*, Boston, Springer Science, 1992, pp. 1-2.

[7] S. Mittal, S. Gupta y S. Dasgupta, «System Generator: The State-of-art FPGA Design tool for DSP Applications,» *Third International Innovative Conference On Embedded Systems, Mobile Communication And Computing,* vol. ICEMC2 2008, pp. 187-190, 2008.

[8] M. Garrido, «A Survey on Pipelined FFT Hardware Architectures,» *Journal of Signal Processing Systems,* nº 94, pp. 1345-1364, 2021.

[9] J. W. Cooley y J. W. Tukey, «An algorithm for the machine calculation of complex Fourier series,» *Mathematics of Computation,* vol. 19, nº 90, pp. 297-301, 1965.

[10] Xilinx, «Nexys A7 Reference Manual,» 10 Julio 2019. [En línea]. Available: https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual. [Último acceso: 10 Abril 2024].

[11] B. P., «TechTerms.com,» 26 octubre 2022. [En línea]. Available: https://techterms.com/definition/pcm. [Último acceso: 2024].

[12] R. Schreier y G. C. Temes, Understanding Delta-Sigma converters, Pearson, 2005.

[13] M. Martínez, L. Gómez, A. J. Serrano y J. G. J. Vila, «OCW-UV,» 2009. [En línea]. Available: http://ocw.uv.es/ingenieria-y-arquitectura/filtros-digitales/tema_7_modificacion_de_la_frecuencia_de_muestreo.pdf. [Último acceso: 11 Abril 2024].

[14] S. K. Mitra, Digital Signal Processing: A Computer-Based Approach, Segunda ed., McGraw-Hill Higher Education, 2001.

[15] Ó. F. Corredor, L. F. Pedraza y C. A. Hernández, Diseño e implementación de filtros digitales, vol. 3, Ricardo Piraján Cantillo, 2009.

[16] A. Chandra y S. Chattopadhyay, «Design of hardware efficient FIR filter: A review of the state-of-the-art,» *Engineering Science and Technology, an International Journal,* nº 19, pp. 212-226, 2015.

[17] Xilinx, «Xilinx,» 26 Octubre 2022. [En línea]. Available: https://www.xilinx.com/support/documents/ip_documentation/fir_compiler/v7_2/pg149-fir-compiler.pdf. [Último acceso: 11 Abril 2024].

[18] D. R. Brown, «spinlab.wpi.edu,» 2014. [En línea]. Available: https://spinlab.wpi.edu/courses/ece503_2014/4-5polyphase_filters.pdf. [Último acceso: 11 Abril 2024].

[19] E. B. Hogenauer, «An Economical Class of Digital Filters for Decimation,» *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vols. %1 de %2ASSP-29, nº 2, pp. 155-162, 1981.

[20] P. Dillien, «And the Winner of Best FPGA of 2016 is...,» *EETimes,* 6 marzo 2016.

[21] A. V. Oppenheim y R. W. Schafer, Discrete-Time Signal Processing, Pearson India Education, 1988, p. 662.

[22] Diligent, «GitHub,» 7 febrero 2019. [En línea]. Available: https://github.com/Digilent/Nexys-A7-100T-DMA-Audio/tree/master. [Último acceso: 2024].

[23] AMD, «Technical Information Portal,» 13 3 2024. [En línea]. Available: https://docs.amd.com/v/u/en-US/ug475_7Series_Pkg_Pinout. [Último acceso: 2024].

[24] Audient, «Audient,» 2016. [En línea]. Available: https://d9w4fhj63j193.cloudfront.net/2021/iD4MKII/English/iD4%20MKII%20Manual%20V1.3.pdf. [Último acceso: 2024].

[25] Microsemi, «Microchip,» Marzo 2014. [En línea]. Available: https://ww1.microchip.com/downloads/aemdocuments/documents/fpga/ProductDocuments/SupportingCollateral/in_circuit_fpga_debug_challenges_solutions.pdf. [Último acceso: 2023].

[26] Promwad, «Promwad Electronics Design,» 22 Julio 2021. [En línea]. Available: https://promwad.com/news/art-fpga-debugging-how-speed-it-smart-design-testing-tricks. [Último acceso: 2024].

[27] M. Garrido, «A New Representation of FFT Algorithms,» *IEEE Transactions on Circuits and Systems Part 1*, nº 63, pp. 1737-1745, 2016.

[28] T. Kite, «Understanding PDM Digital Audio,» 2012. [En línea]. Available: https://users.ece.utexas.edu/~bevans/courses/rtdsp/lectures/10_Data_Conversion/AP_Understanding_PDM_Digital_Audio.pdf. [Último acceso: 11 Abril 2024].

[29] R. Randall y M. Tordon, Encyclopedia of Vibration, 2001.

[30] R. Lyons, Understanding Digital Signal Processing, Upper Saddle River, New Jersey: Prentice Hall, 2004, p. 550–566.

[31] J. López, «Metodología de diseño de sistemas digitales: Representación, Cuantificación y Síntesis,» Madrid, 2024.

ANEXO A: ASPECTOS ÉTICOS, ECONÓMICOS, SOCIALES Y AMBIENTALES

A.1 INTRODUCCIÓN

Este Trabajo de Fin de Máster se centra en desarrollar un sistema de procesamiento de audio utilizando una FPGA Artix-7 en la placa Nexys A7, con el objetivo de implementar un ecualizador de baja latencia. Este sistema permite filtrar las diferentes frecuencias de una señal de audio capturada por un micrófono y reproducirla con una percepción de procesamiento en tiempo real.

Al iniciar este proyecto, se notó que existen muy pocos ejemplos detallados que usen de forma eficiente el micrófono y la salida de audio de la Nexys A7, y ninguno que procese ese audio en tiempo real mediante FFTs. Este es uno de los motivos que han impulsado la realización de este trabajo, que puede servir como introducción a los temas de procesamiento de audio e implementación de arquitecturas digitales de FFT desde un punto de vista práctico para nuevos desarrolladores e investigadores. Este proyecto, al facilitar la introducción de nuevos desarrolladores en este campo, puede contribuir indirectamente a nuevos avances en la experiencia auditiva en producciones musicales y eventos en vivo.

Además, dado el creciente problema de la crisis medioambiental y la creciente escasez de recursos y energía, el proyecto ha tomado medidas conscientes para optimizar el uso del hardware disponible en la placa Nexys A7. Esta preocupación por la sostenibilidad ambiental se ha integrado en todas las etapas del desarrollo del sistema de procesamiento de audio, con el objetivo de minimizar el impacto ecológico y maximizar la eficiencia energética.

A.2 DESCRIPCIÓN DE IMPACTOS RELEVANTES

Síntesis del trabajo realizado en la fase 2, de selección y descripción de impactos. Presentar y justificar las conclusiones a las que se haya llegado sobre cuáles son los asuntos más relevantes relacionados con la sostenibilidad social, económica o ambiental, así como los principales grupos de interés identificados y que se han considerado en los análisis posteriores.

El proyecto tiene cierto impacto en el ámbito social y ético, puesto que aporta a los interesados y nuevos desarrolladores en la materia los conocimientos y medios adquiridos a lo largo de un curso para comprender y mejorar las técnicas de procesamiento de audio y los algoritmos de FFT, que pueden resultar en avances en la experiencia auditiva en producciones musicales y eventos en vivo. Económicamente, optimizar el procesamiento de audio puede reducir costos operativos y ofrecer ventajas competitivas en la industria del audio. Además, la utilización eficiente de hardware contribuye a objetivos ambientales de sostenibilidad al minimizar el consumo energético del sistema.

Se ha llevado a cabo una evaluación exhaustiva de los impactos relacionados con la sostenibilidad social, económica y ambiental del sistema de procesamiento de audio desarrollado. Las conclusiones más relevantes son:

- Impacto social: la implementación de un ecualizador de baja latencia y su documentación facilita la introducción de nuevos desarrolladores en el tratamiento de audio digital mediante FPGAs, lo cual puede conllevar avances en la calidad del audio en producciones musicales y eventos en directo.
- Impacto económico: la optimización del procesamiento de audio permite reducir los costos operativos al disminuir la necesidad de equipos adicionales. Esto puede traducirse en una ventaja competitiva y una mayor eficiencia en estudios de grabación y eventos en directo, haciendo el proyecto económicamente viable y beneficioso para los profesionales del sector.
- Impacto Ambiental: La utilización eficiente de una FPGA contribuye a un menor consumo energético en comparación con otros métodos de procesamiento de audio. Este enfoque reduce la huella de carbono del sistema y alinea el proyecto con objetivos de sostenibilidad ambiental.

Los grupos de interés identificados son:

1. Usuarios finales: productores musicales y técnicos de sonido que se benefician directamente de la mejora en la calidad del audio.
2. Industria del audio: empresas y profesionales que pueden ver reducidos sus costos operativos gracias a una tecnología más eficiente.
3. Reguladores ambientales: entidades que velan por el cumplimiento de normativas ambientales y de eficiencia energética.
4. Desarrolladores y proveedores de tecnología: Empresas que proporcionan el hardware y software necesarios para la implementación del sistema.

Estos grupos de interés han sido considerados a lo largo del análisis para asegurar que el proyecto no solo cumpla con sus objetivos técnicos, sino que también tenga un impacto positivo y equilibrado en todos los ámbitos.

A.3 IMPACTO AMBIENTAL

A lo largo de este proyecto, una de las mayores prioridades ha sido la de comparar las distintas alternativas para encontrar un balance entre la máxima precisión y la minimización del área del circuito, estimando en todo momento el impacto concreto que tiene cada una de las decisiones sobre los recursos y la potencia del sistema. Este análisis puede servir para que otros desarrolladores opten por las opciones más sostenible y eficientes, no solo contribuyendo a un menor consumo energético en comparación con otros métodos de procesamiento de audio, sino que también ahorrando costes y adquiriendo ventaja competitiva.

A.4 CONCLUSIONES

El presente trabajo ha sido evaluado desde perspectivas ética, social, económica y medioambiental, mostrando un impacto positivo en cada área. Socialmente, proporciona una introducción al propiciando avances en el sector y mayor satisfacción de los consumidores. Económicamente, optimiza costos operativos al reducir la necesidad de equipos adicionales. Ambientalmente, se ha hecho un uso eficiente del hardware, minimizando el impacto ambiental al elegir tecnologías energéticamente eficientes. Estos criterios de sostenibilidad han añadido valor al proyecto, garantizando su integridad y eficacia mientras se minimiza su huella ambiental.

ANEXO B: PRESUPUESTO ECONÓMICO

COSTE DE MANO DE OBRA (coste directo)	Horas	Precio/hora	Total
	500	40 €	**20000 €**

COSTE DE RECURSOS MATERIALES (coste directo)	Precio de compra	Uso en meses	Amortización (en años)	Total
Ordenador personal (Software incluido)	1.300,00 €	7	5	152 €
Placa de desarrollo Nexys A7	402,08 €	7	5	47 €

COSTE TOTAL DE RECURSOS MATERIALES	**199 €**

GASTOS GENERALES (costes indirectos)	15%	sobre CD	**3030 €**
BENEFICIO INDUSTRIAL	6%	sobre CD+CI	**1382 €**

SUBTOTAL PRESUPUESTO		**24611 €**
IVA APLICABLE	21%	**5169 €**
TOTAL PRESUPUESTO		**29780 €**