Proyecto final - Filtro adaptivo

Ing. Giribuela

Julio 2024

Índice

1. Introducción

En el presente informe se desarrollan los aspectos teóricos y prácticos del proyecto final para el curso de posgrado "Diseño Digital Avanzado". Con el fin de contextualizar el proyecto y diseño del mismo, se realiza una descripción teórica del algoritmo utilizado junto con sus ventajas y desventajas.

A partir de ello es que se realiza una propuesta de arquitectura para implementar el diseño. Luego, por medio de un análisis vectorial y espectral se comprueba que el módulo funcione de manera esperada.

1.1. Propuesta - Cancelación de ruido

El proyecto final del curso consiste en el diseño e implementación de un filtro adaptivo. La idea fundamental es, por medio del algoritmo LMS, actualizar los coeficientes de un filtro FIR para reducir el ruido ambiente que contamina una señal de interés.

Considere el caso donde una señal de interés $s(n)$ es adquirida en un ambiente altamente ruidoso $\varepsilon(n)$. Hay de hecho muchas situaciones de la vida cotidiana que representan a este escenario, la voz de un piloto en la cabina de un avión, el manos libres del auto, son algunas de ellas. Asuma que en adición a la señal ruidosa, se cuenta con la posibilidad de grabar (con otro micrófono) una señal más, alejada de la señal deseada. En este caso, la otra señal $\eta(n)$ será sólo ruido. Este ruido está altamente correlacionado con el ruido $\varepsilon(n)$ presente en la señal deseada. En estas situaciones un sistema adaptivo como el de la Figura 1 será capaz de cancelar (reducir significativamente) el ruido en la señal deseada.

La señal de entrada es en este caso el ruido $\eta(n)$, mientras que la señal de referencia es $s(n)+\varepsilon(n)$. Es importante resaltar que si bien ambos ruidos provienen de la misma fuente y están altamente correlacionados, no son iguales dado que ambas señales se propagan a través de diferentes caminos, rebotan contra distintos objetos, etc. La señal deseada $s(n)$ es estadísticamente independiente del ruido. El sistema adaptivo, en un caso ideal, generará a la señal $y(n)$ tan cerca como sea posible de $s(s)+\varepsilon(n)$.

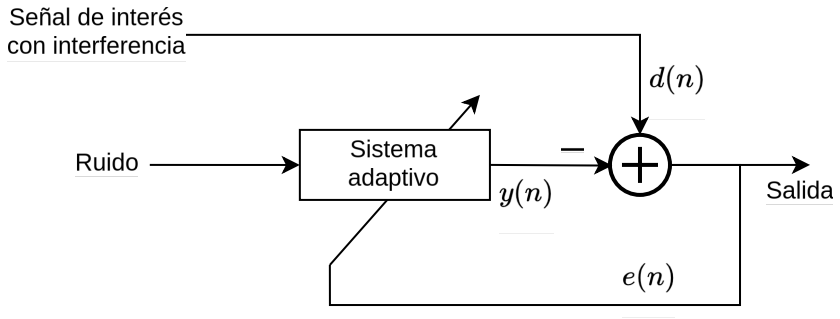


Figura 1: Diagrama en bloques del sistema

Como se aprecia en la Figura 1, el proyecto consiste en a partir de dos micrófonos, obtener muestras de dos señales, un micrófono que obtenga muestras de la señal de interés mas el ruido ambiente, y el otro micrófono que muestree sólo ruido. Luego, a partir del filtro adaptivo y un restador disminuir el ruido presente en la señal grabada con el primer micrófono.

2. Sistemas adaptivos

En lo sistemas de procesamiento de señal clásicos, sus propiedades de diseño son definidas. Sus parámetros son invariantes en el tiempo. Los sistemas adaptivos cambian sus parámetros o forma con el fin de lograr el mejor desempeño posible. Estos sistemas se caracterizan por la habilidad de observar variaciones en el comportamiento de la señal de entrada y reaccionar a esos cambios por medio de la adaptación de sus parámetros en pos de mejorar el desempeño de la señal de salida deseada. Los sistemas adaptivos tienen la habilidad de "aprender", de esta forma pueden adaptar de manera apropiada el desempeño cuando el ambiente del sistema cambia.

La arquitectura de un sistema adaptivo cuya tarea es transformar la señal de entrada de tal forma que sea lo más cercana posible a una señal de referencia es presentada en la Figura 2. La definición de la señal de referencia es crucial y específica de cada aplicación. La diferencia entre la salida y la señal de referencia es utilizada como una medida de la calidad del sistema adaptivo. Esta diferencia es llamada la señal de error y juega un rol importante en la adaptación de los parámetros del sistema.

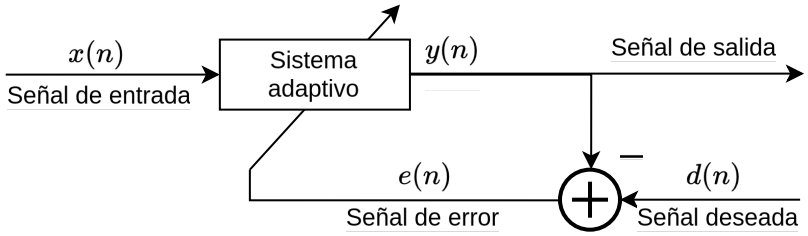


Figura 2: Sistema adaptivo

2.1. Sumador lineal adaptivo

La estructura básica en un sistema adaptivo es un sumador linear (sistema de respuesta impulsional finita). La señal de salida $y(n)$ es una combinación lineal de la señal de entrada $x(n)$ en el instante n considerado y sus $N-1$ valores previos.

$$\begin{aligned} y(n) &= h_0 x(n) + h_1 x(n-1) + \cdots + h_{N-1} x(n-N+1) \\ &= \sum_{i=0}^{N-1} h_i x(n-i) \end{aligned}$$

La descripción y análisis de este tipo de sistemas es relativamente sencilla. El sistema es lineal. En adición, el sistema con respuesta impulsional finita es siempre estable, para cualquier par de coeficientes finitos. Finalmente, la realización de estos sistemas es sencilla. En el caso de los sistemas adaptivos los coeficientes h_i cambian sus valores en el tiempo. Este simple sistema es llamado sumador lineal adaptivo. Tomando en consideración la naturaleza variante en el tiempo de los coeficientes, el sistema es descripto por

$$\begin{aligned} y(n) &= h_0 x(n) + h_1(n) x(n-1) + \cdots + h_{N-1}(n) x(n-N+1) \\ &= \sum_{i=0}^{N-1} h_i(n) x(n-i) \end{aligned}$$

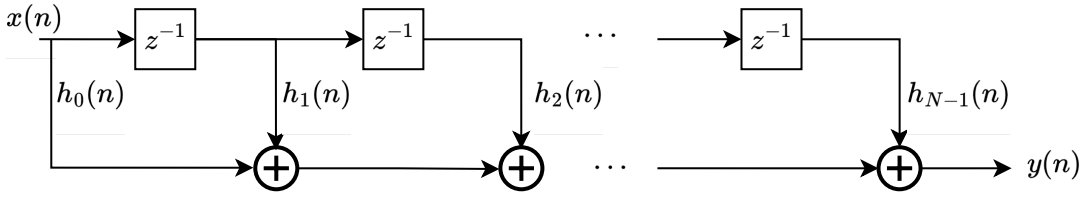


Figura 3: Sumador lineal adaptivo

El proceso de adaptación consiste de un algoritmo apropiado que cambie el valor de los coeficientes $h_i(n)$ con el objetivo de conseguir el desempeño deseado del sistema. La realización del sumador lineal adaptivo es representada en la Figura 3.

Los vectores variantes en el tiempo

$$\mathbf{X}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix}_{N\times 1} \qquad \mathbf{H}(n) = \begin{bmatrix} h_0(n) \\ h_1(n) \\ \vdots \\ h_{N-1}(n) \end{bmatrix}_{N\times 1}$$

son introducidos para la descripción y análisis de este sistema. El vector $\mathbf{X}(n)$ comúnmente consiste del valor actual de la entrada $x(n)$ y sus $N-1$ muestras pasadas, mientras que los elementos de $\mathbf{H}(n)$ son los coeficientes del sistema $h_i(n)$ en el instante actual n. La señal de salida puede ser escrita como el producto de estos dos vectores

$$y(n) = \mathbf{X}^T(n)\mathbf{H}(n) = \mathbf{H}^T(n)\mathbf{X}(n) \tag{1}$$

cuyo resultado es un escalar.

2.1.1. La señal error

La señal error $e(n)$ presente en el diagrama en bloques de la Figura 2 es igual a la diferencia entre la señal de referencia $d(n)$ y la señal de salida $y(n)$,

$$e(n) = d(n) - y(n).$$

En sistemas adaptivos, el objetivo del proceso de adaptación (aprendizaje) es ajustar los coeficientes del sistema de manera que la salida sea lo más cercana posible a la señal de referencia. En un caso ideal $y(n) \equiv d(n)$ debería cumplirse cuando $e(n) = 0$.

El primer paso en la definición de un algoritmo apropiado de adaptación para la modificación de los parámetros del sistema, es definir una métrica de la diferencia entre la señal de referencia y la señal de salida. El valor esperado del error $e(n)$ **no suele ser** una buena opción. Esto puede demostrarse por medio de un simple ejemplo; asuma que se obtienen las siguientes 6 muestras del error en 3 realizaciones distintas

$$\begin{aligned} s_0 &: [0, 0, 0, 0, 0, 0] \\ s_1 &: [-20, 20, -20, 20, -20, 20] \\ s_1 &: [0{,}1, 0{,}1, 0{,}1, 0{,}1, 0{,}1, 0{,}1] \end{aligned}$$

el valor promedio de la señal de error en las primeras dos realizaciones es cero, mientras que en la última es de 0,1. Esto genera que se tomen malas decisiones a partir de esta métrica.

Una métrica comúnmente utilizada es el error cuadrático medio (MSE, por sus siglas en inglés)

$$\varepsilon = E\left[e^2(n)\right],$$

donde $E[\cdot]$ representa el valor esperado. Del ejemplo previo se obtiene $\varepsilon = 0$ para el primer caso, $\varepsilon = 400$ para el segundo caso y $\varepsilon = 0{,}01$ para el último caso. Se nota que este tipo de métrica se comporta de manera esperada según nuestra necesidad.

En general, una función $J(e)$, comúnmente conocida como función de costo, es utilizada para definir la desviación de la señal error $e(n)$ del caso ideal. A partir del ejemplo anterior, una posible función de costo es el error cuadrático medio

$$J(e) = E\left[e^2(n)\right]$$

Considere ahora el cuadrado de la señal error en el sumador lineal adaptivo

$$\begin{aligned} e^2(n) &= (d(n) - y(n))^2 = \left(d(n) - \mathbf{H}^T(n)\mathbf{X}^T(n)\right)^2 \\ &= d^2(n) - 2d(n)\mathbf{H}^T(n)\mathbf{X}(n) + \mathbf{H}^T(n)\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{H}(n) \end{aligned}$$

En el cálculo del error cuadrático medio se debe tener en cuenta que las señales $d(n)$ y $x(n)$ *son aleatorias*, mientras que los coeficientes del sistema $\mathbf{H}(n)$ son variable determinísticas

$$\begin{aligned} \varepsilon &= E\left[e^2(n)\right] \\ &= E\left[d^2(n) - 2d(n)\mathbf{H}^T(n)\mathbf{X}(n) + \mathbf{H}^T(n)\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{H}(n)\right] \\ &= E\left[d^2(n)\right] - 2\mathbf{H}^T(n)E\left[d(n)\mathbf{X}(n)\right] + \mathbf{H}^T(n)E\left[\mathbf{X}(n)\mathbf{X}^T(n)\right]\mathbf{H}(n) \end{aligned} \tag{2}$$

Analizando los términos en (2) se tiene que

$$E\left[d^2(n)\right] = \sigma_d^2 \quad \text{Considerando } \mu_d = 0$$

luego,

$$E\left[d(n)\mathbf{X}(n)\right] = E[d(n)\begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix}] = \begin{bmatrix} E[d(n)x(n)] \\ E[d(n)x(n-1)] \\ \vdots \\ E[d(n)x(n-N+1)] \end{bmatrix}.$$

Los elementos de $E[d(n)x(m)]$ son las correlaciones cruzadas de la señal de referencia con la señal de entrada, denotadas como $r_{dx}(n,m) = E[d(n)x(m)]$. Para señales estacionarias $r_{dx}(n,m)$ es una función de la diferencia de índices solamente, $E[d(n)x(m)] = r_{dx}(n-m)$. De esta forma, diremos que

$$E[d(n)x(m)] = \begin{bmatrix} r_{dx}(n,n) \\ r_{dx}(n,n-1) \\ \vdots \\ r_{dx}(n,n-N+1) \end{bmatrix} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ \vdots \\ r_{dx}(N-1) \end{bmatrix} = \mathbf{r}_{dx} \tag{3}$$

El último término en (2) es de la forma

$$\mathbf{X}(n)\mathbf{X}^T(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix} [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]$$

$$= \begin{bmatrix} x(n)x(n) & x(n)x(n-1) & \ldots & x(n)x(n-N+1) \\ x(n-1)x(n) & x(n-1)x(n-1) & \ldots & x(n-1)x(n-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(n-N+1)x(n) & x(n-N+1)x(n-1) & \ldots & x(n-N+1)x(n-N+1) \end{bmatrix}$$

la media de esta expresión es

$$E[\mathbf{X}(n)\mathbf{X}^T(n)] = \mathbf{R}(n) =$$
$$\begin{bmatrix} r_{xx}(n,n) & r_{xx}(n,n-1) & \ldots & r_{xx}(n,n-N+1) \\ r_{xx}(n-1,n) & r_{xx}(n-1,n-1) & \ldots & r_{xx}(n-1,n-N+1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(n-N+1,n) & r_{xx}(n-N+1,n-1) & \ldots & r_{xx}(n-N+1,n-N+1) \end{bmatrix}$$

Para una señal $x(n)$ estacionaria

$$E[\mathbf{X}(n)\mathbf{X}^T(n)] = \mathbf{R} =$$
$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \ldots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \ldots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \ldots & r_{xx}(0) \end{bmatrix} \tag{4}$$

En la derivación de esta relación, se utiliza la función de autocorrelación $r_{xx}(n,m) = E[x(n)x(m)]$. En el caso de señales estacionarias $r_{xx}(n,m) = r_{xx}(n-m)$. Además, la función de autocorrelación de señales reales es par $r_{xx}(-n) = r_{xx}(n)$. Esto implica que la matriz de autocorrelación $\mathbf{R}$ no depende del instante actual n.

El error cuadrático medio puede ser escrito entonces como

$$\varepsilon = \sigma_d^2 - 2\mathbf{H}^T(n)\mathbf{r}_{dx} + \mathbf{H}^T(n)\mathbf{R}\mathbf{H}(n) \tag{5}$$

La tarea del sistema adaptivo es encontrar los coeficientes en el vector $\mathbf{H}(n)$ que produzcan el mínimo error cuadrático medio ε. En (5) se tiene un vector $\mathbf{r}_{dx}$ de correlaciones cruzadas entre la señal de referencia y la señal de entrada y también una matriz de autocorrelación $\mathbf{R}$ de la señal de entrada. Si el comportamiento estadístico de estas señales es conocido, es posible encontrar $\mathbf{r}_{dx}$ y $\mathbf{R}$. Si ese no es el caso, los elementos de la matriz de autocorrelación pueden ser estimados promediando las muestras de entrada. Esta estimación se deriva de asumir que la señal de entrada es un proceso ergódico.

Considere un ejemplo simple de un sumador lineal adaptivo de segundo orden ($N = 2$). En esta situación se tiene que

$$\mathbf{X}(n) = \begin{bmatrix} x(n) \\ x(n-1) \end{bmatrix} \quad \mathbf{H}(n) = \begin{bmatrix} h_0(n) \\ h_1(n) \end{bmatrix}.$$

El error cuadrático medio (de acuerdo con (5)) es

$$\begin{aligned}\varepsilon &= \sigma_d^2 - 2[h_0 h_1]\begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \end{bmatrix} + [h_0 h_1]\begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix}\begin{bmatrix} h_0 \\ h_1 \end{bmatrix} = \\ &= \sigma_d^2 - 2r_{dx}(0)h_0 - 2r_{dx}(1)h_1 + r_{xx}(0)h_0^2 + 2r_{xx}(1)h_0 h_1 + r_{xx}(0)h_1^2\end{aligned} \tag{6}$$

El error cuadrático medio ε se presenta como una función de dos variables h_0 y h_1 en la Figura 4. A partir de esta figura se concluye que la función error es un paraboloide, cuyo mínimo ocurre en (h_0^*, h_1^*). El objetivo es encontrar esos puntos.

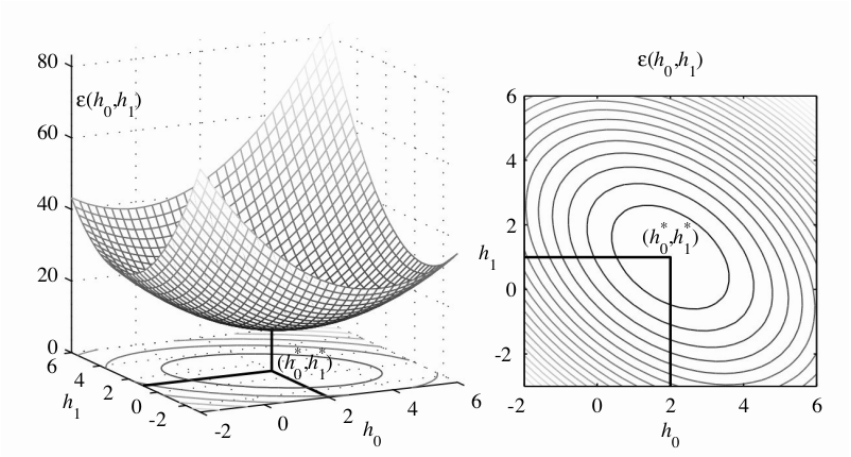


Figura 4: Error cuadrático medio como una función de los coeficientes h_0 y h_1

El proceso de minimización puede ser hecho a través de métodos numéricos o de forma analítica utilizando el cálculo de las derivadas parciales. Tomando derivadas parciales se tiene que

$$\begin{aligned}\frac{\partial \varepsilon}{\partial h_0} &= -2r_{dx}(0) + 2r_{xx}(0)h_0 + 2r_{xx}(1)h_1 \\ \frac{\partial \varepsilon}{\partial h_1} &= -2r_{dx}(1) + 2r_{xx}(1)h_0 + 2r_{xx}(0)h_1\end{aligned}$$

igualando ambas expresiones a 0

$$\begin{aligned}r_{xx}(0)h_0 + r_{xx}(1)h_1 &= r_{dx}(0) \\ r_{xx}(1)h_0 + r_{xx}(0)h_1 &= r_{dx}(1)\end{aligned}$$

o en forma matricial

$$\mathbf{RH} = \mathbf{r}_{dx}.$$

La solución de esta ecuación matricial

$$\mathbf{H}^* = \mathbf{R}^{-1}\mathbf{r}_{dx}$$

produce los coeficientes óptimos del sistema, denotados por h_0^* y h_1^*. Este es el filtro óptimo de Wienner.

Es importante remarcar que este tipo de cálculo para los coeficientes óptimos requiere parámetros estadísticos de la señal de entrada (en $\mathbf{R}$) y su correlación cruzada con la señal de referencia (en $\mathbf{r}_{dx}$). En adición, requiere una inversión matricial de orden N, la cual es una operación numéricamente demandante (una inversión matricial requiere del orden de $O(N^3)$ operaciones).

2.2. Método del paso descendente (Steepest Descend Method)

Considere el filtro óptimo cuyos coeficientes son obtenidos al minimizar el error cuadrático medio

$$\varepsilon = E\left[e^2(n)\right]$$

Esta minimización resulta en la ecuación matricial

$$\mathbf{RH} = \mathbf{r}_{dx}$$

Su solución produce los valores de los coeficientes óptimos del sistema adaptivo. En pos de evitar realizar la inversión matricial de $\mathbf{R}$, la solución de este sistema se obtiene utilizando un método iterativo. Uno de los métodos iterativos es el método "*steepest descend*".

Este método consiste de un valor inicial arbitrario para el vector de coeficientes $\mathbf{H}_0$ en el primer *paso* (usualmente $\mathbf{H}_0 = \mathbf{0}$). Luego los coeficientes son modificados de una forma iterativa para minimizar el error cuadrático medio ε. La dirección de máximo crecimiento de la función ε se encuentra definida por el gradiente de la misma $\nabla\varepsilon$. Sin embargo, para el presente caso de estudio, se debe tomar la dirección de máximo decrecimiento, es decir $-\nabla\varepsilon$. Es esta forma, el primer paso de la iteración es

$$\mathbf{H}_1 = \mathbf{H}_0 + \frac{\mu}{2}\left(-\nabla\varepsilon\right)\Big|_{\mathbf{H}=\mathbf{H}_0}$$

donde $\mu/2$ define el paso en la dirección de decrecimiento. En general las iteraciones están definidas por

$$\mathbf{H}_{n+1} = \mathbf{H}_n + \frac{\mu}{2}\left(-\nabla\varepsilon\right)\Big|_{\mathbf{H}=\mathbf{H}_n}$$

para $n = 0, 1, \ldots$. Una manera usual de detener el proceso iterativo es definido por medio de la diferencia de dos iteraciones consecutivas $\mathbf{H}_{n+1} - \mathbf{H}_n$. Si los coeficientes son muy parecidos, el proceso de iteración finaliza.

El vector gradiente puede ser escrito como

$$\frac{\partial\varepsilon}{\mathbf{H}} = -2\mathbf{r}_{dx} + 2\mathbf{RH}$$

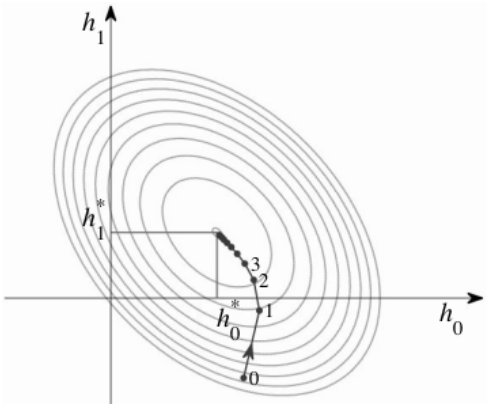
luego, la relación iterativa puede escribirse como

$$\mathbf{H}_{n+1} = \mathbf{H}_n + \mu\left(\mathbf{r}_{dx} - \mathbf{RH}_n\right) \tag{7}$$

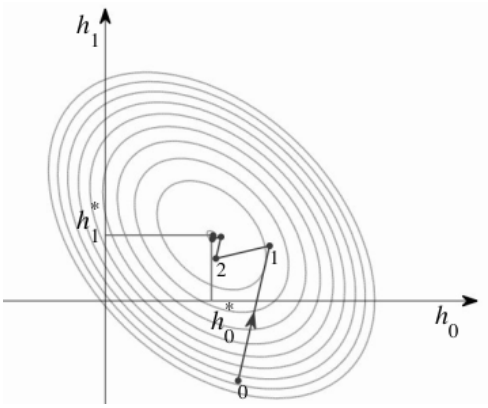
La elección de μ es crucial para el desempeño del algoritmo. Valores muy pequeños de μ garantizarán la convergencia de el algoritmo, pero a costa de gran número de iteraciones. Valores grandes de μ reducen el número de iteraciones pero pueden provocar que el algoritmo diverja o nunca converja y se quede iterando indefinidamente.

El método del paso descendente es ilustrado en la Figura 5.

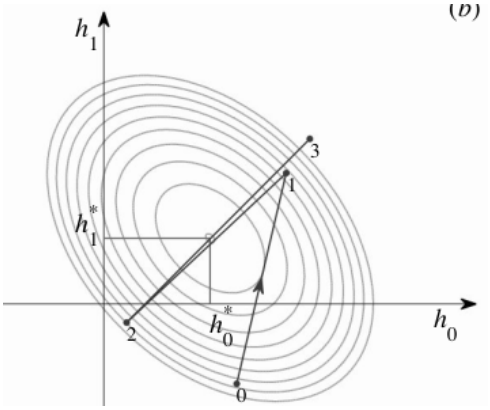
De esta forma, se evita el problema de tener que realizar una inversión matricial. Sin embargo, es necesario tener conocimiento del comportamiento estadístico de la señal de entrada $x(n)$ y la señal de referencia $d(n)$.



(a) Valores pequeños de μ



(b) Valores razonables para μ



(c) Valores grandes de μ

Figura 5: Ilustración del método del paso descendente

2.3. Algoritmo LMS

Considere la primer iteración en el algoritmo de paso descendente cuyo índice de iteración es k. En general, la señal de entrada $x(n)$ y la señal de referencia $d(n)$ son señales no estacionarias. Sus parámetros estadísticos pueden cambiar con el instante actual n. Como consecuencia de ello, los coeficientes del sistema adaptivo cambian en el tiempo. Este procedimiento iterativo es realizado de acuerdo a

$$\mathbf{H}_{k+1}(n) = \mathbf{H}_n + \mu\left(\mathbf{r}_{dx}(n) - \mathbf{R}(n)\mathbf{H}_k(n)\right)$$

Las propiedades estadísticas de las señales no varían lo suficientemente rápido en el tiempo. Para cada instante siguiente n se podrían utilizar los coeficientes obtenidos en el instante previo $n-1$ (en K iteraciones) como valores iniciales

$$\mathbf{H}_0(n) = \mathbf{H}_K(n-1)$$

Asuma que sólo una iteración es realizada para cada instante n. Con $K = 1$ se tiene que

$$\begin{aligned}
\mathbf{H}_1(n) &= \mathbf{H}_0(n) + \mu\left(\mathbf{r}_{dx}(n) - \mathbf{R}(n)\mathbf{H}_0(n)\right) \\
\mathbf{H}_1(n) &= \mathbf{H}_1(n-1) + \mu\left(\mathbf{r}_{dx}(n) - \mathbf{R}(n)\mathbf{H}_1(n-1)\right) \\
\mathbf{H}_1(n+1) &= \mathbf{H}_1(n) + \mu\left(\mathbf{r}_{dx}(n+1) - \mathbf{R}(n+1)\mathbf{H}_1(n)\right)
\end{aligned}$$

Para simplificar la notación, el índice que denota la iteración se omite (se ha asumido que es 1), entonces se tiene que

$$\mathbf{H}(n+1) = \mathbf{H}(n) + \mu\left(\mathbf{r}_{dx}(n+1) - \mathbf{R}(n+1)\mathbf{H}(n)\right)$$

En el algoritmo LMS la matriz de autocorrelación $\mathbf{R}(n+1)$ y la correlación cruzada $\mathbf{r}_{dx}(n+1)$ son aproximadas por sus valores instantáneos

$$\mathbf{r}_{dx}(n+1) \approx \mathbf{r}_{dx}(n) \approx d(x)\mathbf{X}(n)$$
$$\mathbf{R}(n+1) \approx \mathbf{R}(n) \approx \mathbf{X}(n)\mathbf{X}^T(n)$$

Utilizando esta aproximación, la fórmula de iteración resulta en

$$\begin{aligned}\mathbf{H}(n+1) &= \mathbf{H}(n) + \mu\left(d(n)\mathbf{X}(n) - \mathbf{X}(n)\mathbf{X}^T(n)\mathbf{H}(n)\right) \\ &= \mathbf{H}(n) + \mu\mathbf{X}(n)\left(d(n) - \mathbf{X}^T(n)\mathbf{H}(n)\right)\end{aligned}$$

y sabiendo que $y(n) = \mathbf{X}^T(n)\mathbf{H}(n)$

$$\mathbf{H}(n+1) = \mathbf{H}(n) + \mu\left(d(n) - y(n)\right)\mathbf{X}(n)$$

donde la diferencia $d(n) - y(n)$ es la señal error $e(n)$. Una representación habitual del algoritmo LMS es

$$\mathbf{H}(n+1) = \mathbf{H}(n) + \mu e(n)\mathbf{X}(n) \tag{8}$$

En cada instante de tiempo, los coeficientes del sistema adaptivo cambian respecto a sus valores previos en la dirección del vector señal de entrada $\mathbf{X}(n)$. La intensidad del cambio es determinada por el paso μ y la señal error del instante previo $e(n)$.

Para un sistema de orden N, el algoritmo LMS es muy eficiente numéricamente ya que en cada instante n realiza sólo $N+1$ multiplicaciones y N sumas.

3. Filtro adaptivo

Una vez desarrollado los conceptos teóricos detrás de la propuesta de proyecto, se inicia con el modelado del problema a través de simulaciones con Python. En primer lugar se realiza un modelado del filtro utilizando una representación en punto flotante, luego, con el fin de obtener un modelo fiel del diseño se utiliza la representación en punto fijo de las señales.

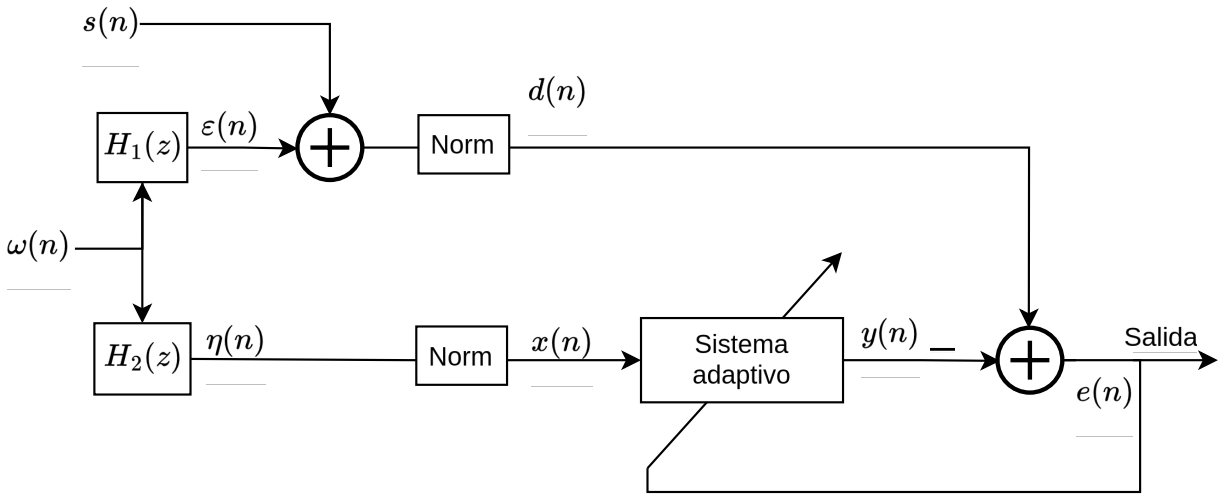


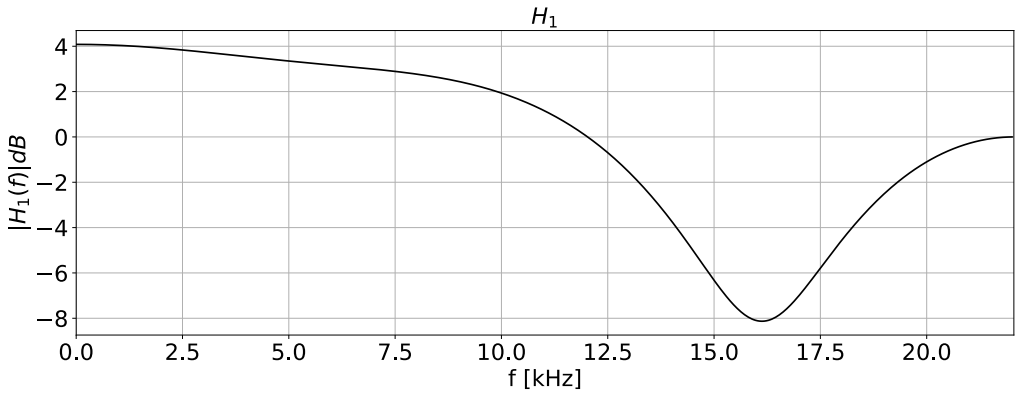
Figura 6: Modelado de filtro

En primer lugar es necesario modelar y representar de manera adecuada a las señales con las que se trabajan. En la Figura 6 se observa el diagrama en bloques del filtro y sus señales.

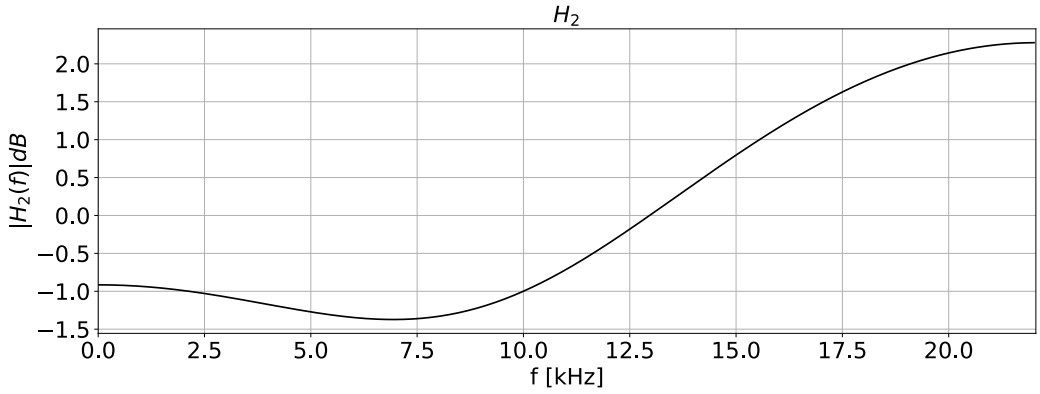
Allí se puede observar la presencia de dos filtros digitales cuyas transferencias son $H_1(z)$ y $H_2(z)$. Estos filtros se utilizan para modelar el efecto del multicamino y rebote en distintos objetos que experimenta la fuente común de ruido $\omega(n)$. En (9) se muestran el valor de los coeficientes de cada filtro

$$\begin{aligned} H_1(z) &= 1 + 0{,}5z^{-1} + 0{,}2z^{-2} - 0{,}2z^{-3} + 0{,}1z^{-4} \\ H_2(z) &= 1 - 0{,}2z^{-1} + 0{,}1z^{-2} \end{aligned} \tag{9}$$

La respuesta en frecuencia de los filtros $H_1(z)$ y $H_2(z)$ pueden observarse en la Figura 7. Resulta interesante observar que se trata de filtros antagónicos, uno pasa bajos y otro pasa altos. De esta forma, la señal de interés se verá fuertemente contaminada con ruido para las frecuencias bajas (frecuencias para las cuales la voz es muy usual).



(a)



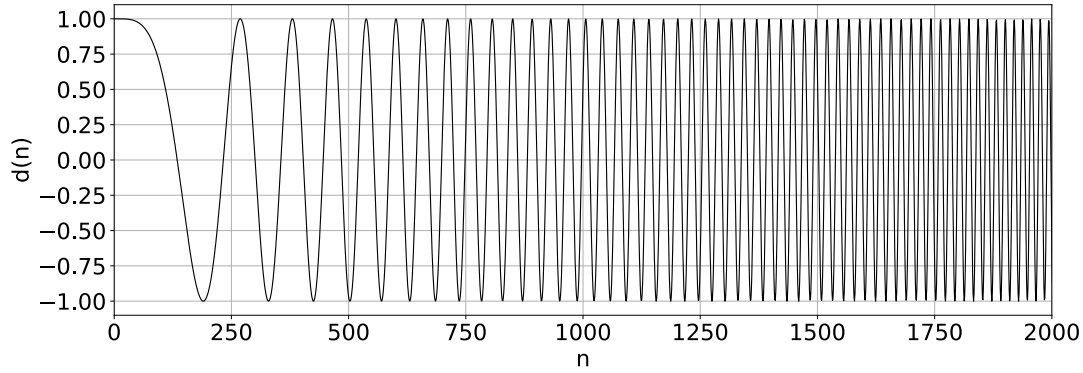
(b)

Figura 7: Respuesta en frecuencia de los filtros del ruido

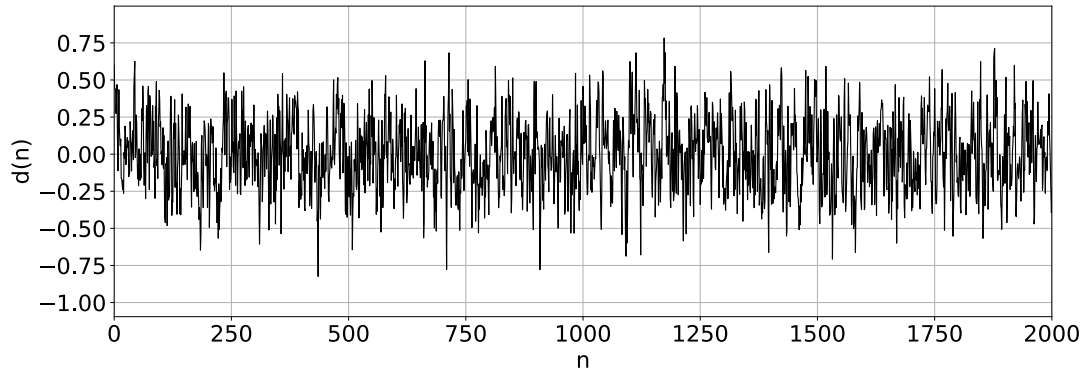
La frecuencia de muestreo utilizada es de $f_s = 44{,}1$ kHz. Esto es debido a que el filtro es diseñado con el objetivo de ser utilizado para aplicaciones que involucren al oído humano (cuyo ancho de banda va desde los 0 a 20 kHz en promedio).

Por otro lado, es necesario definir una señal de estímulo $s(n)$ que sea representativa para el rango audible. De esta forma, se propone utilizar una señal del tipo *Chirp* con un barrido en frecuencia que inicie en los 0 Hz y se extienda hasta la frecuencia de Nyquist ($f = fs/2$). Al mismo tiempo, la fuente de ruido $w(n)$ común a ambos filtros, es ruido

blanco Gaussiano de varianza $\sigma_\omega^2 = 5$. En la Figura 8 se puede observar las primeras 2.000 muestras de la señal de entrada y su representación luego de agregarle el ruido ambiente filtrado y normalizarla para que valor máximo sea 1.



(a) Señal sin ruido



(b) Señal contaminada

Figura 8: Estimulos de entrada

Como se observa en la Figura 8(b) la señal de entrada queda completamente oculta e indistinguible cuando se le suma el ruido ambiente.

3.1. Arquitectura

3.1.1. RTL - FIR

Comenzando desde un nivel de abstracción superior, se presenta la propuesta de arquitectura para el filtro adaptivo. En la Figura 9 se muestra el RTL del filtro. En dicha imagen es posible observar la presencia de 4 registros correspondientes al registro de desplazamiento debido al orden del filtro, 4 multiplicadores para realizar los productos parciales, y otros 4 sumadores. Por otra parte, se incluye el bloque LMS, el cual se encarga de calcular los valores de los coeficientes para la próxima iteración.

3.1.2. RTL - LMS

Una vez presentada la arquitectura del filtro, se procede a diseñar el bloque LMS que contiene el algoritmo encargado del cálculo de los coeficientes. En la Figura 10 se muestra la arquitectura de dicho algoritmo.

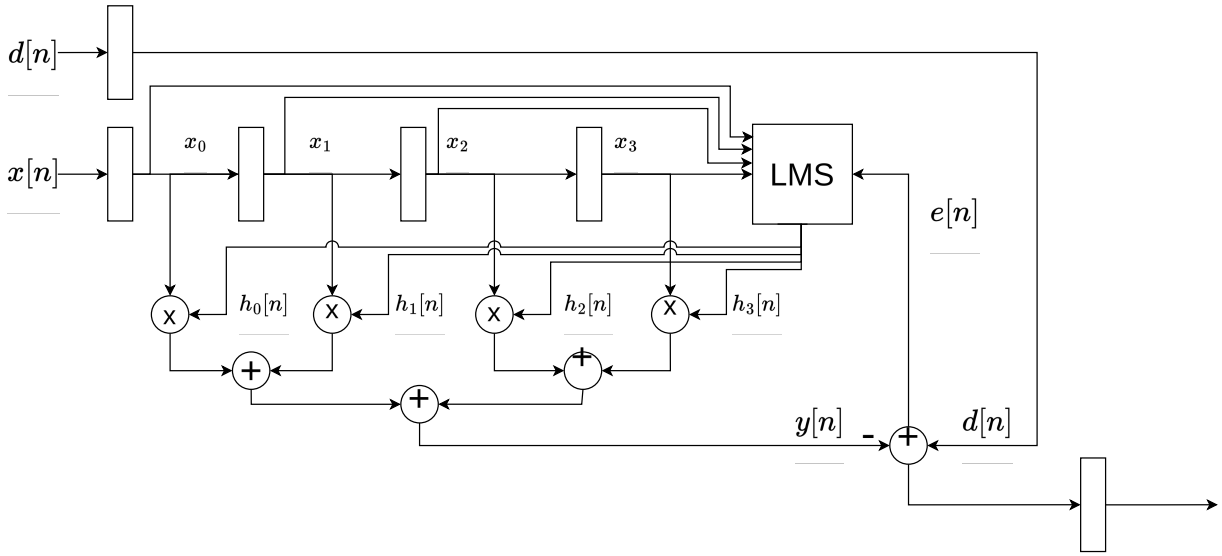


Figura 9: RTL del filtro

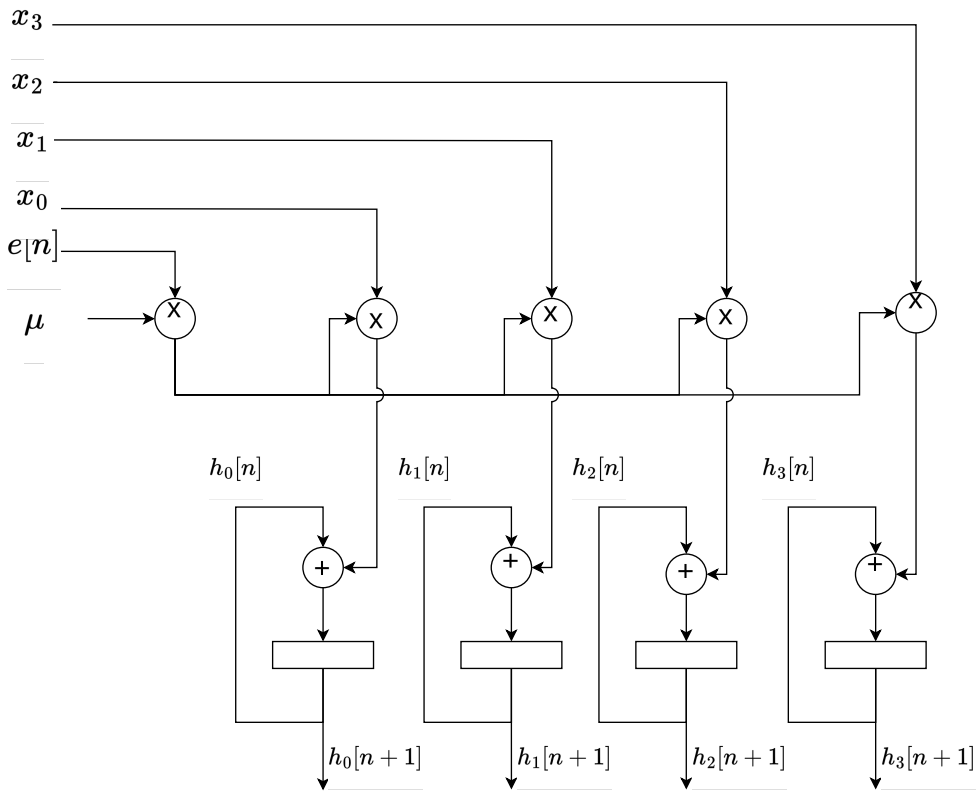


Figura 10: RTL del bloque LMS

3.2. Modelado en punto flotante

Como se menciona al inicio de la presente sección, es necesario realizar un modelado en punto flotante para comprobar el correcto funcionamiento del algoritmo y al mismo tiempo seleccionar los parámetros de configuración del filtro (como por ejemplo, el orden del filtro y el paso de iteración μ).

Para evitar seleccionar valores arbitrarios de diseño es necesario obtener una métrica sobre la cuál referirse para determinar cuándo un diseño es mejor que otro, es por ello que

se decide utilizar la relación señal a ruido a la salida del filtro (SNR). Su cálculo se basa en obtener la potencia de la señal error $e(n)$ y dividirla por la potencia de ruido, la cual se obtiene al restar el ruido $\varepsilon(n)$ con el ruido $\eta(n)$ filtrado por el filtro adaptivo ($y(n)$). De esta forma se tiene que

$$\text{SNR} = \frac{P_{e(n)}}{P_{\varepsilon(n)-y(n)}}$$

es la métrica que se utiliza para caracterizar el desempeño del sistema.

3.2.1. Seleccionando parámetros

El criterio seleccionado en el presente informe consiste en que la SNR presente a la salida del filtro no sea menor que 10dB. De esta forma, se comienza a iterar y buscar distintos valores de μ y M (el orden del filtro) que satisfagan dicho umbral.

Por medio del script de Python `filtro_adaptivo.ipynb` se corren distintas simulaciones con distintos valores de μ y M. Luego de 16 combinaciones distintas de parámetros se obtiene el Cuadro 1 en la cual se resume este proceso iterativo.

μ	**M=1**	**M=2**	**M=3**	**M=4**
0.01	0.734 dB	3.620 dB	5.518 dB	18.210 dB
0.05	0.722 dB	3.545 dB	5.383 dB	17.998 dB
0.1	0.718 dB	3.479 dB	5.320 dB	18.009 dB
0.5	0.734 dB	3.589 dB	5.472 dB	17.721 dB
1	0.614 dB	3.702 dB	5.450 dB	15.959 dB

Cuadro 1: Resultado del proceso iterativo para obtener parámetros de diseño

A partir del Cuadro 1 se deduce que el menor orden posible que cumple con el requerimiento de la SNR es el filtro de orden M=4, mientras que aquellos filtros cuyo orden sea menor a 4 presentan un desempeño menor al esperado. Por otro lado, queda en evidencia que a medida que el valor del paso de iteración μ aumenta, el desempeño se reduce, sin embargo, la ventaja es que se obtiene una menor cantidad de iteraciones para obtener el valor de los coeficientes del filtro.

Para decidir qué valor resulta adecuado, se debe considerar el contexto del diseño, al tratarse de un sistema adaptivo cuya aplicación está destinada al audio, la latencia del mismo puede relajarse debido a la respuesta en frecuencia relativamente lenta que tiene el oído humano. Esto permite seleccionar un valor de μ pequeño con el fin de obtener un mejor desempeño a costa de una latencia mayor que resulta indistinguible a fines prácticos para el oído humano.

A fines comparativos, en la Figura 11 se muestra la evolución de los coeficientes con $\mu = 1$ y $\mu = 0{,}05$ para un filtro de orden M=4 durante las primeras 5.000 muestras. En dicha imagen se observa que a partir de las 2.000 muestras los coeficientes logran estabilizarse alrededor de un valor para el caso de $\mu = 0{,}05$, mientras que para $\mu = 1$ lo hacen mucho antes (cerca de las 300 muestras). Sin embargo, considerando que la frecuencia de muestreo es de $f_s = 44{,}1$ kHz y que $\mu = 0{,}05$, los coeficientes alcanzarían la estabilidad aproximadamente a los 45 mS (2000/44.1 kHz) con una SNR de 18 dB, un tiempo completamente razonable.

De esta forma, los parámetros de diseño del filtro se definen como

- Orden del filtro: M=4
- Paso de iteración: $\mu = 0{,}05$

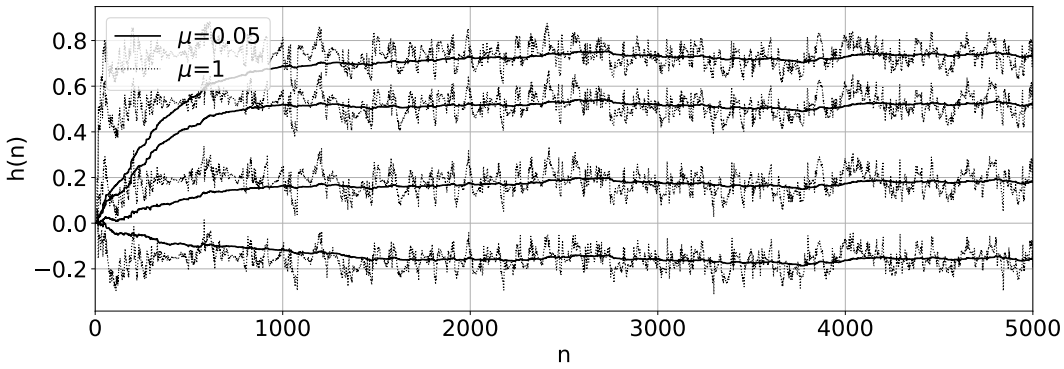


Figura 11: Evolución de coeficientes para diferentes pasos de iteración

3.2.2. Espectro

Una forma de visualizar el efecto del filtrado en la señal de salida es graficar su espectro y compararlo con la señal de entrada.

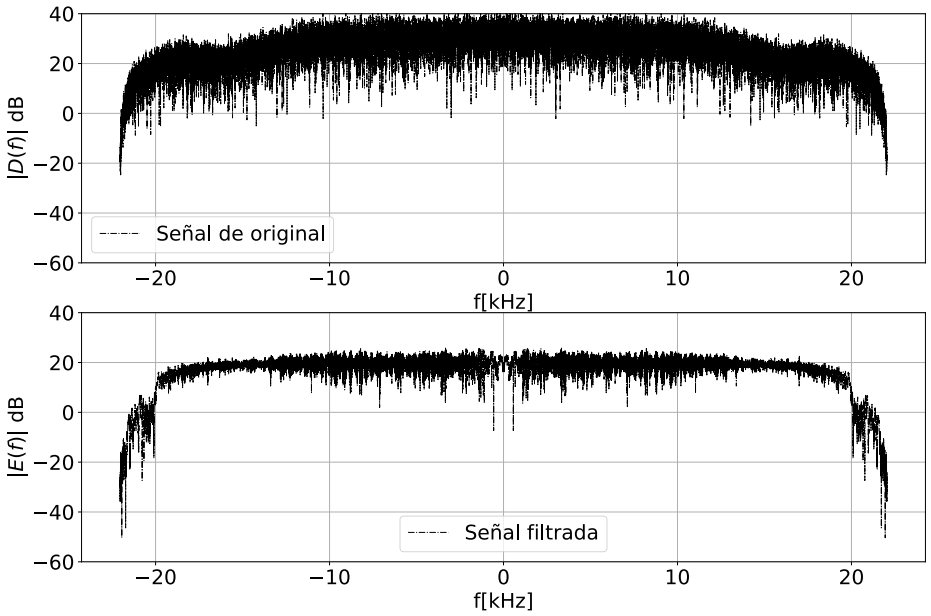


Figura 12: Espectro pre y post filtrado

En la Figura 12 se muestra dicha comparación. Allí se observa, en el espectro de la señal de entrada, la componente en frecuencia del ruido (presente en todo el ancho de banda por ser ruido blanco) junto con la componente en frecuencia de la señal de interés, que también ocupa todo el ancho de banda dado que se trata de una Chirp, como se ha visto. Al mismo tiempo se grafica el espectro de la señal de salida, cuya componente "ruidosa" se ve altamente atenuada debido al efecto del filtrado adaptivo. Esto permite deducir que el filtrado está comportándose de manera esperada (al mismo tiempo se reproduce la Chirp para comprobar de manera auditiva los efectos).

3.3. Modelado en punto fijo

Una vez definido el orden del filtro y paso de iteración del algoritmo LMS, se procede a realizar una análisis de implementación del módulo cuando se utiliza una representación de punto fijo.

Al igual que en la Sección 3.2.1, es necesario definir una métrica para determinar el desempeño de una representación sobre otra. Para ello, se decide utilizar como métrica de decisión al error cuadrático medio

$$\varepsilon = \frac{1}{N}\sum_{i=0}^{N-1} |x(i)|^2 \tag{10}$$

del espectro de la señal cuantizada respecto a la de punto flotante. Es decir

$$x(n) = E_{fix}(n) - E_{float}(n)$$

donde $E(n)$ corresponde con el espectro de salida del sistema $y(n) - d(n)$. De esta forma, es posible comparar el desempeño de una representación con otra, a través del impacto que genera en el espectro.

La Figura 13 muestra la posición en la cuál se realiza la cuantización de la señal de entrada. De esta forma, se evita tener que lidiar con los efectos de la cuantización de los filtros encargados de modelar los efectos de multicamino y rebotes presente en la señal de ruido común. El bloque Q indica el momento en donde las señales son cuantizadas, al mismo tiempo, el subíndice "fp" indica que se trata de señales de punto fijo.

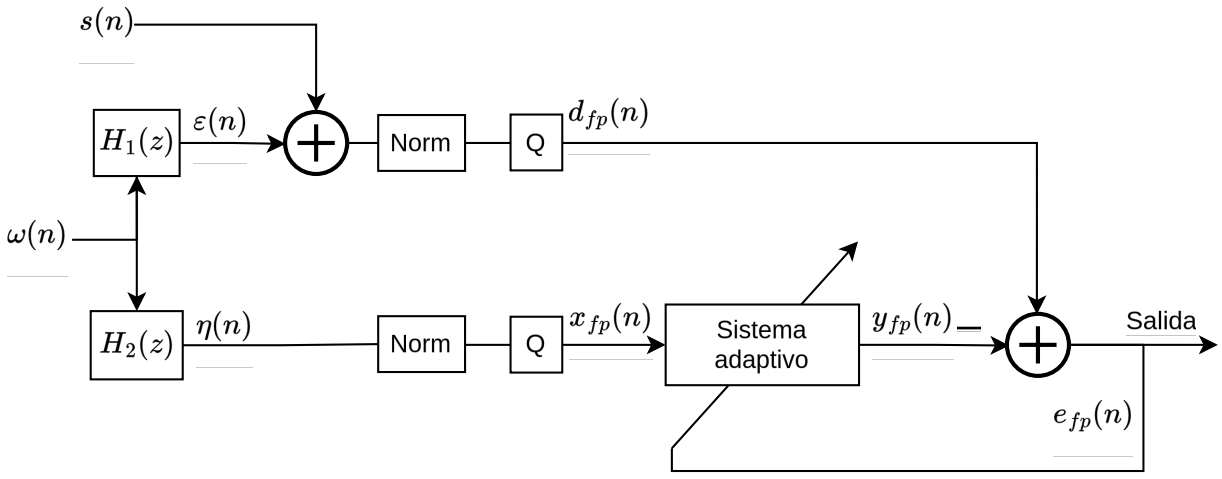
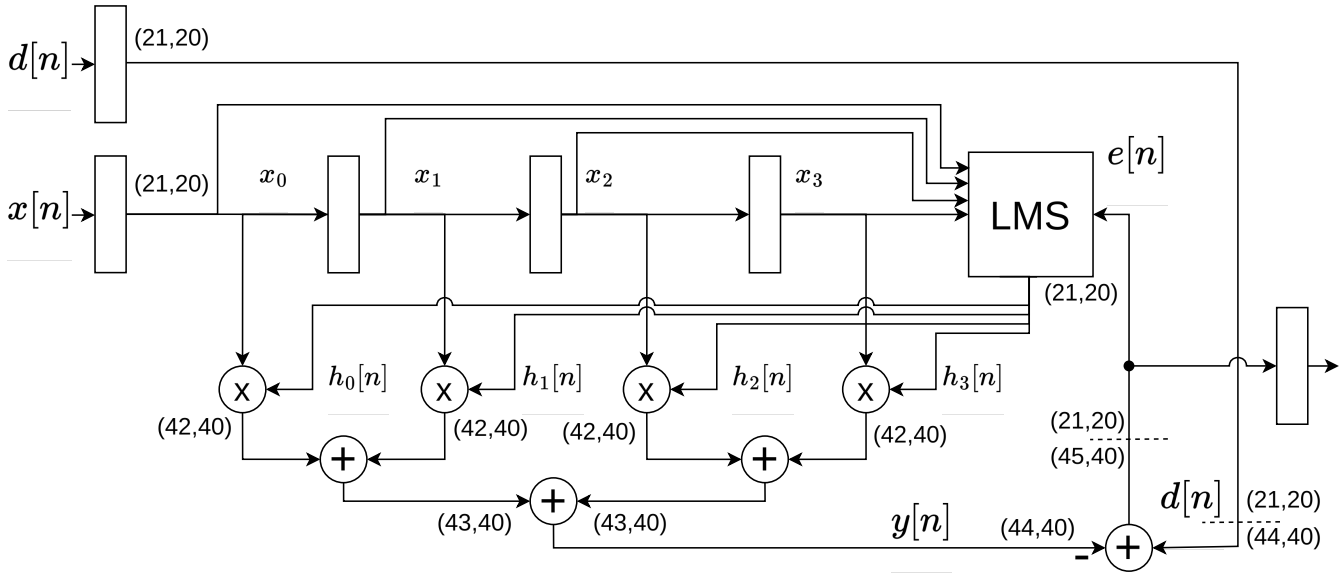


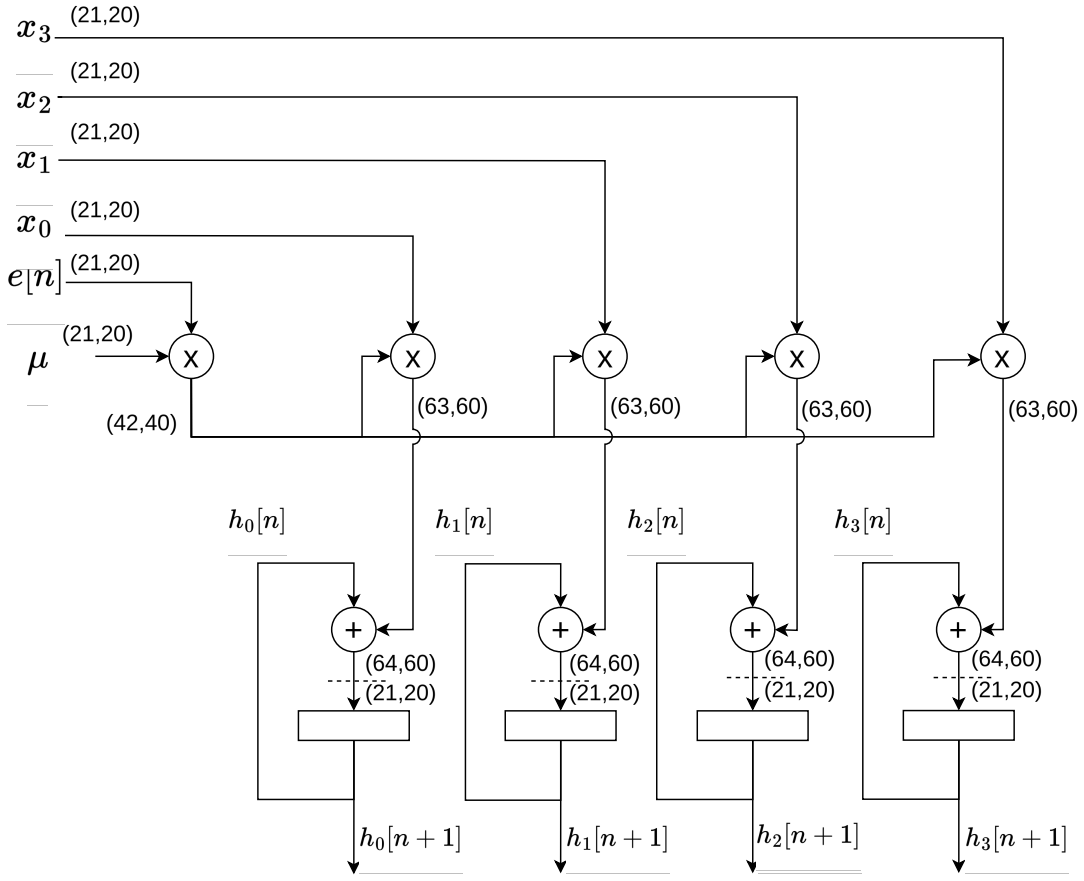
Figura 13: Diagrama en bloques del proceso de cuantización

3.3.1. Truncado y saturación

Las operaciones de producto junto con las sumas parciales y la recursividad del algoritmo LMS, genera la necesidad de aplicar operaciones de truncado y saturación para evitar un consumo de área excesivo en el diseño. Para evitar el uso de lógica extra y congestión en el ruteo de cables, se aplican las operaciones de truncado y saturación sólo en los puertos de salida y en operaciones recursivas para evitar lo que se conoce como *bit growing*. En la Figura 14(a) y 14(b) se muestra con línea punteada dónde se aplican dichas operaciones.



(a) FIR



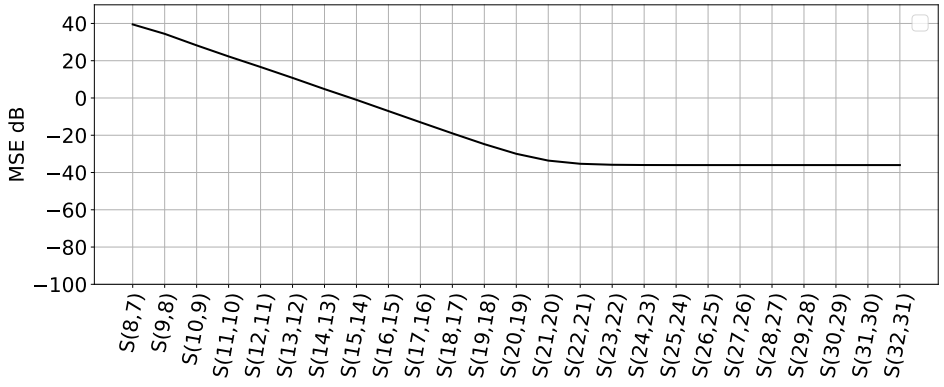
(b) LMS

Figura 14: Posición de operación de truncado y saturación

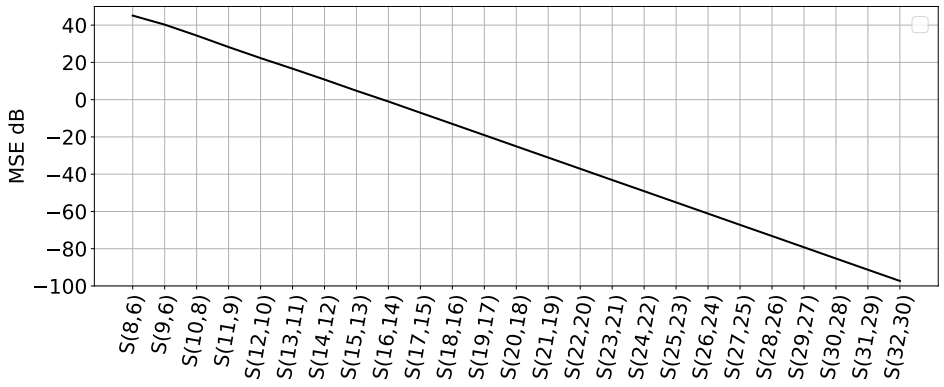
El siguiente paso es definir la dimensión con la cuál se debe representar a las señales. Realizando un procedimiento similar al realizado en el cálculo del valor de μ y el orden del filtro M, se corren varias simulaciones con diferentes tipos de representación, comenzando desde la representación S(8,7) y finalizando con S(32,31).

Luego, utilizando (10) se calcula el error cuadrático medio en cada caso. En función de ese resultado junto con la consideración de área ocupada, es que se selecciona una representación.

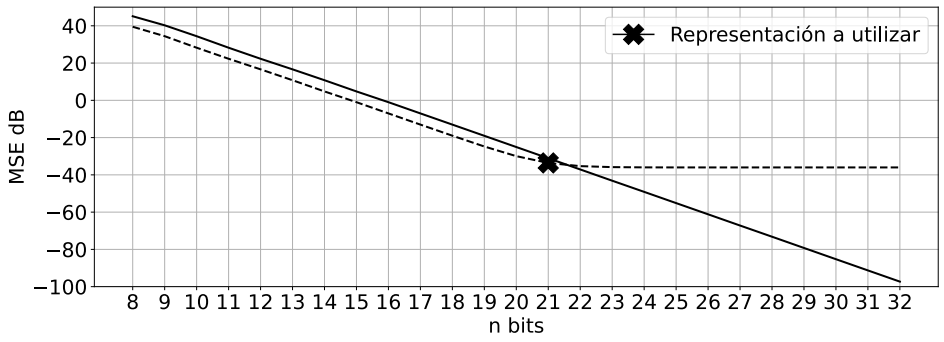
En las Figuras 15(a) y 15(b) se muestra el error cuadrático medio en dB en función del tipo de representación para 1 y 2 bits de parte entera. Luego, superponiendo ambas figuras se obtiene la Figura 15(c), a partir de ella es que se selecciona la representación S(21,20). Una representación de mayor orden con 2 bits enteros mejora aún mas el desempeño del filtro pero aumenta la utilización debido al incremento de los buses.



(a) 1 bit para la parte entera



(b) 2 bits para la parte entera



(c) Bits totales

Figura 15: MSE de espectro un punto flotante vs punto fijo

3.3.2. Espectro

En la Figura 16 se muestra el espectro de la señal en punto fijo en relación a la de punto flotante. Como se observa, ambas señales son muy similares

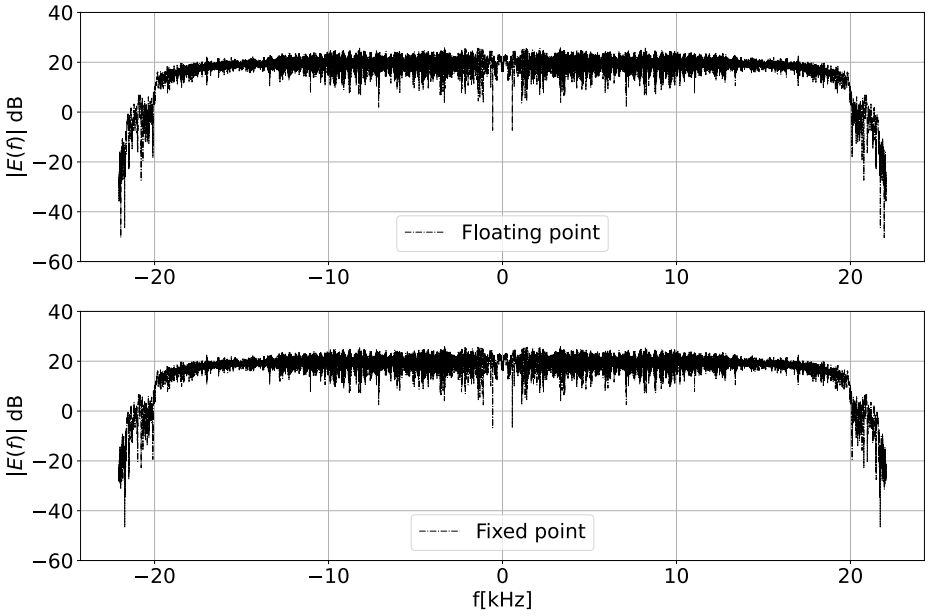


Figura 16: Espectro de la señal en punto flotante vs punto fijo

3.3.3. Testbench y VM

Ya con todos los parámetros definidos se procede a codificar el RTL con su respectivo testbench. Para comprobar el funcionamiento a nivel RTL, se generan los estímulos utilizando Python y se guardan esas señales en un archivo externo que luego será cargado en el diseño por medio de blocks RAMs para hacer las veces de las señales de entrada. En la Figura 17 se muestra un diagrama en bloques del testbench.

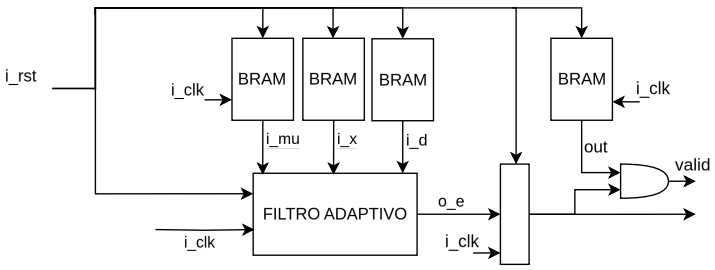


Figura 17: Diagrama en bloques del testbench

Al mismo tiempo, se genera una block RAM auxiliar que contiene los datos de salida esperados. Esto se realiza con el fin de comparar todos los datos de salida del DUT con los generados por simulación y ver si efectivamente coinciden los datos.

En la Figura 18 se observan las formas de onda de entrada `mic1` y `mic2`, salida `o_filter`, y comparación lógica para validar el dato `valid_out = out_mem == o_filter`, donde `out_mem` es otra block RAM generada en Python.

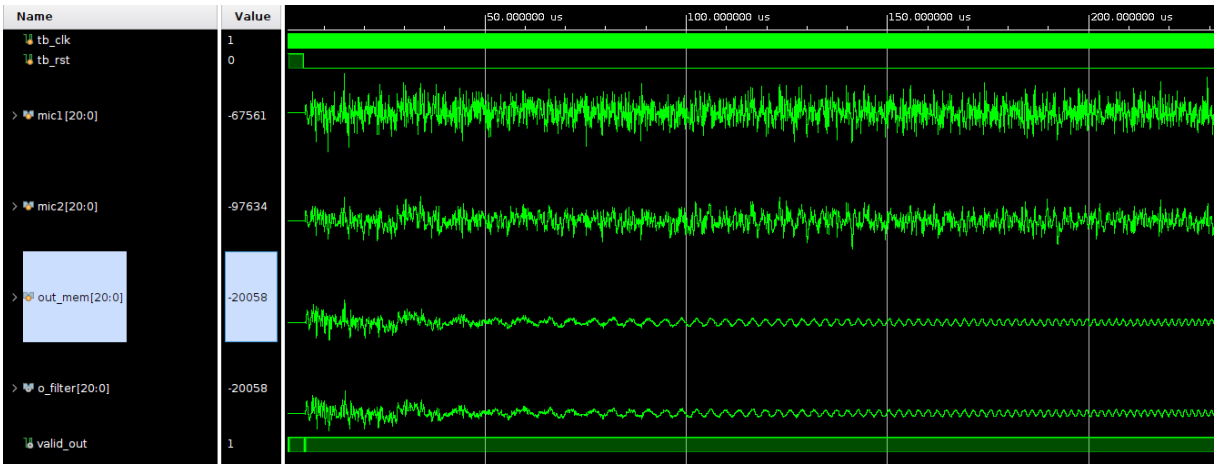


Figura 18: Waveforms de la simulación

3.4. Validación

Para validar el diseño se utilizan los IP cores VIO e ILA. El *Virtual Input/Output* es utilizado para generar las señales de estímulos al DUT, donde sus salidas son las entradas al DUT y las entradas son las salidas del mismo. Por otro lado, el *Integrated Logic Analyzer* se utiliza para poder visualizar la salida del DUT a partir de una señal de disparo.

El IP VIO consta de una única salida que controla la señal de reset del filtro.

El ILA por otra parte, contiene 2 entradas, una de 21 bits para visualizar los datos de salida del filtro y otra de 1 bit para utilizar como señal de disparo. En la configuración de este bloque, la cantidad de muestras que se obtengan del diseño juega un papel importante en tiempo de síntesis ya que si la cantidad de muestras que se desean guardar es muy elevada, deben utilizarse block RAM's cada vez más grandes, provocando que eventualmente no haya espacio suficiente en la FPGA. Es decir, ambos bloques ocupan espacio físico en el diseño, y su configuración debe realizarse teniendo en cuenta este aspecto. Es por eso que para este caso se decide utilizar un tamaño de logeo de 2^{14} muestras para cada señal.

3.4.1. Generación de clock

Debido a que el filtro funciona con una frecuencia menor a 100 MHz, es necesario generar otro reloj que sea divisor de esa frecuencia. De esta forma, por medio del IP *Clock Wizard* se genera un clock de 10 MHz a partir del clock de 100 MHz, en la Figura 19 se muestra el diagrama en bloques utilizado para la validación.

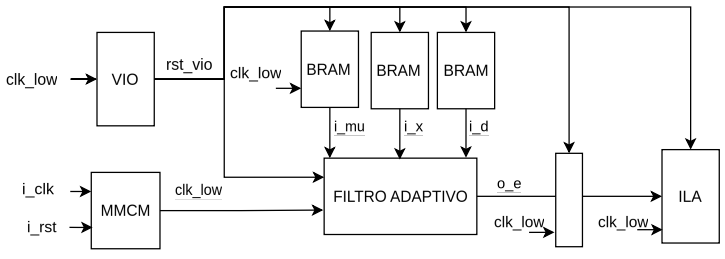


Figura 19: Diagrama en bloques de la validación

Finalmente, se realiza la conexión remota con el servidor, se programa la FPGA, y se obtienen las formas de onda a través del ILA. En la Figura 20 se muestran los resultados obtenidos.

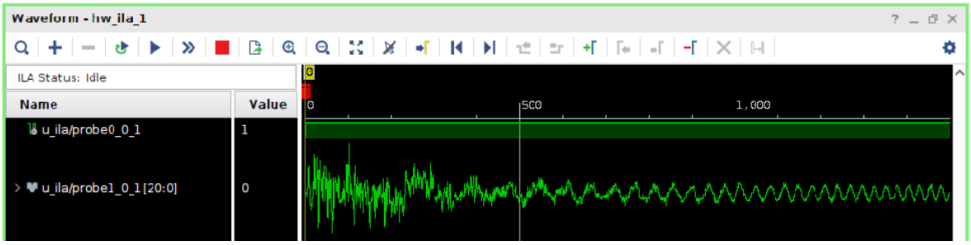
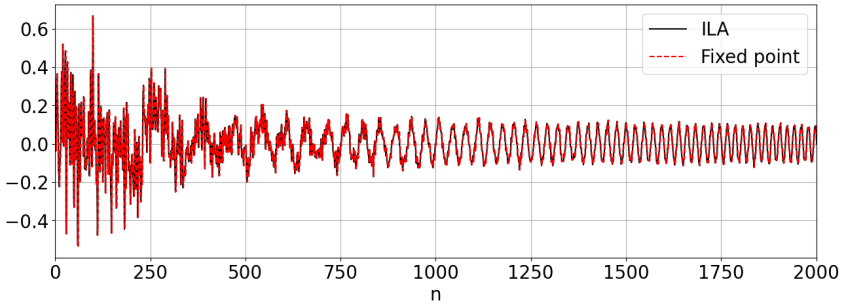
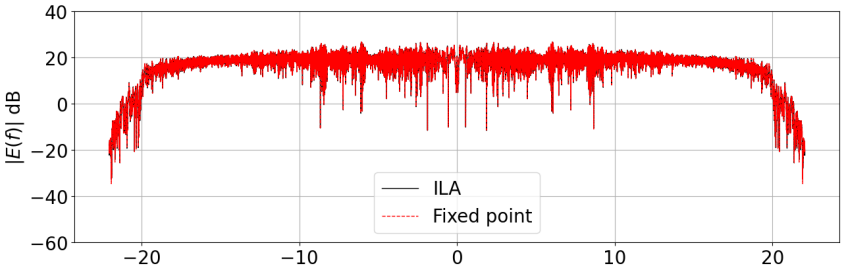


Figura 20: Waveforms del ILA

Por último, para comprobar si el diseño se comporta de manera esperada una vez que se haya grabado la FPGA, se exporta la forma de onda de salida, se calcula su espectro y se lo compara con el espectro obtenido por simulación. En la Figura 21 se muestra el espectro de la señal obtenida en la FPGA junto con el de la esperada por simulación.



(a) Respuesta temporal



(b) Respuesta en frecuencia

Figura 21: Datos de salida del ILA

3.5. Uso de recursos y desempeño

Una vez que el funcionamiento del filtro es validado, se procede a analizar los recursos que consume y cuál es la frecuencia máxima a la cual el diseño puede funcionar sin inconvenientes.

3.5.1. Área consumida

En primer lugar se obtiene el reporte de utilización del diseño. En el se encuentra información acerca de la cantidad de LUTs, celdas, bloques de memoria RAM, bloques DSP, etc, utilizados. Al mismo tiempo se puede conocer el porcentaje de área que ocupa dicho filtro en el dispositivo. En la Figura 22 se muestra el reporte de utilización.

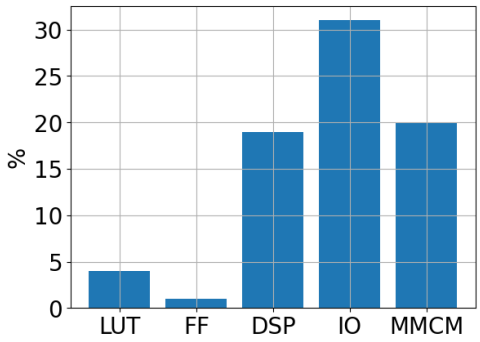


Figura 22: Reporte de utilización

A partir de la Figura 22 se observa que hay una cantidad considerable de bloques DSP utilizados. Esto se debe principalmente a las operaciones de producto realizadas. Por otra parte, dado que se cuenta con un tamaño de bus de datos de 21 bits, es lógico que al necesitar 3 entradas de datos (63 bits), 1 de salida (21 bits) 1 de reset, y 1 de clock (total 86 bits) la cantidad de puertos de entrada/salida aumente y se lleve un buen porcentaje de utilización.

En cuanto a los flip flop y las tablas de look-up, se observa que la cantidad de recursos demandados por el diseño es de 0,42 % y 4,21 % respectivamente, consumos relativamente bajo de recursos. Por el lado de las LUT, es debido a los bloques encargados de realizar las operaciones de truncado y saturación (comparaciones lógicas). Mientras que por el lado de los FF, se debe principalmente a los registros utilizados en los registros de desplazamiento.

3.5.2. Timing

Continuando con los aspectos de desempeño del sistema, lo siguiente que se analiza es la frecuencia máxima de funcionamiento del filtro. Para ello, se procede a realizar un histograma de slack con la herramienta Vivado y se comparan los resultados.

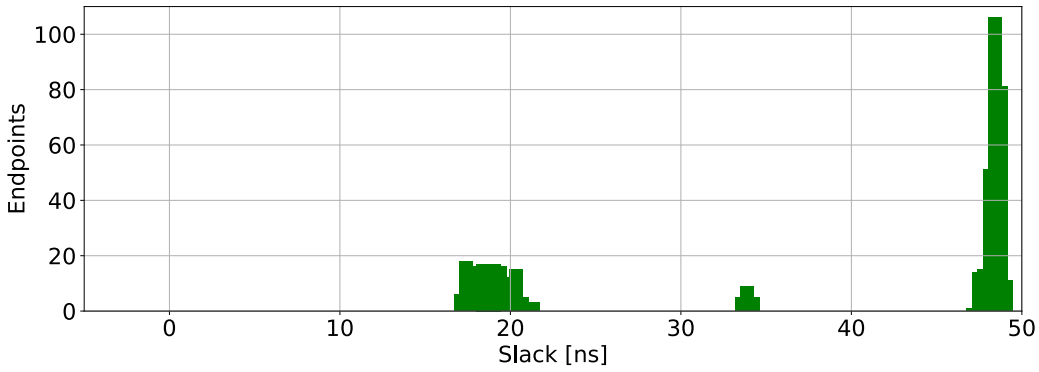
En la Figura 23 se muestran tres histogramas de slack para distintas frecuencias de trabajo. En primer lugar, en la Figura 23(a) se muestra el histograma del diseño para una frecuencia de trabajo de 20 MHz. Allí se observa que el peor slack ronda los 20 nS aproximadamente, por lo que el filtro puede funcionar sin preocupaciones en este rango de frecuencias. Estos endpoints representados en los 20 nS pueden atribuirse a las operaciones de producto ya que son las más demandantes computacionalmente.

Al aumentar la frecuencia de trabajo a 33 MHz, vemos en la Figura 23(b) que el peor slack se desplaza de manera considerable hacia el rango de los 0 nS, es sabido que trabajar bajo estas condiciones es riesgoso ya que podría llegar a no cerrar timing si se sube el diseño a la FPGA debido a que la estimación no es 100 % certera. El diseñador debe evitar trabajar en esta rango de frecuencias en la medida de lo posible.

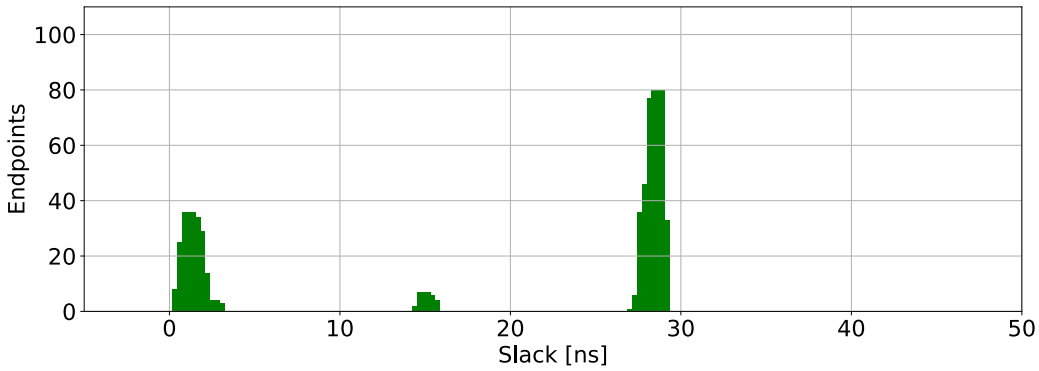
Finalmente, en la Figura 23(c) se muestra un histograma de slack para una frecuencia de trabajo de 35 MHz, aquí se deja en evidencia que el filtro no es capaz de funcionar a tan elevada frecuencia.

El peor slack negativo para cada frecuencia es entonces:

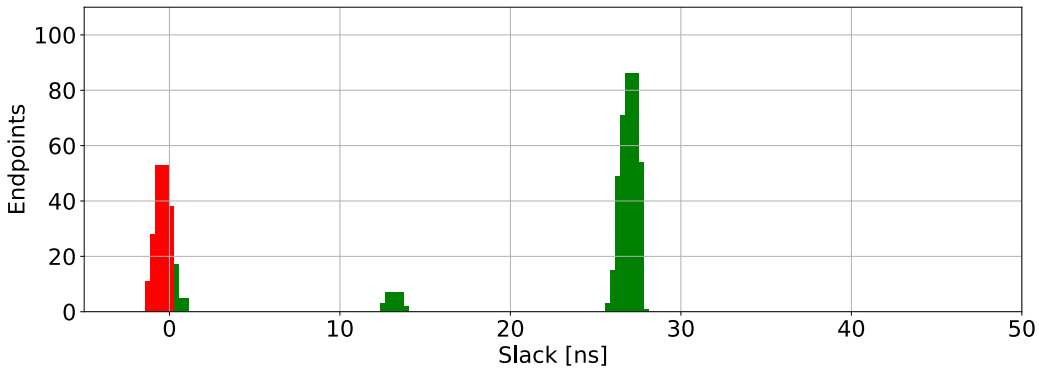
- 20 MHz: 17,100 nS
- 30 MHz: 0,554 nS
- 33 MHz: −1,005 nS



(a) 20 MHz



(b) 30 MHz



(c) 35 MHz

Figura 23: Histograma de slack a distintas frecuencias de trabajo

De esta forma, se establece que la frecuencia máxima de trabajo que soporta el filtro ronda aproximadamente los 33 MHz. Sin embargo, se decide utilizar una frecuencia de 10 MHz para subir el diseño a la FPGA con el fin de evitar posibles problemas de timing. Esta última decisión es basada en el hecho de que la aplicación final del dispositivo es audio, donde las frecuencias de operación son casi doscientas veces menor.

4. Técnicas de optimización opcionales del módulo

Si bien no es el objetivo del proyecto aplicar técnicas para mejorar el desempeño del dispositivo y verificar su correcto funcionamiento, a continuación se discuten las posibles técnicas que pueden aplicarse a este tipo de sistemas para mejorar su desempeño en los aspectos de timing, área, y potencia.

4.1. Timing

Dado de que se trata de un sistema de procesamiento digital de señal, la presencia de multiplicadores y sumadores es inminente, operaciones como las de MAC son muy comunes en este tipo de sistemas. Sin embargo, como se ha visto, las restricciones de tiempo quedan limitadas en gran parte por este tipo de operaciones. A continuación se muestran técnicas que podrían ser aplicadas al diseño para mejorar el desempeño temporal.

4.1.1. Retiming

El retiming es lo primero que se suele realizar cuando se trata de mejorar. Consiste en a través del teorema de transferencia de registros, desplazar registros dentro del diseño para disminuir el camino crítico. Esta técnica se aplicó al diseño del filtro pero los resultados fueron los mismos.

4.1.2. Pipeling

Otra técnica que suele utilizarse es la de pipeline. Si bien este diseño es un filtro FIR, presenta el comportamiento de un sistema feedback debido a la presencia de la realimentación de la salida para el algortimo LMS. Esto provoca que la tarea de aplicar pipeline al diseño sea nada trivial.

Una propuesta sugerida consiste en aplicar registros de pipeline en la salida la última suma parcial, de esa forma se corta el camino crítico y se puede aumentar el la frecuencia de operación. Sin embargo, se debe tener la precaución de mantener la coherencia en el datapath.

Al agregar registros de pipeline a la señal de error y los registros de desplazamiento se consiguió que el módulo funcione a 55 MHz. El reporte de utilización arrojó prácticamente los mismos resultados que sin pipeline. En la Figura 24 se puede observar la comparación. En él se puede observar un leve incremente en la utilización de FF. Esto es algo esperado debido a la inserción de registros de pipeline.

4.1.3. C-slow

Como se menciona arriba, el sistema se lo puede asimilar más a un sistema feedback que forward. Cuando se trabaja con sistemas feedback, una técnica comúnmente utilizada para mejorar el desempeño temporal del sistema es la de C-slow. Esta técnica permite al filtro trabajar con C señales en paralelo, por lo que es posible hacer que el sistema funcione a una frecuencia más elevada (pero con mayor latencia) y C señales a su entrada. El diseño y propuesta de mejora queda como trabajo a futuro del proyecto.

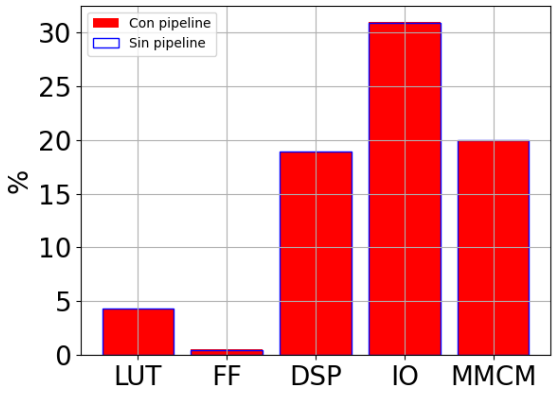


Figura 24: Reporte de utilización con pipeline

4.2. Área

Con una breve descripción de posibles mejoras a aplicar al sistema en cuanto a los aspectos de timing, se continua el análisis de desempeño pero enfocado desde una perspectiva de área.

En este sentido, no existen muchas técnicas populares para aplicar y mejorar el área. Una opción disponible es analizar detalladamente si es posible a través del teorema de transferencia de registros, disminuir la cantidad de registros. Esto impactaría de manera directa en el desempeño temporal del sistema ya que al aplicar un cierto movimiento de registros es posible que aumente el camino crítico del diseño, o en el mejor de los casos, que disminuya. Esto último provoca que no sea una opción muy atractiva.

La variable más propensa a sufrir modificaciones es el número de bits de datos utilizados para representar a las señales en punto fijo. Como se ha visto en la Figura 15, existe una relación de compromiso entre la cantidad de bits utilizados para representar a las señales y el error cuadrático medio que se obtiene al comparar el espectro de la señal en punto flotante.

La propuesta que se realiza es, en caso de presentar un porcentaje de área ocupada mayor a la permitida, disminuir la representación de la señal hasta S(16,15) aceptando una inminente degradación en la señal de salida. Mayores reducciones en el tipo de representación provocan una distorsión fuertemente perceptible al oído humano.

Sin embargo, una opción interesante que se puede barajar es utilizar un valor para μ que sea potencia negativa de 2 (por ejemplo 2^{-4} o 2^{-5}), de esta forma, cuando se multiplique al error por μ habrá que realizar solo un desplazamiento a derecha. Si bien se ahorra área, se pierde en potencia y al mismo tiempo en SNR.

4.3. Potencia

Por último, una opción tentadora para ahorrar potencia de switcheo es la de apagar el algoritmo LMS una vez que los coeficientes llegan a un estado estacionario. Por supuesto, este apagado podría realizarse de manera periódica para que los coeficientes no se alejen mucho de valor óptimo en caso de que la señal de entrada cambie.

5. Conclusiones

Se ha desarrollado de manera exitosa un filtro adaptivo de orden 4 mediante el uso del algoritmo LMS para adaptar sus coeficientes. Por medio del análisis de señales generadas en Python se pudieron generar las señales de estímulo y las memorias que luego serían cargadas en el diseño para su verificación.

Luego de haber corrido diferentes simulaciones se llegó a la conclusión de que la frecuencia máxima de trabajo del mismo no supera los 33 MHz, sin embargo, agregando registros de pipeline la frecuencia puede elevarse hasta 55 MHz, frecuencia la cual se considera más que suficiente para la aplicación para la cual fue diseñado este filtro. Al mismo tiempo, se realizó un análisis del impacto que genera una representación con menos bits fraccionales por medio de la comparación del error cuadrático medio con la señal representada en punto flotante.

Luego de haber diseñado y verificado el diseño en la FPGA por medio de los IP cores VIO e ILA, se proponen distintas técnicas para diseños en los cuales los recursos no son abundantes.

Referencias

[1] K. Shoab, Ahmed, *Digital Design of Signal Processing Systems*, 2nd ed. New Jersey: John Wiley, 2011.