IMPLEMENTACIÓN DE UN SISTEMA GENERADOR Y ELIMINADOR DE ECO CON FILTRO ADAPTATIVO USANDO EL ALGORITMO LMS EN UN FPGA ARTIX-7

IMPLEMENTATION OF AN ECHO GENERATOR AND ELIMINATOR SYSTEM WITH ADAPTIVE FILTER USING LMS ALGORITHM IN AN ARTIX-7 FPGA

Enrique Gerardo Hernández Vega
Tecnológico Nacional de México / Instituto Tecnológico de Chihuahua, México
ehernand@itchihuahua.edu.mx

José Luis Chacón Muñoz
Tecnológico Nacional de México / Instituto Tecnológico de Chihuahua, México
jlchacon@itchihuahua.edu.mx

Jesús Alejandro Navarro Acosta
Tecnológico Nacional de México / Instituto Tecnológico de Chihuahua, México
janavarroa@itchihuahua.edu.mx

Recepción: 21/octubre/2019 **Aceptación:** 23/noviembre/2019

Resumen

En este trabajo se presenta una forma de realizar un procesamiento de señales a través de filtros digitales en sistemas basados en FPGA para dos situaciones distintas: cuando se conoce el modelo a implementar y cuando no se conoce. En este último caso se recurre a los filtros adaptativos, que usan parámetros ajustables basados en un modelo de predicción estadístico. La aplicación es en concreto para el fenómeno de eco, el cual se genera en un FPGA y se elimina en otro. Para la generación se aplicó la ecuación en diferencias, y para la eliminación se usó un filtro adaptativo LMS de orden 40 creado con la herramienta HDL Coder de MATLAB, obteniendo resultados de atenuación de hasta 1/10 de la componente no deseada. El aporte principal de esta investigación es el método de implementación, diseñado con pocos recursos usando la versión gratuita del ambiente de desarrollo Vivado de Xilinx.

Palabras Claves: Eco, filtros adaptativos, FPGA, LMS.

Abstract

In this paper a form of signal processing is presented through digital filters on FPGA based systems for two different situations: when the desired model is known, and when is not; in the latter case, adaptive algorithms are the alternative, which use adjustable parameters that are based on a statistical prediction model. The concrete application is in the echo phenomena, which is generated in one FPGA and eliminated in another. For generation, difference equation is applied, and for cancellation, it was used an 40th order LMS adaptive filter created with HDL Coder tool of MATLAB, getting attenuation results up to 1/10 of the undesired component. The main contribution in this work is the implementation method itself, which is designed to use lower resources, using a free-version of the Xilinx Vivado development environment.

Keywords**:** *Adaptive filters, echo, FPGA, LMS.*

1. Introducción

Los filtros digitales se diseñan en base a ciertos criterios que dependen de la naturaleza de la señal de entrada y de aquello que se desee eliminar. Cuando se conoce esta información, el filtro tendrá un funcionamiento óptimo sin mayor problema. Si esta información no es conocida de antemano, se recurren a los filtros adaptativos; en ellos existen parámetros variables que son ajustados en base a una estimación estadística acorde a la forma en que se presenta la señal a filtrar. De esta manera, los filtros son “adaptables” a un ambiente cambiante o sistema desconocido [Kumudini, 2015].

El diagrama a bloques de un filtro adaptativo se muestra en la figura 1. El bloque consiste en un filtro lineal, un algoritmo adaptativo que altera el comportamiento de este, y la aplicación de la ecuación 1 para encontrar la señal de error.

$$e(n) = d(n) - y(n) \tag{1}$$

Donde:

$y(n)$: Salida del filtro.

$d(n)$: Señal de referencia.

$e(n)$: Señal de error

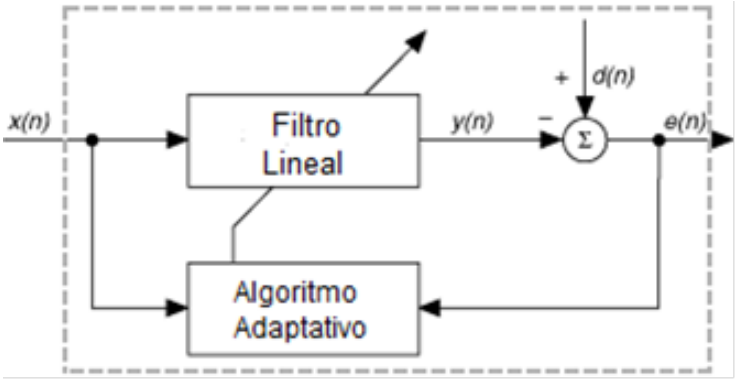


Figura 1 Diagrama a bloques de un filtro adaptativo común.

En un funcionamiento correcto del sistema, la señal de error tiende a hacerse pequeña comparada a las otras dos, puesto que se retroalimenta al sistema para el reajuste de los parámetros internos.

El procesamiento en un filtro adaptativo es computacionalmente costoso comparado con un filtro común, ya que depende del cálculo de estadísticas. Uno de los algoritmos adaptativos más comunes es el de mínimos cuadrados promediados, LMS, también llamado regla de aprendizaje de Widrow-Hoff. Se basa en el gradiente estocástico de los pesos del filtro para converger a una solución óptima de Wiener, el cual es un filtro lineal óptimo en el sentido de que consigue un error cuadrático medio mínimo [Proakis, 2007]. Los pesos ajustables son actualizados con cada iteración tal como se describe en la ecuación 2 [Behrouz, 2013].

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + 2\mu e(n)\boldsymbol{x}(n) \qquad (2)$$

Donde:

$\boldsymbol{x}(n)$: Vector de entradas. $[x(n)\ x(n-1) \ldots x(n-N+1)]$.

$\boldsymbol{w}(n)$: Vector de pesos o coeficientes. $\boldsymbol{w}(n) = [w_0(n)\ w_1(n) \ldots\ w_{N-1}(n)]$

$e(n)$: Señal de error, que se obtiene a partir de la ecuación 1.

μ : Valor de paso.

El valor de paso es una constante que altera la rapidez de convergencia y su estabilidad. Las ecuaciónes 1 y 2 permiten obtener la salida del filtro, que se ajusta continuamente por los pesos y se rige por la ecuación 3 [Behrouz, 2013].

$$y(n) = \boldsymbol{w}^T(n)\boldsymbol{x}(n) \qquad (3)$$

Donde:

$y(n)$: Salida del filtro.

$\boldsymbol{w}(n)$: Matriz de parámetros ajustables o pesos.

$\boldsymbol{x}(n)$: Matriz de entrada actual y entradas pasadas.

Se trata de un algoritmo simple, que no necesita recolectar datos estadísticos. Como se menciona anteriormente, se requiere de plataformas de procesamiento eficientes para la implementación de filtros adaptativos. Los FPGA, al ser reconfigurables, ofrecen la versatilidad de crear sistemas eficaces para aplicaciones en tiempo real que requieran del procesamiento de señales digitales [Safarian, 2015]. Se ha demostrado que el uso de una plataforma de hardware con un FPGA de la familia Virtex-5 de Xillinx para implementar un filtro adaptativo eliminador de ruido provee resultados aceptables al comparar el desempeño del sistema implementado en FPGA y la simulación del modelo ideal del sistema eliminador de ruido, esto enfocado a aplicaciones en el área militar [Bharath, 2012]. También se ha estudiado el desempeño de este tipo de filtros para aplicaciones del área automotriz, utilizando un FPGA de la familia Spartan 3E de Xilinx [Thilagam, 2013].

Otra aplicación de los filtros adaptativos LMS es la cancelación del eco, el cual es el enfoque del presente trabajo. El eco consiste en una única reflexión que retorna al punto original después de emitido el sonido. Para llamar eco al sonido repetido, la separación temporal entre ambos debe ser superior a 100 milisegundos. Las ondas sonoras, al rebotar con un material y reflejarse, sufren además una atenuación. La ecuación en diferencias del eco se muestra en la ecuación 4.

$$y(n) = x(n) + \alpha \cdot x(n-k) \tag{4}$$

Donde:

$y(n)$: Salida del sistema.

$x(n)$: Entrada del sistema.

α : Constante de atenuación del fenómeno.

k : Constante de atraso o desplazamiento.

La aplicación de efectos de audio, tales como la generación del eco, también pueden ser implementados utilizando sistemas basados en FPGA, como en

[Chhetri, 2014]. Una de las formas de evaluar el desempeño de un sistema eliminador de eco es a través del factor "Echo Return Loss Enhancement" o ERLE, donde se aplica la ecuación 5.

$$ERLE(n) = 10\log_{10}\left(\frac{E\left(\boldsymbol{e}^2(n)\right)}{E\left(\boldsymbol{d}^2(n)\right)}\right) \quad (5)$$

Donde:

$\boldsymbol{e}(n)$: Vector de la señal de error del valor actual y valores pasados.

$\boldsymbol{d}(n)$: Vector de la señal de referencia del valor actual y valores pasados.

El operador $E()$ se refiere al termino de esperanza matemática y es a efectos prácticos, un promediado [Yemdji, 2010].

En [Homana, 2011] el sistema eliminador de eco se ha implementado en un FPGA de la familia Virtex 5 de Xilinx, obteniendo resultados favorables, basados en los valores del factor ERLE obtenidos.

También, se ha implementado este sistema en un FPGA Cyclone II de Altera [Pujari, 2014]. En ambos casos, las señales de entrada y de referencia son generadas utilizando un software computacional. En este trabajo se añade un sistema generador de eco a una señal de audio, para eliminar la necesidad de utilizar un software para producir las señales de entrada del sistema, usando un FPGA Artix-7 7A35T de Xilinx embebido en tarjeta Arty.

2. Métodos

El sistema diseñado consiste en dos bloques principales sobre los cuales se implementan la generación y eliminación de eco, respectivamente. Tal como se muestra en la figura 2, el primer bloque refiere a un FPGA que genera una señal de salida con eco que resulta de aplicar la ecuación en diferencias de este fenómeno a una señal analógica de entrada d(n), con las constantes de α y k del fenómeno controladas por el usuario; por otra parte, el bloque siguiente utiliza un segundo FPGA que aplica el filtro adaptativo en base a dos entradas, la señal analógica original que será la señal de base o referencia, y la señal con eco $x(n)$, a la que se le desea eliminar este fenómeno. La señal $y(n)$ se define como la salida del sistema.

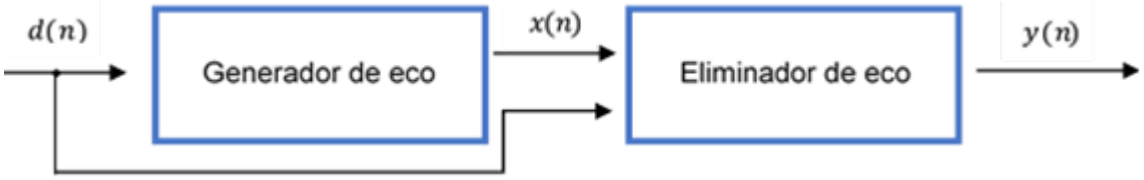


Figura 2 Diagrama a bloques genérico del sistema.

Para realizar estos procesamientos a las señales analógicas en ambos bloques, es necesario convertirlas a su equivalente digital, actuar sobre ellas y regresarlas a una forma analógica; por lo que en cada uno de estos bloques principales se usa un convertidor analógico-digital, el XADC, para la adquisición de la señal y un convertidor digital-analógico, el PmodDA2, para reconstruir la señal resultante después de procesarla. Ambos convertidores tienen una resolución de 12 bits, con una frecuencia de muestreo de 40 kHz, debido a la naturaleza de las señales.

La generación de eco implica implementar en el primer FPGA la ecuación en diferencias descrita en la ecuación 4, de modo que se requiere de memoria para poder tomar valores anteriores de la entrada, así como también de una multiplicación por una constante y una suma. El generador de eco se diseñó para tres posibles constantes de desplazamiento, k, de 0.25, 0.50 y 0.75 segundos, así como dos constantes de atenuación α de 0.5 y 0.25.

La cantidad de memoria requerida depende del desplazamiento del eco que se desee aplicar según los tiempos descritos.

Dado que la frecuencia de muestreo es de 40,000 muestras por segundo estas constantes se traducen a 10,000, 20,000, y 30,000 muestras a guardar en memoria para realizar los desplazamientos. El manejo de la memoria se realizó a través de un núcleo de propiedad intelectual, IP, llamado “Block memory generator” que se configura para una capacidad de 30,000 datos de 16 bits en memoria RAM, con puerto siempre habilitado y en modo de operación “No change”. Con esta configuración, la escritura y lectura en memoria se realizó en base al diagrama de tiempos que se observa en la figura 3.

El esquema para aplicar el desplazamiento del eco se muestra en la figura 4; en el momento de la escritura en la dirección ‘n’ del dato en tiempo actual, en la dirección ‘n+1’ se encuentra el dato que se escribió hace N muestras.

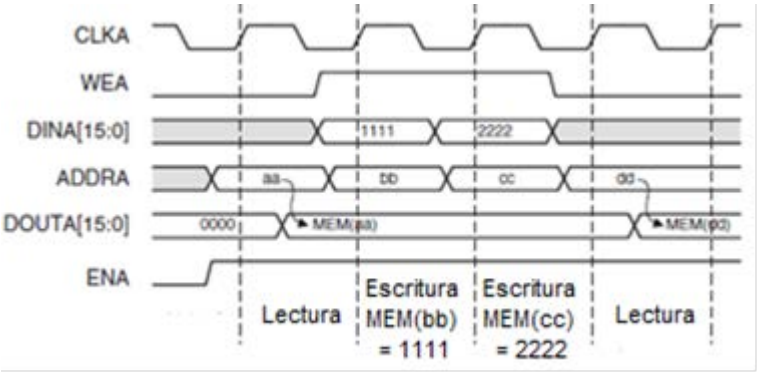


Figura 3 Diagrama de tiempos de lectura y escritura del bloque de memoria.

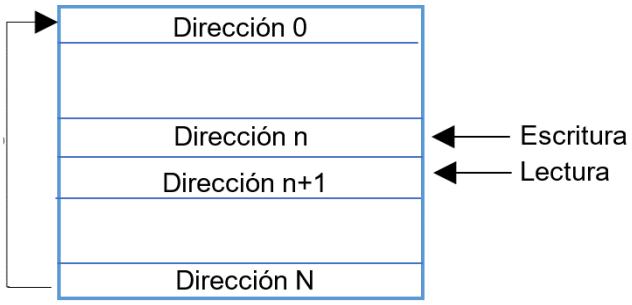


Figura 4 Escritura de muestras actuales y lectura de muestras pasadas.

El procedimiento de lectura y escritura antes mencionado se implementó con una Máquina de Estados Finito, FSM, cuyo ciclo de lectura/escritura está sincronizado a la adquisición del dato por el XADC, que tiene una señal llamada “end of conversion” que manda un pulso en el momento de adquirir el dato para cualquier modo de operación. Para la selección de los tres posibles desplazamientos descritos, se colocan tres contadores que funcionan en los rangos de dirección correspondientes conectados a un multiplexor, el cual asigna el contador correcto de acuerdo a una entrada definida para tal propósito.

La multiplicación, que se realiza a la muestra desplazada por un factor de atenuación entre los valores posibles de 0.5 y 0.25, se llevó a cabo mediante un bloque que realiza un corrimiento a la derecha de uno o dos bits dependiendo de la opción definida por una entrada del sistema, resultando en una división de dato entre dos o cuatro que son equivalentes a los factores mencionados.

Para la suma de la entrada actual y la desplazada con un factor de atenuación se utilizó la IP “adder/substracter” configurada en modo de suma, para dos vectores de

12 bits de tipo de dato con signo y cuya salida se recorta a 12 bits. La salida de este bloque es $y(n)$ de la ecuación 4 y se aplica como entrada de la FSM que realiza la transmisión de datos al DAC.

La señal de eco generada en el primer FPGA y la señal original se toman como entradas en un segundo FPGA para aplicar el filtro adaptativo LMS para producir la salida final del sistema $y(n)$ tal como se mostró en la figura 2. Se trató el algoritmo como un único bloque donde se implementan las ecuaciones 1 y 2 con el valor de paso μ como entrada al sistema; se hizo de esta manera para observar el comportamiento de la señal de salida ante distintos valores de este parámetro. El XADC en este segundo bloque eliminador de eco, usa un modo simultáneo en el que, al mismo tiempo, captura las dos señales analógicas, de entrada ($x(n)$) y de referencia ($d(n)$), a través de 2 canales distintos. Sin embargo, solo existe un único bus de salida de los datos, por lo que se escriben estos datos en dos registros de memoria por separado cuyas salidas se conectan al bloque que aplicará el algoritmo. Se implementó una FSM para este punto. Para el diseño del bloque de aplicación del algoritmo se utilizó la herramienta de MATLAB llamada HDL Coder para obtener un archivo HDL del bloque LMS que se puede tomar de la herramienta de Simulink del mismo programa, tal como se realizó en [Manalo, 2012], tomando el ejemplo “hdlcoderlms”, donde se observa un diagrama como el de la figura 5.

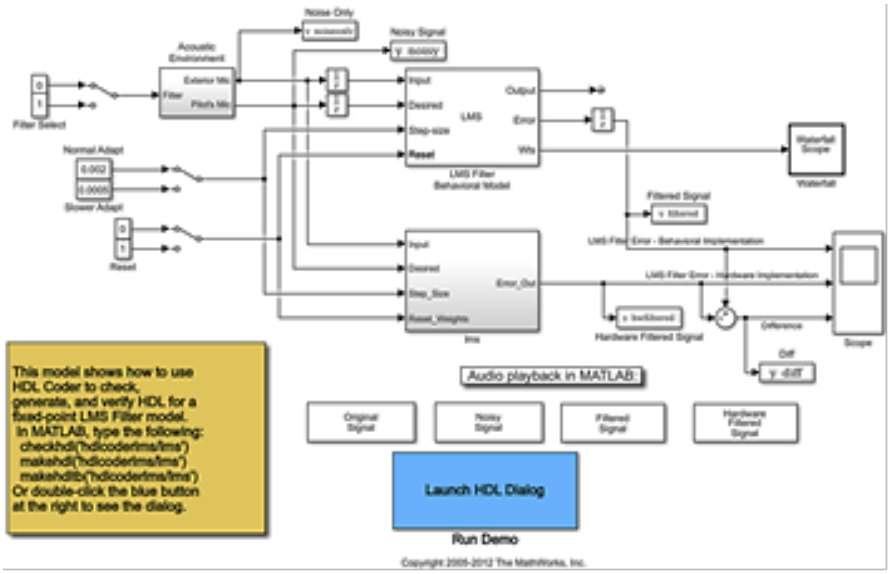


Figura 5 Ejemplo del algoritmo LMS para cancelación de ruido en MATLAB.

El bloque sobre el que se quiere generar la descripción de hardware es el llamado “lms”. Antes de generar el archivo HDL del bloque, se realizan las siguientes modificaciones al ejemplo (figura 6):

- Se cambia, para cada entrada del bloque de interés, el periodo de muestreo de 1/8000 segundos a 1/40000 segundos, puesto que ese fue el periodo planteado en el análisis. Esto se hace dando doble click en las entradas del bloque sin importar el tipo de dato y se cambia este valor como se observa en la figura 6.

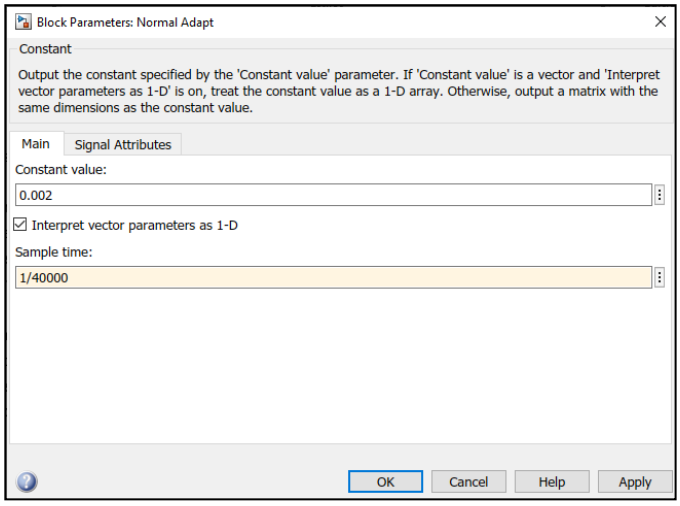
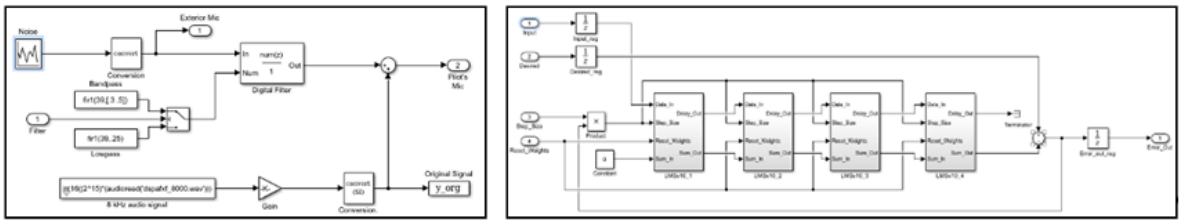


Figura 6 Modificación del periodo de muestreo de las entradas.

- Se modifica el tipo de dato de la señal en el bloque “Conversion”, que se observa al dar doble click en el bloque de “Acoustic Environment” por un formato fixdt(1,12,9), pues los convertidores a implementar son de 12 bits. Lo mismo para el punto de suma que se observa a la derecha al dar doble click al bloque ‘lms’. Ambas vistas de los bloques se muestran en la figura 7, mientras que la ventana de modificación se muestra en la figura 8.

Tras realizar estas modificaciones se generaron los distintos archivos HDL que conforman el bloque y se agregaron al proyecto para su implementación en el segundo FPGA. El bloque obtenido para implementar el algoritmo se muestra en la figura 9.



a) Modificación en bloque de conversión *b) Modificación en punto de suma*

Figura 7 Bloques donde se deben cambiar el tipo de dato a manejar.

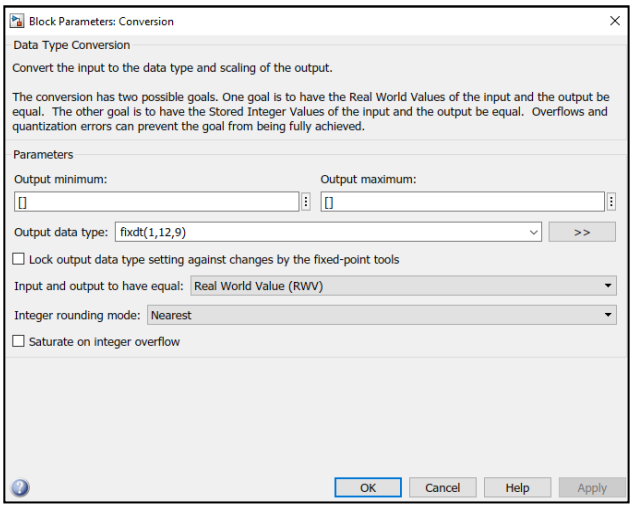


Figura 8 Modificación del tipo de dato fixdt(1,12,9).

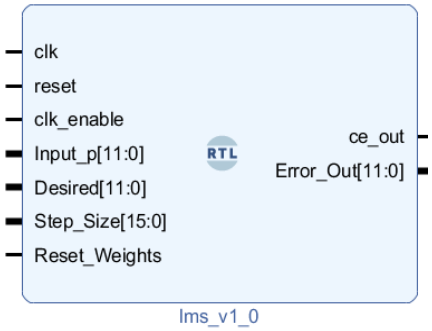


Figura 9 Bloque LMS visto desde un diagrama a bloques.

Para poder cambiar el valor de paso μ a usar dentro del bloque del algoritmo LMS, se implementó una interfaz con un contador de 16 bits de pulsaciones de un botón con tamaño de paso de cuatro y cuya operación puede ser de incremento o decremento de este al valor anterior del contador que depende de un switch. El valor actual del contador es el que se introduce como μ al bloque que implementa el

algoritmo. Para el control del botón se diseñó una descripción de hardware anti-rebote. La ecuación 6 define el modo de trabajo de la interfaz:

$$\mu(p) = 4p + 2 \tag{6}$$

Donde:

$\mu(p)$: Valor de paso.

p : Incrementos menos decrementos en el tiempo.

La inclusión de la constante evita que el resultado pueda ser cero. El valor de p tiene la restricción de que este valor debe ser mayor o igual a cero. Para conocer el valor de u(p), se mostraba el número de incrementos o decrementos que el valor tenía con respecto al valor inicial en dos pantallas de 7 segmentos, con un “00” en condiciones iniciales. Para mostrar los dos dígitos al mismo tiempo, se elaboró un bloque divisor de frecuencia de la señal de reloj que sirva como señal de multiplexeo de ambas pantallas. Para la elaboración de los diseños completos de generación y eliminación de eco en cada FPGA, se utilizó el ambiente Vivado de Xilinx, específicamente la versión 2018.3 en su edición WebPACK, que es gratuita. La herramienta llamada IP Integrator disponible en este software, que permite diseñar el sistema por bloques a través de instanciaciones de componentes y sus respectivas interconexiones, fue la herramienta escogida para el desarrollo de la arquitectura e implementación de ambos sistemas, por permitir un diseño despegado del hardware que reduce razonablemente los tiempos de desarrollo.

Debido a la capacidad limitada de las herramientas usadas (versión libre) y la complejidad de los sistemas diseñados, una simplificación de cada uno se observa en las figuras 10 y 11. Se han omitido los bloques de tipo “slice” usados como adaptadores entre los bloques que manipulan señales de 12 bits, como los sumadores, y los que lo necesitan de 16 bits, como los bloques de memoria, así mismo se omitieron los bloques de conversión a complemento a dos, usados para linealizar la salida del XADC. Se utilizó la IP llamada “Clock Wizard” para generar las distintas señales de reloj de los demás bloques, siendo de 26 MHz para el XADC, de 17 MHz para el módulo PmodDA2, y se definió arbitrariamente una señal de reloj de 13 MHz para los bloques relacionados a la memoria, aplicándose en ambos

diseños para que cada uno funcione a 40,000 muestras por segundo. La interconexión física entre los dos sistemas se muestra en la figura 12, donde los bloques principales son conformados por el conjunto XADC/FPGA/DAC.

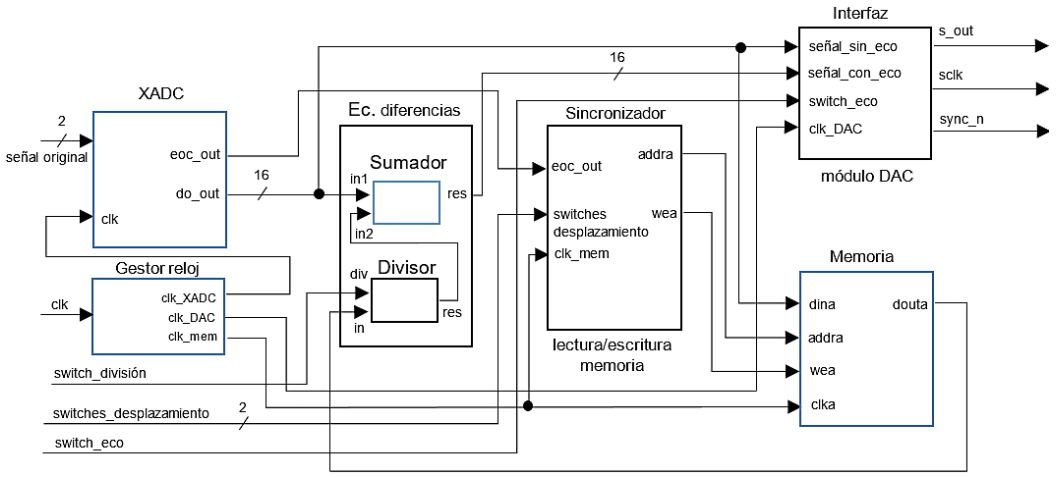


Figura 10 Diagrama a bloques simplificado del sistema generador de eco.

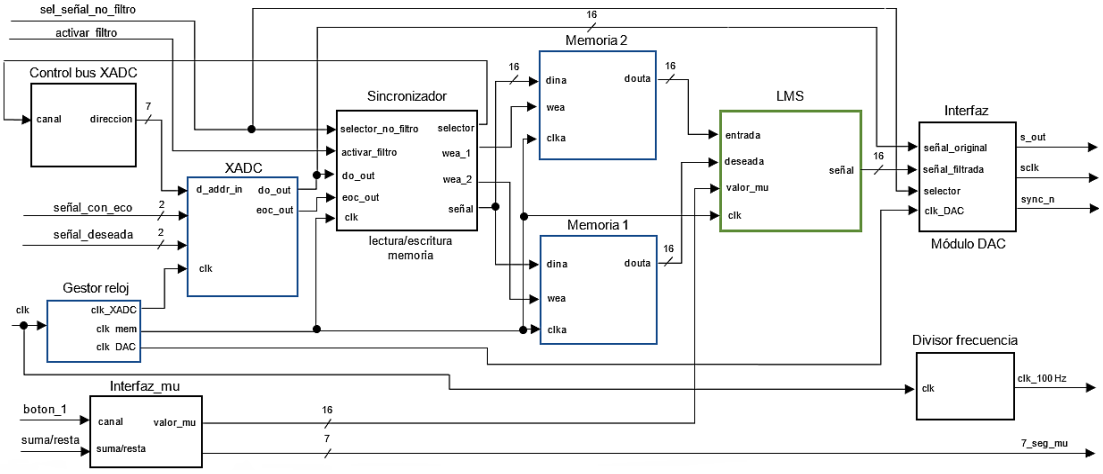


Figura 11 Diagrama a bloques simplificado del sistema eliminador de eco en el FPGA.

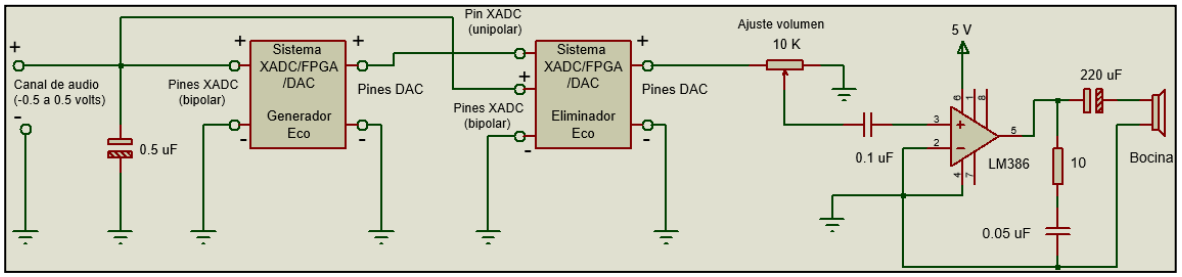


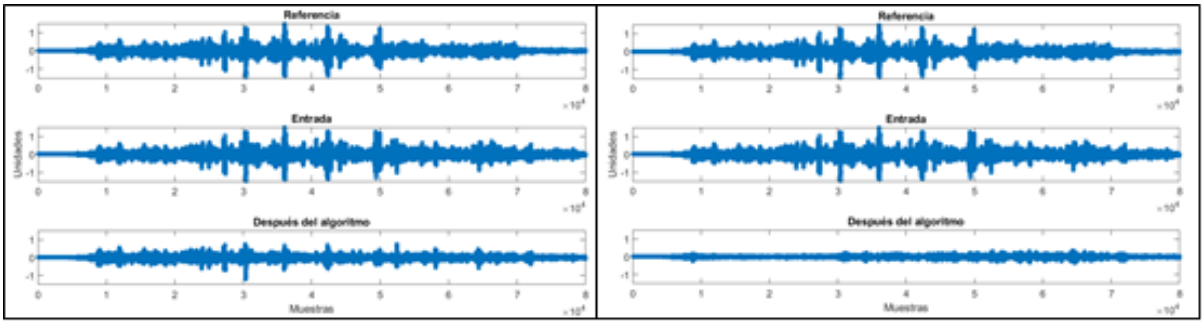
Figura 12 Sistema generador – eliminador de eco con acondicionamiento de la señal.

La señal de audio es obtenida de un cable auxiliar conectado a un sistema de audio. Las interconexiones entre ambos FPGA no requieren de ningún acondicionamiento, como se mencionó anteriormente, de manera que las escalas no implican un problema mientras se convierta a complemento a dos alguno de ellos al momento de manipular ambas señales (con y sin eco) al mismo tiempo. La última parte del diagrama refiere a un circuito amplificador con control de volumen.

El capacitor que se encuentra después del potenciómetro funge como eliminador de componentes de corriente directa ya que la señal se montó en 1.67 volts a causa de la naturaleza del DAC, y lo que se observa después es un circuito amplificador y un filtro pasivo en una configuración clásica dentro de la literatura electrónica, donde finalmente se conecta una bocina.

3. Resultados

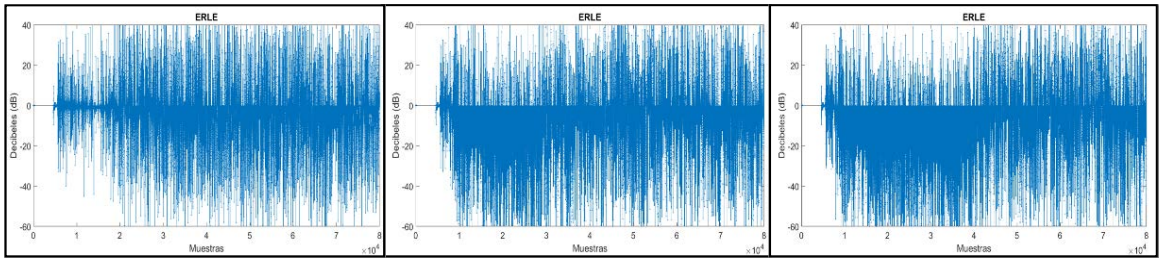
Se realizó la simulación del bloque LMS que se obtuvo por medio de la herramienta de Simulink y fue sometido a pruebas con distintas señales de audio y su respectiva señal con eco, que se generaba aplicando la ecuación 4 en un programa desarrollado en MATLAB. El ejemplo fue probado por una señal de audio de una duración de dos segundos. La figura 13 muestra las señales de entrada del bloque del audio con eco x(n), el audio sin eco d(n) y la señal de error e(n) de la ecuación 1, para algunos ejemplos de las señales de prueba resultantes. Se puede observar en figura 13b como se presenta una señal de error generalmente más pequeña.



a) Desplazamiento de 0.50 segundos, $u(p)$ = 2 *b) Desplazamiento de 0.50 segundos, $u(p)$ = 98*

Figura 13 Pruebas de la simulación con distintos desplazamientos y valores de μ.

Usando la ecuación 5 descrita anteriormente, para una ventana de 250 muestras, la señal ERLE presentó un rango de [-55, 40] decibeles aproximadamente, tal como se muestra en la figura 14, donde se comparan los distintos parámetros.



a) 0.250 segundos u(p) = 2. *b) 0.500 segundos; u(p) = 98.* *c) 0.750 segundos; u(p) = 130*

Figura 14 Señal ERLE para distintos desplazamientos y valores de μ en simulación.

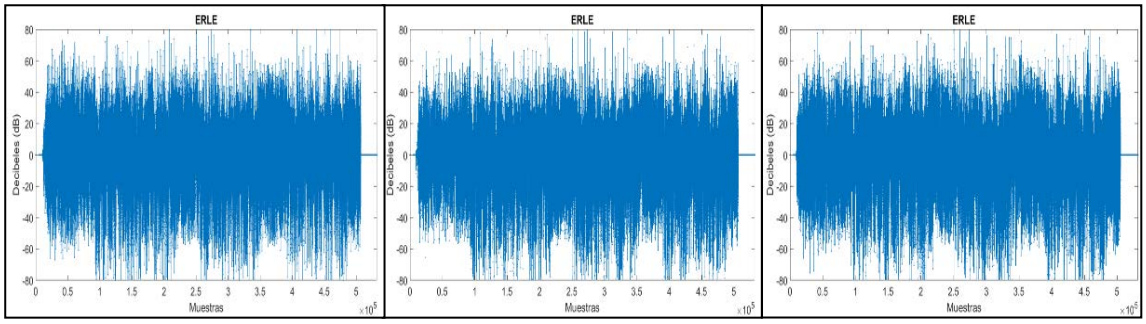
Al implementar el sistema físico completo, experimentando con distintas señales y haciendo uso de las opciones desarrolladas para el sistema generador de eco en términos de su desplazamiento y atenuación, se obtuvieron, para una constante de atenuación α de 0.5, los siguientes resultados, que fueron grabados con un sistema de mediana calidad y finalmente analizados en MATLAB, siendo resultados de la implementación. En el sistema eliminador de eco, el valor de paso μ que estaba descrito por la ecuación 6 y era variable en relación a la interfaz aplicada para modificar el valor de paso μ, resultó tener un rango de valores para los cuales el eco se reducía y además la señal mantenía su inteligibilidad.

El valor que se considera para observar la energía total de la señal de error a lo largo de la señal es el “Root Mean Square” o valor RMS cuya técnica considera valores positivos y negativos sin que se eliminen. La tabla 1 muestra los resultados más significativos para los cuales la interfaz marcaba los valores de u(p) y su RMS normalizado. El porcentaje más eficiente de error recae en el rango 34 a 98 del valor de u(p), siendo el más bajo de 10.57%.

A su vez, usando la misma ecuación 5 descrita para obtener el factor ERLE, para una ventana de 250 muestras, el rango perteneciente de estas señales para los tres posibles desplazamientos es de [-60, 50] decibeles, tal como se observa en la figura 15, donde se muestra el espectro para un mismo valor de paso μ, pero con distintos desplazamientos, y que, de manera práctica, tienen poca diferencia entre ellos.

Tabla 1 Valores de paso y valor RMS normalizado del error.

n	$u(p)$	$u(p)_b$	$\frac{RMS(e(n))}{RMS(d(n))} * 100\%$		
			0.250 segundos	0.500 segundos	0.750 segundos
00	2	0000 0000 0000 0010	11.30%	13.92%	13.53%
08	34	0000 0000 0010 0010	11.90%	11.84%	10.57%
16	66	0000 0000 0100 0010	11.95%	11.90%	11.75%
24	98	0000 0000 0110 0010	11.36%	10.96%	11.31%
32	130	0000 0000 1000 0010	20.85%	14.09%	15.73%
40	162	0000 0000 1010 0010	17.15%	17.22%	17.54%
48	194	0000 0000 1100 0010	19.01%	17.91%	18.90%
56	226	0000 0000 1110 0010	19.20%	17.70%	19.80%



a) 0.250 segundos *b) 0.500 segundos* *c) 0.750 segundos*

Figura 15 Señal ERLE para los distintos desplazamientos y u(p) = 98.

Los recursos utilizados para la implementación del sistema, tal como se observa en la figura 16, son razonablemente mayores para el diseño del sistema eliminador que para el generador de eco. El FPGA Artix-7 usado (7A35T) cuenta con 90 unidades DSP48 y que son utilizados por los distintos núcleos IP basados en ellos; el sistema generador utilizó apenas 4, sin embargo, el sistema eliminador utilizó todos los disponibles, debido a su mayor complejidad interna, así como una cantidad considerable de bloques LUT.

Resource	Utilization	Available	Utilization %
LUT	107	20800	0.51
FF	73	41600	0.18
BRAM	15	50	30.00
DSP	4	90	4.44
IO	12	210	5.71
BUFG	4	32	12.50
MMCM	1	5	20.00

Resource	Utilization	Available	Utilization %
LUT	11401	20800	54.81
FF	1307	41600	3.14
BRAM	1	50	2.00
DSP	90	90	100.00
IO	23	210	10.95
BUFG	5	32	15.63
MMCM	1	5	20.00

a) Sistema generador de eco. *b) Sistema eliminador de eco.*

Figura 16 Recursos usados para cada sistema.

4. Discusión

En el presente trabajo se presenta una implementación de un generador de eco desarrollado por ecuación en diferencias y un eliminador del mismo a través de un filtro adaptativo bajo el algoritmo LMS con resultados de atenuación de hasta un décimo de la componente indeseada de la señal usando como plataformas a los FPGA. Sin embargo, cabe mencionar que se realizó bajo medidas de recursos y cálculos principalmente bajos, pues se trata de un filtro de orden 40 cuyas características podrían mejorarse considerablemente si se usara un filtro de un orden mayor, y que en parte la propia aplicación lo estaría demandando, ya que el eco producido planteado fue de hasta 0.75 segundos, que equivalen a 30,000 muestras de desplazamiento a la frecuencia de muestreo manejada, la cual no se puede bajar a causa de una perdida de fidelidad del audio. En este sentido, es importante mencionar que el FPGA dedicado a la eliminación del eco agotó sus recursos de tipo DSP de 90 unidades posibles, lo cual resultó ser una limitante en la búsqueda de un filtro de mayor orden tal y como se propuso el diseño. Por otra parte, el algoritmo propuesto LMS, es el más sencillo de todos y sobre el cual se basan el resto de algoritmos de media cuadrática; sin embargo, su sencillez en computación le da una predicción de la señal menos eficiente a los demás algoritmos y si se buscara una mayor cancelación de eco esta sería otra alternativa a considerar. En cuanto al rango del factor ERLE resultante del sistema, este tuvo variación en la implementación con respecto a lo que arrojó la simulación (de hasta 10 decibeles), pero finalmente esto es en parte a las posibles modificaciones de la señal al ser grabada en condiciones imperfectas. En cualquier caso, el rango que abarca este factor es más ancho que la mayoría de los sistemas adaptativos, lo cual es un problema a causa del rango superior de la banda. Tal como se ha propuesto en este trabajo, y considerando la necesidad de encontrar un valor adecuado al valor de paso μ, el sistema puede implementarse en aplicaciones sencillas.

5. Conclusiones

El sistema generador y eliminador de eco implementado en sistemas FPGA es un sistema completo con el objetivo de demostrar las capacidades de estas

plataformas para filtros digitales cuyo modelo se conoce (generación de eco a través de su ecuación en diferencias) o bien no es posible conocer, en cuyo caso se puede hacer uso de un filtro adaptativo, que representa una mejora considerable a la aplicación de filtros digitales dedicados a fenómenos más complejos de desarrollar por falta de información o identificación del modelo propio. En este caso, la aplicación del filtro adaptativo LMS, uno de los algoritmos más simples y con poco requerimiento de recursos, resultó de utilidad en aplicaciones sencillas, donde se demostró un factor ERLE de valores de hasta -55 decibeles y un RMS normalizado de hasta 1/10, habiendo previamente sintonizado un valor de paso μ correcto. Sin embargo, de ser necesaria una mayor calidad de cancelación, las alternativas son cambiar el algoritmo adaptativo a uno más complejo, como el NLMS, o también, utilizar un FPGA con mejores recursos buscando un mayor orden del filtro (en el caso de las unidades de tipo DSP, existen dispositivos FPGA de serie 7 de Xilinx que pueden tener hasta 2,520, lo cual posibilitaría un mayor orden), o bien cambiar a una plataforma más orientada a ese tipo de aplicaciones, como los DSP. Sin embargo, es importante mencionar que los FPGA son una opción a considerar actualmente gracias a su versatilidad para hacer sistemas a la medida y por su estructura interna que brinda sistemas de mayor precisión y con mayor capacidad de actuar sobre los datos de una manera paralela en aplicaciones con gran cantidad de ellos. A pesar de usar recursos de hardware y software relativamente limitados, la forma en que se llevó a cabo este procedimiento es instructiva en temas de desarrollo de sistemas basados en FPGA, por lo que este trabajo se puede usar para fines educativos, así como también tiene una posible trascendencia como base para plantear los cambios antes mencionados en investigaciones posteriores a favor de una mejor calidad de eliminación de eco, habiendo considerado que cada uno de estos puntos tiene sus ventajas y desventajas.

6. Bibliografía y Referencias

[1] Chhetri, S.R., Poudel, B., Ghimire, S., Shresthamali, S., & Sharma, D.K. Implementation of Audio Effect Generator in FPGA. Nepal Journal of Science and Technology, ISSN: 1994-1412, Vol. 15, No.1, 89-98, 2015.

[2] Behrouz, F.B. Adaptive Filters. Theory and Application, 139-141. Wiley. USA. 2013.

[3] Bharath, K.S., Ara, A., Ramani, N., Bindu, K., & Hegde, R. Adaptive Noise Cancellation Filter Using LMS Algorithm on an FPGA for Military Applications. International Journal of Knowledge Engineering, ISSN: 0976-5816 & E-ISSN: 0976-5824, Vol. 3, No. 2, 207-211, 2012.

[4] Homana, I., Muresan, I., Topa, M., & Contan, C. FPGA Implementation of an Acoustic Echo Canceller. ACTA TECHNICA NAPOCENSIS: Electronics and Telecommunications, ISSN: 1221-6542, Vol. 52, No. 2, 44-47, 2011.

[5] Kumudini, S., & Rahul, S. Simulation of NLMS Adaptive Filter for Noise Cancellation. International Journal of Engineering and Applied Sciences (IJEAS), ISSN: 2394-3661, Vol. 2, No. 7, 27-29, 2015.

[6] Manalo, M., & Ashrafi, A. Implementing Filters on FPGAs. Department of Electrical and Computer Engineering Real-Time DSP and FPGA Development Lab. San Diego. USA. 2012.

[7] Pujari, S.S., Panda, A., & Dash, P.K. Design & Implementation of FPGA based Adaptive Filter for Echo Cancellation. International Conference on Convergence of Technology, IEEE, 2014.

[8] Proakis, J. G., & Manolakis, D. G. Tratamiento digital de señales, 804-815. Pearson Education. Mexico. 2007.

[9] Safarian, C., Ogunfunmi, T., Kozacky, W.J., & Mohanty, B.K. FPGA Implementation of LMS-based FIR Adaptive Filter for Real Time Digital Signal Processing Applications. IEEE International Conference on Digital Signal Processing (DSP), ISSN: 1546-1874 & E-ISSN: 2165-3577, Singapore, 1251-1255, 2015.

[10] Thilagam, S. Efficient Implementation of Adaptive Noise Canceller Using FPGA for Automobile Applications. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, ISSN: 2320–3765 & E-ISSN: 2278–8875, Vol. 2, No. 12, 6218-6224, 2013.

[11] Yemdji, C., Mossi Idrissa, M., Evans, N. W., & Beaugeant, C. Efficient Low Delay Filtering For Residual Echo Supression. Dinamarca. IEEE. 2010.