Revista EIA, ISSN 1794-1237 / Año XI / Volumen 11 / Edición N.22 / Julio-diciembre 2014 / pp. 133-146
Publicación semestral de carácter técnico-científico / Escuela de Ingeniería de Antioquia —EIA—, Envigado (Colombia)

DISEÑO DE UN MICROSISTEMA PROGRAMABLE PARA EFECTOS DE AUDIO DIGITAL USANDO FPGAS

John Michael Espinosa Durán[1]
Pedro P. Liévano Torres[2]
Claudia P. Rentería Mejía[3]
Jaime Velasco Medina[4]

RESUMEN

Este artículo describe el diseño de un microsistema programable para el procesamiento de efectos de audio digital implementado en un FPGA. El microsistema es diseñado usando un procesador de propósito específico y reconfigurable, un banco de RAMs y una interfaz gráfica de usuario basada en una pantalla táctil LCD. El procesador es diseñado usando 15 efectos de audio basados en retardos y procesamiento en el dominio dinámico y de la frecuencia. Los efectos son diseñados usando Megafunciones y el compilador FIR de Quartus II, son simulados en Simulink[5] usando DSP Builder[6], y son configurados utilizando una interfaz gráfica de usuario. El microsistema programable es implementado en el sistema de desarrollo DE2-70, y su funcionamiento es verificado usando un reproductor MP3 y un parlante. Adicionalmente, el microsistema permite la generación de efectos con alta fidelidad usando una tasa de muestreo máxima de 195.62 MSPS, y puede ser embebido en un SoC.

PALABRAS CLAVES: efectos de audio digital; dominio dinámico; dominio de la frecuencia; sistemas embebidos; FPGAs.

DESIGN OF A PROGRAMMABLE MICROSYSTEM FOR DIGITAL AUDIO EFFECTS USING FPGAS

ABSTRACT

This paper describes the design of a programmable microsystem for processing of digital audio effects implemented in an FPGA. The microsystem is designed using an specific purpose and reconfigurable processor, a bank of RAMs and a user graphic interface based on an LCD touch panel. The processor is designed using 15 audio effects based on delays, and dynamic and frequency domain processing. The effects are designed using Megafunctions and the FIR compiler of Quartus II, simulated in Simulink using DSP Builder, and configured using a user graphic interface. The programmable

1 Ingeniero electrónico, MsC. en Ingeniería, Universidad del Valle. PhD. (c) en Química. Profesor Asociado en Universidad de Indiana, Bloomington, Estados Unidos.
2 Ingeniero electrónico, Universidad del Valle. Coordinador de proyectos y desarrollo en IP Innovatech, Cali, Colombia.
3 Ingeniera electrónica, Universidad del Valle. Magíster en Ingeniería, Universidad del Valle. Estudiante de Doctorado en Ingeniería en Universidad del Valle, Cali, Colombia.
4 Ingeniero electricista, Universidad del Valle. Ph.D. en Microelectrónica, Instituto Nacional Politécnico de Grenoble, Grenoble, Francia. Profesor Titular y director del grupo de investigación en Bionanoelectrónica, Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Colombia.

Autor de correspondencia: Velasco-Medina, J. (*Jaime*).
Calle 13 N. 100 - 00, Universidad del Valle, Cali (Valle, Colombia). Tel: (572) 330 34 36.
Correo electrónico: jaime.velasco@correounivalle.edu.co

Historia del artículo:
Artículo recibido: 8-X-2013 / Aprobado: 17-III-2014
Disponible online: 30 de diciembre de 2014
Discusión abierta hasta diciembre de 2015

DOI: http:/dx.doi.org/10.14508/reia.2014.11.22.133-146

microsystem is implemented on the DE2-70 development board, and its operation is verified using an MP3 player and a speaker. Additionally, the microsystem allows the generation of effects with high fidelity using a maximum sample rate of 195.62 MSPS, and can be embedded into a SoC.

KEYWORDS: Digital Audio Effects; Dynamic Domain; Frequency Domain; Embedded Systems; FPGAs.

PROJETO DE UM MICROSISTEMA PROGRAMÁVEL PARA EFEITOS DE ÁUDIO DIGITAL USANDO FPGAS

RESUMO

Este artigo descreve o desenho de um microsistema programável para o processamento de efeitos de auidio digital implementado em um FPGA. O microssistema é projetado usando um processador específico e reconfigurável um banco de RAMs e uma interface gráfica de usuario baseada em uma tela sensível ao toque de LCD. O processador foi projetado com 15 efeitos de áudio com base em atrasos e domínio de processamento e frequência dinâmica. Os efeitos são projetados usando Megafunciones e o compilador FIR de Quartus II são simulados utilizando Simulink 1 usando DSP Builder2 e são configurados através de uma interface gráfica de usuario. O microssistema programável é implementado no sistema de desenvolvimento DE2-70, e seu desempenho é verificado através de um leitor de MP3 e um alto-falante. Além disso, o micro-sistema permite a geração de efeitos, com elevada fidelidade, utilizando uma taxa de amostragem máxima de 195,62 MSPS, e pode ser incorporado em um SoC.

PALAVRAS-CHAVE: Efeitos de áudio digital, De domínio dinâmico, Domínio da freqüência, Sistemas embarcados, FPGAs.

1. INTRODUCCIÓN

Los efectos de audio son usados por los músicos para generar sonidos especiales al interpretar un instrumento. Estos efectos pueden ser generados usando sistemas de procesamiento analógico o digital. Actualmente los efectos de audio digital son más usados y son implementados con sistemas electrónicos que llevan a cabo procesamiento basado en retardos, o procesamiento en el dominio dinámico o de la frecuencia (Zölzer, 2002).

Los efectos de audio digital generalmente son implementados en DSPs, PCs ó GPUs (Berdahl y Smith, 2006; Fernández y Casajus, 2000; Guillermard, Ruwwue y Zölzer, 2005; Karjalainen, Penttinen y Valimaki, 2000; Ling, Khuen y Radhakrishnan, 2000; Oboril, e*t al.*, 2000; Schimmel, Smekal y Krkavec, 2002; Tsai, Wang y Su, 2010; Verfaille, Zölzer y Arfid, 2006). Sin embargo, los diseños presentados en los trabajos anteriores no tienen altas tasas de muestreo y la mayoría de los efectos son basados en retardos. En Pfaff, *et al.* (2007), los autores implementan los efectos *chorus*, *delay*, *echo cancellation*, *flanger* y *wah-wah* usando co-diseño *Hardware-Software*. El *flanger* se implementó usando una *look-up table* para generar la señal sinusoidal y el *wah-wah* usando un filtro paso-todo de segundo orden y una *look-up table* para variar la frecuencia de corte, pero en ese trabajo no se presentan las pruebas de verificación en *hardware* de los diseños. En Byun, *et al*. (2009) los autores implementan los efectos *reverb*, *chorus*, *flanger*, *phaser*, *tremolo*, *auto wah*, *pitch shift*, *distortion* y *multi-band equalizer* usando lenguaje C y un DSP embebido en un FPGA. Sin embargo, este artículo no describe en detalle los algoritmos usados para implementar estos efectos y no presenta las pruebas de verificación de los diseños. En un trabajo previo (Liévano, Espinosa y Velasco, 2013) se implementaron 14 efectos de audio usando procesamiento basado en retardos (5) y procesamiento en el dominio dinámico (8) y de la frecuencia (1). Estos efectos son descritos en VHDL, simulados en Simulink usando DSP Builder y sintetizados en un FPGA usando Quartus II.

Teniendo en cuenta lo anterior, en este trabajo se presenta el diseño de un microsistema para procesar efectos de audio digital, el cual es basado en un pro-

cesador dedicado y configurable. El microsistema fue implementado en el sistema de desarrollo DE2-70, y el procesador fue sintetizado en el FPGA de esta tarjeta.

La principal contribución de este trabajo es el diseño de un procesador de efectos de audio digital en tiempo real y con muy alta fidelidad debido a que tiene una tasa de muestreo de 195,62 MSPS usando datos de 16 bits. La tasa de muestreo corresponde a la frecuencia máxima de operación del procesador (195.62 MHz). Adicionalmente, el procesador diseñado permite generar efectos de audio no muy comunes, debido a que es posible mezclar efectos de diferente y/o el mismo tipo de procesamiento. En este procesador, se implementaron 15 efectos de audio, 14 de los cuales fueron presentados en Liévano, Espinosa y Velasco (2013). Sin embargo, los efectos *chorus*, *reverb* y *wah-wah* fueron rediseñados y se adicionó un *multi-band equalizer*. Las pruebas de verificación del microsistema se realizaron usando un reproductor MP3 y un parlante conectados en la entrada y la salida de audio de la tarjeta DE2-70, respectivamente.

Este artículo está organizado de la siguiente manera: en la sección 2 se describe la metodología para el diseño de los efectos *chorus, reverb, wah-wah* y *multi-band equalizer*; también, en esta sección se presenta el diseño del microsistema y la interfaz gráfica de usuario. En la sección 3 se presentan los resultados de síntesis y verificación en *hardware*. Finalmente, en la sección 4 se presentan las conclusiones.

2. METODOLOGÍA

El diseño del sistema *hardware* para procesamiento de efectos de audio digital se llevó a cabo en cuatro etapas: a) selección y simulación funcional de los efectos en Simulink usando DSP Builder; b) implementación en *hardware* de los efectos usando Quartus II y verificación del funcionamiento de los efectos usando Matlab para graficar las señales de salida; c) diseño del procesador y la interfaz gráfica táctil de usuario para configurar los parámetros de los efectos; d) diseño y verificación del microsistema para procesamiento de efectos de audio.

2.1 Selección y simulación funcional de los efectos de audio

Los efectos de audio seleccionados son basados en retardos, o procesamiento en el dominio dinámico o de la frecuencia. Estos efectos son implementados en el procesador, 14 de los cuales son descritos en Liévano, Espinosa y Velasco (2013). Adicionalmente, se implementó un *multi-band equalizer*, y los efectos *chorus*, *reverb* y *wah-wah* fueron rediseñados. La simulación funcional de estos efectos de audio es llevada cabo en Simulink usando DSP Builder.

2.1.1 Procesamiento basado en retardos

El procesamiento basado en retardos consiste en sumar la señal de audio con ella misma, atenuada y/o desfasada. Cinco efectos de este tipo fueron implementados en el procesador, donde *delay*, *flanger* y *phaser* son descritos en Liévano, Espinosa y Velasco (2013), y *chorus* y *reverb* se describen a continuación.

El *chorus* es descrito por la **Ecuación 1** y es un efecto que emula a dos o más músicos interpretando simultáneamente el mismo tipo de instrumento y la misma pieza musical (Zölzer, 2002). Este efecto se obtiene al sumar la señal de entrada actual con una señal de entrada anterior atenuada por el factor aleatorio *g*.

$$y\,(n) = x(n) + g * x\,(n + del) \qquad \textbf{(1)}$$

Donde, *del* es el retardo entre 10 y 25 ms.

El efecto *reverb* es descrito por la **Ecuación 2** y es generado cuando se suma la señal de audio con sus reflexiones acústicas. Este efecto es emulado sumando la señal de entrada con sus respectiva réplicas, las cuales tienen diferentes retardos y atenuaciones (Zölzer, 2002).

$$y(n) = x(n) + g_1 x(n+del) + g_2 x(n+2del) + g_3 x(n+3del) + g_4 x(n+4del) \qquad \textbf{(2)}$$

2.1.2 Procesamiento en el dominio dinámico

Este tipo de procesamiento es generalmente no lineal y considera la dinámica de la señal. En el procesador se implementaron ocho efectos: *compressor*, *expander*, *noise gate*, *soft* y *hard clipping*, *sigmoidal distortion*, *sigmoidal piecewise distortion*, *polynomial distortion* y *ring modulator,* los cuales son descritos en Liévano, Espinosa y Velasco (2013).

2.1.3 Procesamiento en el dominio de la frecuencia

El procesamiento en el dominio de la frecuencia se basa en la modificación del espectro del sonido usando filtros digitales (Khosravi, 2007). En el procesador se implementaron dos efectos *wah-wah* y *multi-band*

equalizer. El efecto *wah-wah* se obtiene al filtrar la señal de entrada usando un filtro pasa-banda de banda angosta que tiene una frecuencia central variable, lo cual genera un sonido similar a la palabra *wah-wah* (Zölzer, 2002). El efecto *multi-band equalizer* modifica el espectro de audio de una señal mediante la amplificación de determinadas bandas de frecuencia. Este efecto es implementado usando filtros *shelving* y *peak* de primer y segundo orden, los cuales son conectados en serie y controlados independientemente. Los filtros *shelving* amplifican o atenúan las bandas de alta o baja frecuencia usando los parámetros frecuencia de corte f_c y ganancia G. Los filtros *peak* amplifican o atenúan las bandas de mediana frecuencia usando los parámetros frecuencia de corte f_c, ancho de banda f_b y ganancia G (Zölzer, 2002).

2.2. Implementación en *hardware* y verificación de los efectos de audio

2.2.1 Implementación de chorus y reverb

Estos efectos usan un buffer circular, el cual es implementado en una RAM de doble puerto (Altera, 2011). Las direcciones para escribir y leer en la RAM de 16Kx16 bits son generadas por un contador de 14 bits. En este caso, la lectura usa un direccionamiento indexado, donde el índice es el *retardo* descrito por la **Ecuación 3** como se puede observar en la **Figura 1**.

$$retardo = \frac{del}{1000} * fs \qquad \textbf{(3)}$$

El efecto *chorus* es diseñado usando un buffer circular, un sumador, un *right-shifter*, un *NCO* de baja frecuencia y un *LFSR* (Pérez, 2006), como se muestra en la **Figura 2**. En un trabajo previo (Liévano, Espinosa y Velasco, 2013), el retardo del efecto *Chorus* es implementado usando un LFSR, pero en este trabajo el retardo es implementado con un NCO debido a que en Pérez (2006) se recomienda usar un LFO.

El efecto *reverb* es diseñado usando cuatro *buffers* circulares, cuatro *right-shifters*, un filtro IIR paso-bajo y cuatro sumadores (Pérez, 2006), como se muestra en la **Figura 3**. En este caso, se usan cuatro valores configurables para la atenuación y el retardo con el propósito de emular las reflexiones acústicas de la señal de entrada. Los primeros valores para la atenuación y el retardo son parámetros de configuración, y los otros valores son generados multiplicando los primeros valores por 2, 4 y 8. El filtro IIR fue implementado usando la **Ecuación 4**.

Figura 1. Diagrama de bloques del *buffer* circular

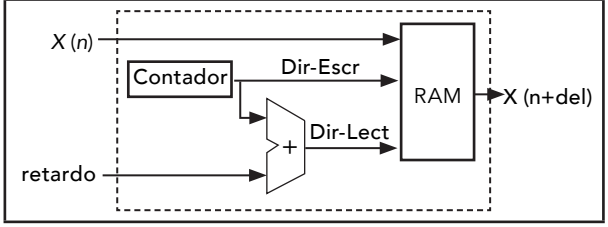


Figura 2. Diagrama de bloques del *chorus*

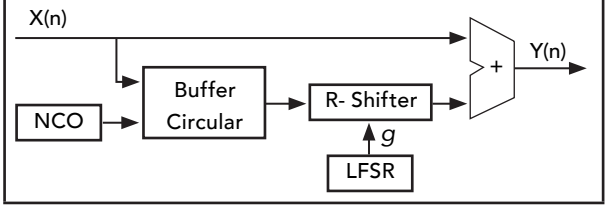
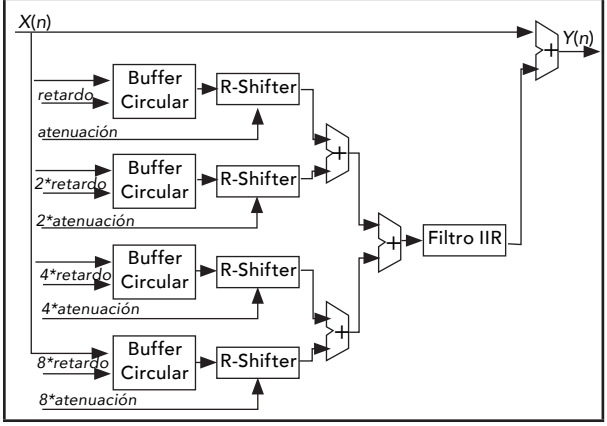


Figura 3. Diagrama de bloques de la *reverb*



$$Y(n) = 0{,}4y(n) - 0{,}2499y(n\text{-}2) + 0{,}0441y(n\text{-}3) + 0{,}5814x(n\text{-}1) + 0{,}2142x(n\text{-}2) \qquad \textbf{(4)}$$

En un trabajo previo (Liévano, Espinosa y Velasco, 2013), el efecto *reverb* es implementado sin usar un filtro IIR, pero en este trabajo se usa el filtro IIR paso bajo con el propósito de obtener una emulación realista de la reverberación (Pérez, 2006).

2.2.2 Implementación del wah-wah

El efecto *wah-wah* es implementado usando un filtro FIR pasa-banda de orden 30, el cual es diseñado

usando una ventana *Hamming*, una arquitectura simétrica, frecuencia central configurable y un ancho de banda de 1000 Hz. En este caso, el filtro FIR es configurado desde una pantalla táctil (ver **Figura 11**), específicamente variando el icono control del *wah-wah*, seleccionando uno de los filtros FIRs presentados en la **Tabla 1**. Por ejemplo, el filtro FIR 6 tiene frecuencias de corte inferior y superior de 750 Hz y 1750 Hz, respectivamente. Entonces, el sonido *wah-wah* será generado si la señal de audio se encuentra en la respectiva ventana del filtro seleccionado.

La implementación en *hardware* del filtro FIR de orden 30 usa 16 coeficientes y fue diseñado usando el FIR Compiler 10.1 de Altera (Altera, 2013). En este diseño se usan 16 RAMs de 16x16 bits para almacenar los 16 coeficientes de cada uno de los 16 filtros presentados en la **Tabla 1**, donde la señal *coeff-set* es la dirección para seleccionar los coeficientes del filtro. En la **Figura 4** se muestra el diagrama de bloques del filtro FIR configurable.

2.2.3 Implementación del multi-band equalizer

Holters y Zölzer (2006) proponen el diseño de un *digital parametric equalizer* de tres bandas usando una tasa de muestreo de 48 kHz. El *equalizer* es implementado como una cascada de filtros *shelving*. En este caso, se diseñó el *multiband-equalizer* para atenuar o amplificar componentes de frecuencias bajas, media-bajas, media-altas y altas, y es implementado usando cuatro filtros FIRs de orden 100 y cuatro registros de desplazamiento configurables. Los filtros usados son uno paso-bajo y tres pasa-bandas, los cuales son diseñados con una ventana *Blackman* usando *FIR Compiler* y sus frecuencias de corte son presentadas en la **Tabla 2**. Los registros de desplazamiento L-R *shifters* amplifican o atenúan la señal de salida de los filtros FIRs, multiplicando o dividiendo la señal por o entre 2^n, donde n es igual a uno o dos. En la **Figura 5** se muestra el diagrama de bloques del *multi-band equalizer*.

El diseño de cada efecto de audio fue verificado mediante el cálculo de la correlación entre los resultados de la simulación funcional en DSP-Builder que usa aritmética de punto flotante de 64 bits, y los resultados de la verificación en *hardware* que usa aritmética de punto fijo de 16 bits. En este caso, el error se calculó usando

Figura 4. Diagrama de bloques del *wah-wah*

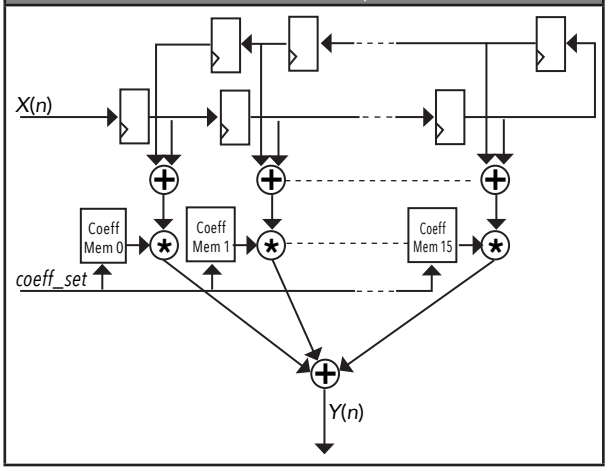


Tabla 1. Parámetros de configuración para el filtro pasa-banda del *wah-wah*

Filtro	Coeff-set	Frecuencia de corte inferior (Hz)	Frecuencia corte superior (Hz)
1	0000	1.000	---------
2	0001	150	1.150
3	0010	300	1.300
4	0011	450	1.450
5	0100	600	1.600
6	0101	750	1.750
7	0110	900	1.900
8	0111	1.050	2.050
9	1000	1.200	2.200
10	1001	1.350	2.350
11	1010	1.500	2.500
12	1011	1.650	2.650
13	1100	1.800	2.800
14	1101	1.950	2.950
15	1110	2.100	3.100
16	1111	2.250	3.250

la **Ecuación 5**, donde *Ym*Yr* es la correlación entre los resultados de simulación y las pruebas de verificación en *hardware*.

$$error = 100 \times (1 - (Ym * Yr)) \quad \textbf{(5)}$$

Los efectos *compresssor*, *expander*, *chorus*, *noise gate* y *phaser* tienen el error mínimo, 0,001 %, y el *flanger* tiene el error máximo, 0,674 % (Liévano, Espinosa y Velasco, 2013).

2.3. Diseño del procesador para efectos de audio digital

El procesador es diseñado usando una unidad de flujo de datos y una unidad de control, tal como se muestra en la **Figura 6**. La unidad de flujo de datos es diseñada usando un arreglo de bloques de efectos de audio digital (*Arreglo FX*) y un banco de RAMs para almacenar la configuración de los efectos de cada bloque. La unidad de control es diseñada usando una máquina de estados (*FSM*) descrita en VHDL comportamental. Adicionalmente, el procesador tiene un registro de entrada serial y salida paralela de 16 bits (*Reg-In*) para almacenar la señal de entrada X_i y un registro de entrada paralela de 16 bits y salida serial (*Reg-Out*) para almacenar la señal de salida Y_i.

El Arreglo *FX* es implementado usando 10 bloques de efectos de audio digital, *FX1-FX10,* conectados en cascada, tal como se muestra en la **Figura 7**. El orden o la secuencia de la cascada de los bloques de efectos corresponde al usado por las pedaleras comerciales (Sound Laboratory Zoom, 2013). Entonces, los dos primeros bloques de efectos adecuan la señal para ser procesada por los otros bloques, es decir, el efecto *noise gate* elimina el ruido de la señal y el bloque *compressor/expander* comprime o expande la señal según su nivel de voltaje. El orden de los efectos *ring modulator*, *phaser/flanger*, *chorus* y *delay* es irrelevante. Los efectos *multi-band equalizer* y *reverb* mejoran la señal de audio procesada por los efectos anteriores. El efecto *wah-wah* está al final de la cascada de efectos porque su

Tabla 2. Filtros FIRs para el *multi-band equalizer*

Filtro	Frecuencia de corte inferior (Hz)	Frecuencia de corte superior (Hz)
1	125	----------
2	125	500
3	500	2.000
4	2.000	20.000

Figura 5. Diagrama de bloques del *multiband-equalizer*

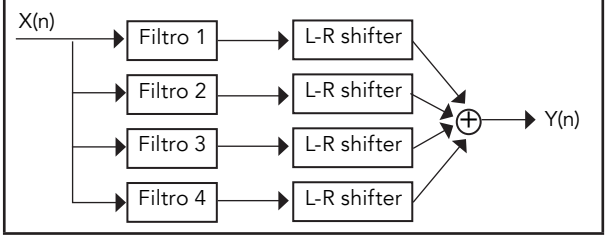
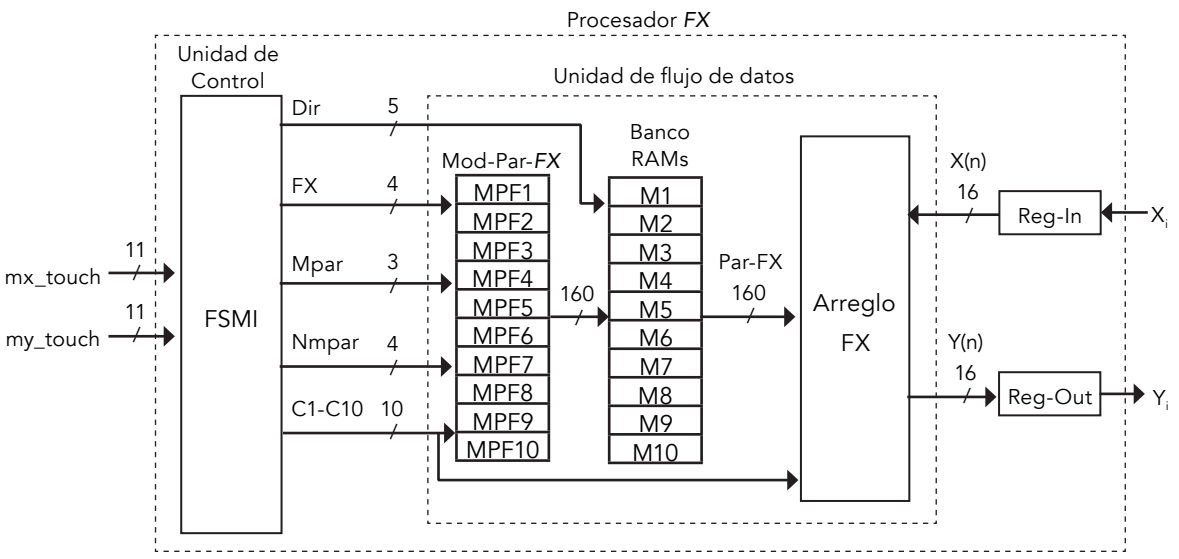


Figura 6. Arquitectura del procesador de efectos de audio digital



implementación requiere de muchos recursos de área; esto implica que la señal se degrada cuando este bloque es ubicado en medio de los otros efectos. La mayoría de los bloques tienen un efecto, a excepción de los bloques *FX2*, *FX3* y *FX6*, que tienen 2 ó 3 efectos. Los bloques son configurados usando 10 registros, *RC1-RC10*, los cuales almacenan la palabra de configuración de 16 bits para cada efecto, y la salida de cada bloque se conecta a un *multiplexor,* el cual permite seleccionar la señal de entrada del próximo bloque, es decir, los multiplexores permiten elegir el conjunto de bloques de efectos que procesan la señal de entrada X_i.

El banco de memorias es conformado por 10 RAMs y cada RAM tiene 32 palabras de 16 bits, donde cada palabra almacena una configuración de un efecto de audio. Por ejemplo, si un bloque tiene un solo efecto, este efecto puede tener 32 opciones de configuración. Las primeras 16 posiciones de cada RAM almacenan las configuraciones predeterminadas para cada bloque de efectos usando un archivo *.mif*, y las últimas 16 posiciones de cada RAM son usadas para almacenar las nuevas configuraciones desarrolladas por el usuario.

La configuración del efecto en cada una de las RAM M_i se lleva a cabo usando la señal de escritura *WE*, la señal de dirección *Dir* y la palabra de configuración de 16 bits obtenida desde el bloque Mod-Par-FX, el cual está basado en comparadores y multiplexores, tal como se muestra en la **Figura 8**.

Desde la **Figura 8** es posible observar que la modificación de uno de los parámetros de la palabra de configuración del efecto es llevada a cabo si los valores de la señal *FX*&*Mpar* (efecto FX y parámetro *Mpar*) y la constante $Const_i$ son iguales; en caso contrario la palabra de configuración no es modificada. También se observa que el código del efecto *FX* tiene 4 bits, el código del parámetro *Mpar* tiene 3 bits, el valor del parámetro *Nmpar* tiene 2, 3 ó 4 bits y cada constante $Const_i$ tiene 7 bits, donde *i* es un entero entre 1 y 4.

La unidad de control del procesador es implementada usando la máquina de estados *FSM1*, la cual controla el sensado de la pantalla táctil LCD de la interfaz de usuario y genera las señales de control para el Arreglo *FX*, el bloque *Mod-Par-FX* y el banco de RAMs. La *FSM1* supervisa todo el tiempo los objetos gráficos o íconos de la interfaz gráfica para generar las señales de control que permiten modificar una palabra de configuración de un efecto. Este procedimiento se lleva a cabo en cuatro pasos: 1) cargar en cada registro

Figura 7. Arreglo FX del procesador de efectos de audio

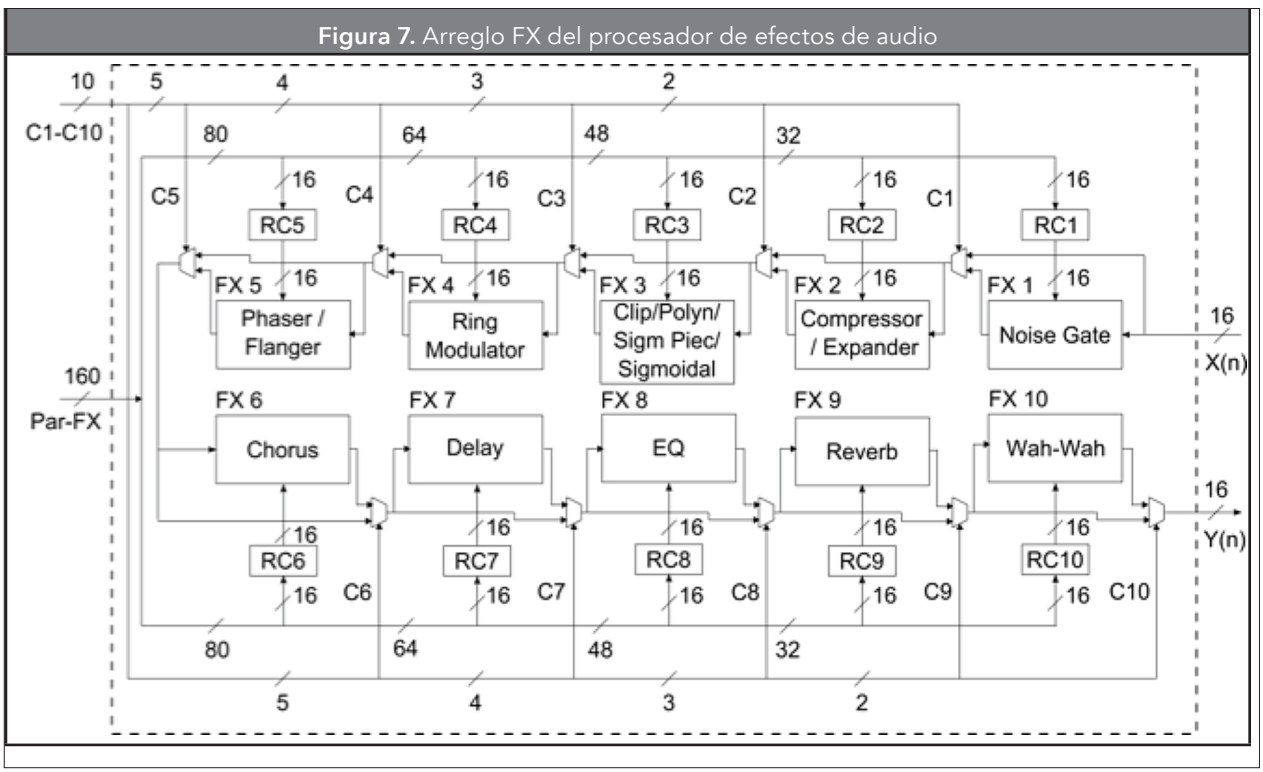
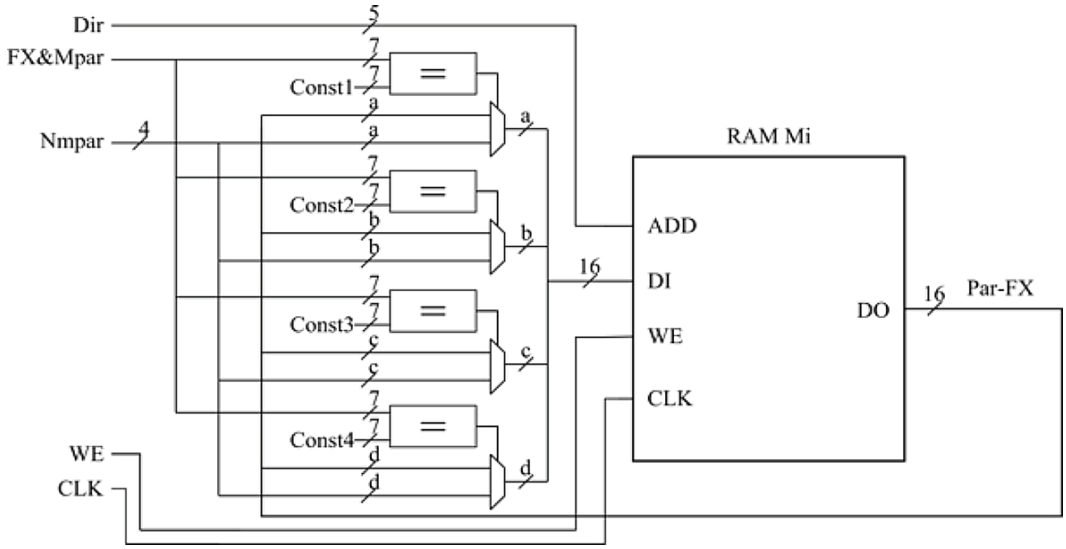


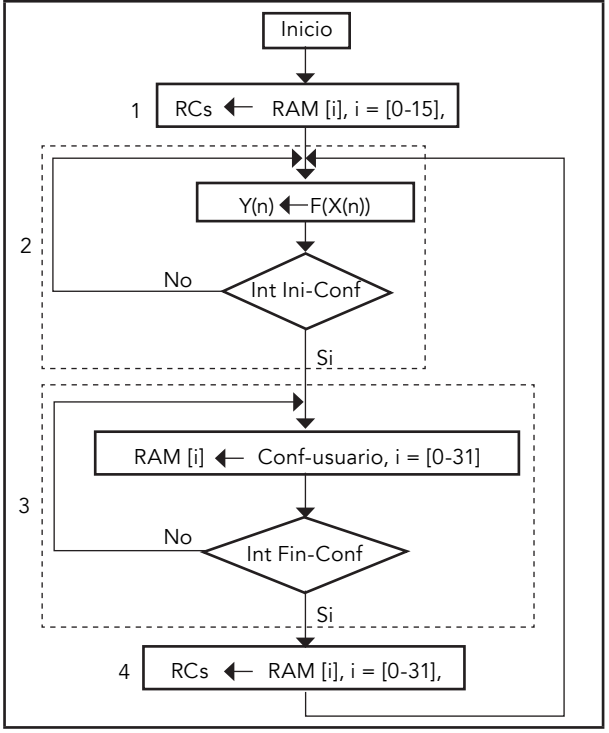
Figura 8. Diagrama de bloques para configurar los parámetros de un efecto



de configuración *RC* una de las 16 configuraciones predeterminadas para cada bloque de efectos. 2) Seleccionar el conjunto de bloques de efectos de audio del Arreglo *FX* que van a procesar la señal de audio digitalizada *X(n)* usando las señales *C1-C10.* 3) Cargar la nueva configuración determinada por el usuario en una posición de la RAM correspondiente al bloque de efectos. Esta configuración se lleva a cabo modificando una predeterminada que se encuentra en una de las 16 primeras posiciones de la RAM o generando una nueva que se almacena en una de las 16 últimas posiciones de la RAM. 4) Cargar en los registros *RCs* la nueva configuración desde la RAM, es decir una configuración predeterminada, predeterminada-modificada o nueva. En la **Figura 8** se muestra el diagrama *ASM* (*Algorithmic State Machine*) de la *FSM1*.

En la **Figura 9** se muestra que las señales de control *Ini-Conf* y *Fin-Conf* son usadas para inicializar y finalizar una nueva configuración, respectivamente. Estas dos señales son generadas por el usuario desde la pantalla táctil usando el controlador *LTM-SoPC* (Altera, 2011a).

Figura 9. Diagrama ASM de la FSM1



2.4. Diseño del microsistema *hardware* para procesar efectos de audio

Con el propósito de usar el procesador en una aplicación real se diseñó un microsistema de procesamiento para guitarra eléctrica, el cual es implementado usando una unidad *hardware* y una interfaz gráfica de usuario basada en una pantalla táctil LCD para configurar el procesador diseñado. La unidad *hardware* es

Figura 10. Diagrama de bloques del microsistema para efectos de audio

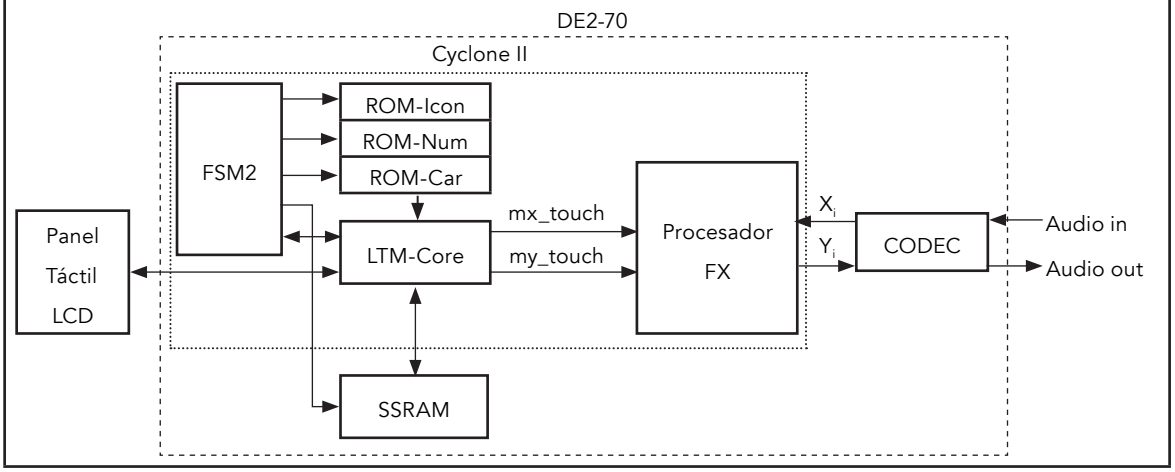
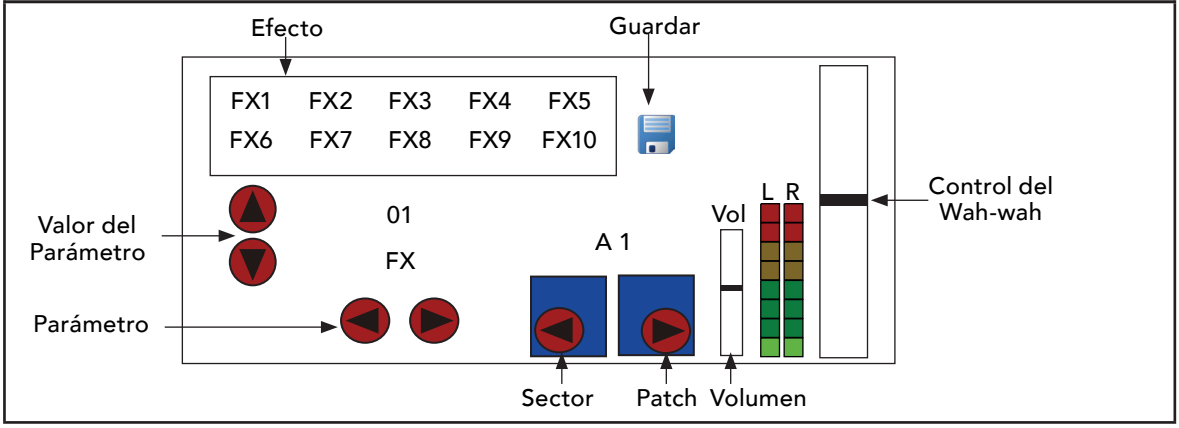


Figura 11. Interfaz gráfica de usuario para programar los efectos de audio digital



implementada en el sistema de desarrollo DE2-70 de Terasic usando el FPGA, el códec y la memoria DRAM. En el FPGA se sintetizaron el procesador de efectos de audio, tres memorias ROMs y el controlador *LTM-SoPC* para la pantalla táctil LCD (*LTP: LCD Touch Panel*), tal como se muestra en la **Figura 10**.

La interfaz gráfica de usuario es diseñada usando la máquina de estados *FSM2*, la cual genera los pixeles de la pantalla táctil LCD correspondientes a las imágenes para el fondo, y los objetos gráficos fijos y variables; estos últimos son generados por la unidad *hardware*, por ejemplo, la imagen del volumen del sonido.

La FSM2 controla la generación de las imágenes usando la información almacenada en las tres memorias ROMs, y lleva a cabo la transferencia de las imágenes a la memoria RAM y a la pantalla táctil *LCD* usando el controlador *LTM-SoPC*.

En la **Figura 11** se muestra la interfaz gráfica de usuario, la cual tiene los siguientes objetos gráficos y sus respectivos visualizadores: efectos, parámetro, valor del parámetro, sector, *patch*, volumen, control del *wah-wah* y guardar. La modificación del parámetro de un efecto se lleva a cabo seleccionando un sector y un *patch*, los cuales se usan para direccionar la posición de la memoria RAM donde se va a almacenar la nueva palabra de configuración del efecto.

La interfaz gráfica tiene los sectores *A*, *B*, *C* y *D* para seleccionar las 32 palabras de configuración, donde cada sector tiene ocho posiciones de memoria y cada posición del sector se denomina *patch*. Los sectores

Figura 12. Interfaz de usuario visualizada en la pantalla táctil LCD

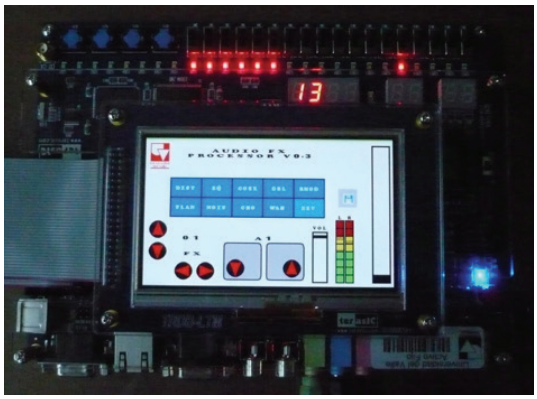
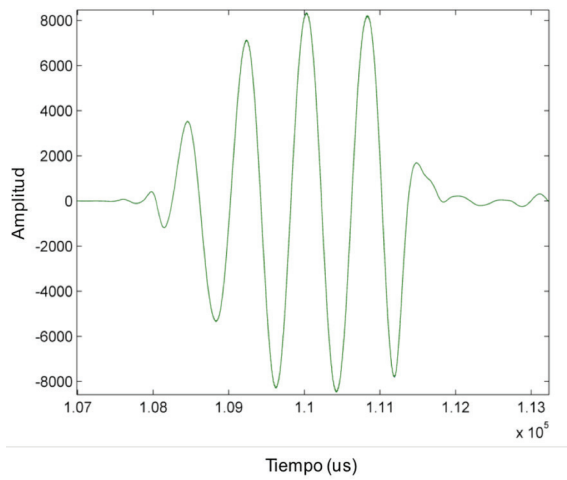


Figura 13. Señal de audio en la entrada análoga del códec



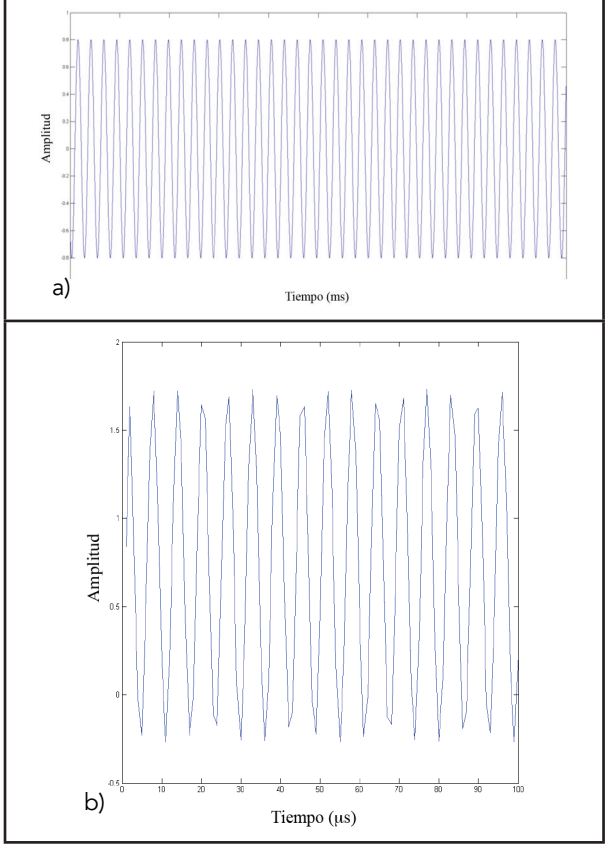
A-B y *C-D* permiten direccionar las 16 primeras y las 16 últimas posiciones de la RAM M_i, respectivamente.

La modificación de un parámetro en la palabra de configuración de un efecto se realiza de la siguiente manera: 1) Seleccionar uno de los diez bloques de efectos; 2) Seleccionar el parámetro, y 3) Modificar el valor del parámetro. En la **Figura 12** se muestra la interfaz gráfica de usuario desplegada en la pantalla táctil *LCD*.

2.5 Verificación del microsistema para procesar efectos de audio

El microsistema es verificado usando un reproductor MP3 y un PC. Inicialmente, se verificaron

Figura 14. Señal de salida del efecto reverb: **a)** simulación en Matlab. **b)** procesador



cada uno de los bloques de efectos del procesador, y posteriormente se realizaron verificaciones de funcionamiento para dos conjuntos de efectos conectados en cascada. Para verificar cada efecto en el procesador se usó una señal de audio almacenada en un reproductor MP3 cuya salida de audio es conectada a la entrada análoga del *códec* del sistema de desarrollo DE2-70. En la **Figura 13** se muestra la señal de audio de prueba en la entrada del *códec*, el cual fue configurado para muestrear la señal de prueba a una frecuencia de 44,1 kHz. Entonces, en la salida del *códec* se obtiene la señal de audio digitalizada X_i de 16 bits, la cual tiene un formato de representación en complemento a dos y es usada como la señal de entrada para el procesador.

La señal de salida Y_i del procesador de efectos de audio es conectada a la entrada digital del códec, cuya salida análoga es conectada a la entrada de audio de un PC. La señal análoga generada por el *códec* es digitalizada por

Tabla 3. Recursos de área de los efectos de audio

Efecto	ALUTs	% ALUTs del microsistema	Registros	% Registros del microsistema
Compressor	84	0,28	50	0,49
Expander	84	0,28	50	0,49
Noise gate	65	0,22	48	0,47
Soft and hard clipping	80	0,27	49	0,48
Sigmoidal distortion	1225	4,08	891	8,70
Sigmoidad piecewise distortion	231	7,69	131	1,28
Polynomial distortion	64	0,21	63	0,61
Ring modulator	539	1,79	405	3,95
Delay	49	0,16	49	0.48
Chorus	44	0.15	44	0,43
Flanger	534	1,78	390	3,81
Reverb	214	0,71	49	0,48
Phaser	516	1,72	386	3,77
Wah-wah	24009	79,92	3743	36,55

Tabla 4. Recursos de área de los bloques del microsistema programable para efectos de audio digital

Bloque	ALUTs	% ALUTs del microsistema	Registros	% Registros del microsistema
Arreglo FX	27738	92.34	6348	62.00
Mod_par	278	0.92	0	0
Unidad de Control	2025	6.74	3891	38.00

el códec del PC y esta señal es graficada usando Matlab. La **Figura 14** muestra la señal de salida del efecto *reverb* simulado en Matlab usando como entrada una señal sinusoidal, y la señal de salida Y_i del procesador para el mismo efecto usando como entrada la señal de audio.

Desde la **Figura 14**, se puede observar que la señal de salida del efecto *reverb* simulado en Matlab es similar a la señal de salida del procesador para el mismo efecto.

3. RESULTADOS

El procesador de efectos de audio, las tres memorias ROMs, la máquina de estados *FSM2* y el controlador *LTM-SoPC* son sintetizados sobre el FPGA EP2C70F896C6, usando 30.040 ALUTs (*Adaptative Look-Up Tables*) y 10.239 registros, los cuales corresponden al 44 % y 15 % de los recursos de área del FPGA, respectivamente. La frecuencia máxima de operación del procesador es de 195,62 MHz. Adicionalmente, en la **Tabla 3** se presentan los recursos de área usados en el FPGA para cada bloque de efectos de audio y el porcentaje de recursos de área de los bloques de efectos de audio con respecto al total de recursos del microsistema. Desde los resultados de la **Tabla 3**, se puede concluir que el efecto que usa menos recursos de área es el *chorus*, mientras que el *wah-wah* usa la mayor cantidad de recursos área, el cual corresponde al 80 % del total de recursos del microsistema.

Por otro lado, la **Tabla 4** presenta los recursos de área de los bloques del microsistema en el FPGA y el porcentaje de recursos de área de cada bloque del microsistema con respecto al total de recursos de área del mismo. Desde los resultados anteriores se puede concluir que el bloque Arreglo *FX* usa la mayor cantidad de recursos de área, el cual es el 92 % para las ALUTs y el 62 % para los registros.

Con el propósito de verificar en forma experimental (procedimiento auditivo) el funcionamiento del microsistema para procesar efectos de audio, se

Figura 15. Señal sinusoidal

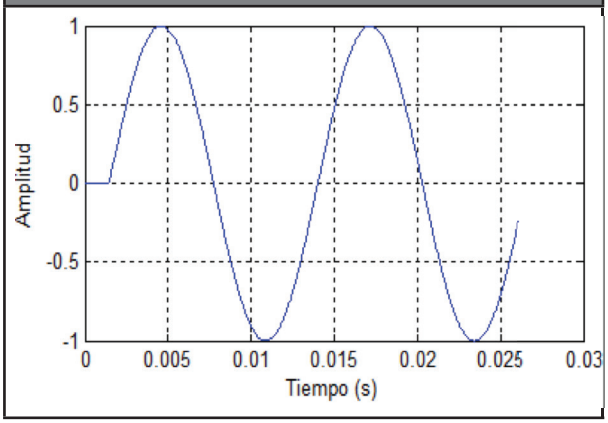


Figura 16. Canción reproducida desde un MP3

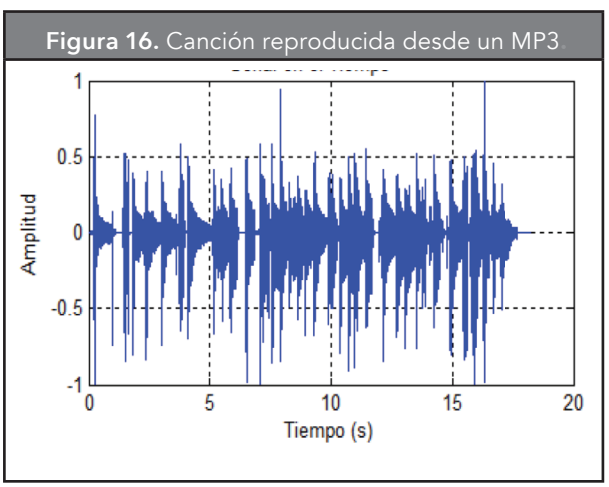
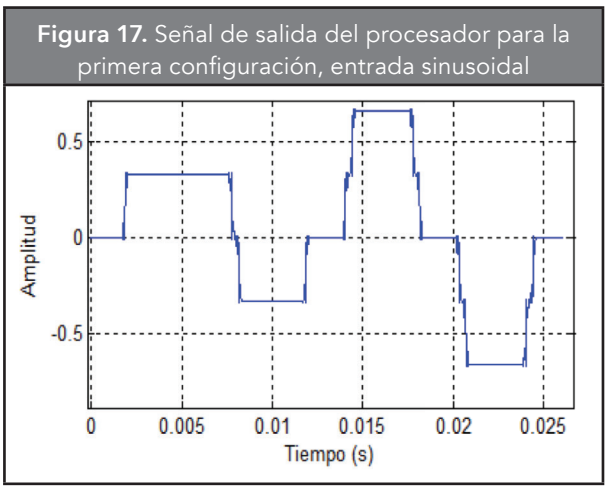


Figura 17. Señal de salida del procesador para la primera configuración, entrada sinusoidal



implementaron dos configuraciones del procesador y se usaron dos señales, una señal sinusoidal y una canción reproducida desde un MP3, las cuales son mostradas en las **Figuras 15** y **16**, respectivamente.

La primera configuración del procesador consiste en conectar en cascada los efectos *noise gate*, *hard clipping* y *delay*, cuyos parámetros de operación son: umbral 0,1, umbral 0,4 y retardo 10 ms, respectivamente. Las señales de salida del procesador para esta configuración, usando como entradas la señal sinusoidal y la canción, son graficadas usando Matlab, las cuales se muestran en las **Figuras 17** y **18**, respectivamente.

La segunda configuración del procesador conecta en cascada los efectos *compressor*, *soft clipping*, *ring modulator* y *reverb*, cuyos parámetros de operación son: umbral 0,4 y atenuación 0,6; umbral 0,6; sinusoidal con frecuencia de 150 Hz; y atenuación 4 y retardo 15 ms, respectivamente. Las señales de salida del procesador para la segunda configuración, usando las señales anteriores, son graficadas usando Matlab, las cuales se muestran en las **Figuras 19** y **20**, respectivamente.

Desde las **Figuras 17** hasta la **20**, se puede observar que las señales de prueba son modificadas por los efectos de audio programados en el procesador, es decir la señal de entrada digitalizada X_i es procesada por la respectiva secuencia de efectos. Sin embargo, para verificar el correcto procesamiento de los efectos configurados en el procesador se debe realizar un procesamiento digital a la señal de salida usando las transformadas FFT ó *Wavelet*, o en su defecto se debe recurrir a una persona con excelente oído musical.

Teniendo en cuenta la literatura revisada, una implementación similar de un procesador de efectos de audio es presentada en (Pfaff, *et al.*, 2007), en ese trabajo se diseña un procesador sintetizado en el FPGA EP2C35F usando co-diseño *hardware/software*, y son implementados cinco efectos, los cuales pueden ser configurados por el usuario. Sin embargo, nuestro procesador tiene más efectos y son organizados en un orden que permite llevar a cabo un correcto procesamiento de la señal de audio. Adicionalmente, los autores no presentan resultados de síntesis y verificación, por lo tanto no es posible efectuar una comparación real con nuestro procesador.

Figura 18. Señal de salida del procesador para la primera configuración, entrada la canción

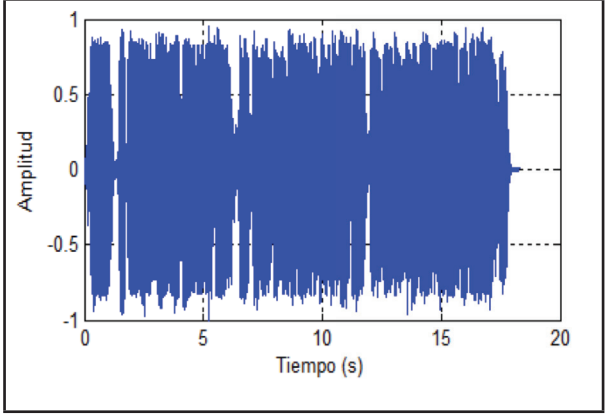


Figura 19. Señal de salida del procesador para la segunda configuración, entrada Sinusoidal

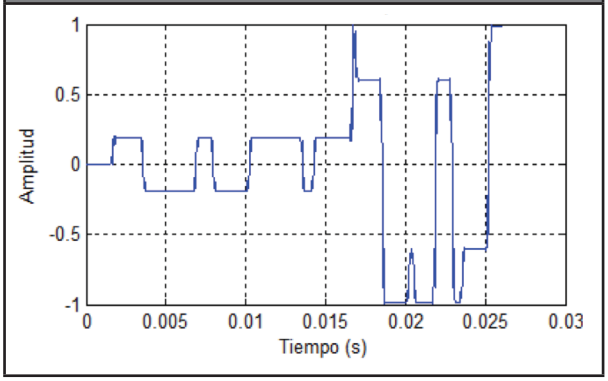
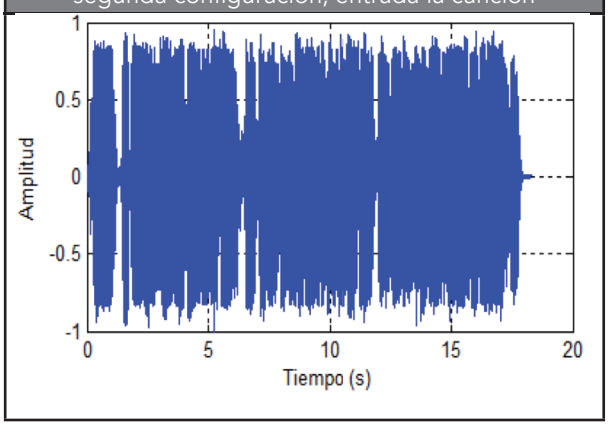


Figura 20. Señal de salida del procesador para la segunda configuración, entrada la canción



4. CONCLUSIONES

Este trabajo presenta la implementación de un microsistema para procesar efectos de audio digital, el cual está basado en un procesador de propósito específico y reconfigurable. La principal ventaja de este microsistema, con respecto a las implementaciones basadas en DSPs, PCs o GPUs, es que permite obtener una tasa de muestreo muy alta para procesar un audio de gran calidad en tiempo real, es decir el sistema *hardware* basado en el procesador de efectos de audio digital podría utilizarse a nivel profesional.

Adicionalmente, el usuario puede generar efectos de audio no muy comunes debido a que puede configurar una cadena de efectos del mismo tipo. Esto se debe a la facilidad y flexibilidad para modificar los parámetros de los efectos, y a la baja latencia de procesamiento que tiene el procesador.

La configuración del procesador se lleva a cabo usando una interfaz gráfica de usuario basada en una pantalla táctil *LCD.* En este caso, el usuario puede seleccionar una de las 16 configuraciones de efectos de audio predeterminadas o almacenar 16 nuevas configuraciones. Esta interfaz proyecta al sistema como un prototipo que puede generar un producto comercial basado en un FPGA.

El microsistema es descrito en VHDL, sintetizado en el FPGA EP2C70F896C6, e implementado en el sistema de desarrollo DE2-70 con una pantalla táctil *LCD*. Las pruebas de síntesis y verificación en *hardware* permiten concluir que el procesador de efectos de audio tiene una tasa de muestreo máxima de 195,62 MSPS, y puede ser usado como un *core* embebido en un SoC para aplicaciones de audio, como por ejemplo una pedalera.

AGRADECIMIENTOS

Los autores agradecen al profesor Jaime Andrés Arteaga por su colaboración y asesoría en el diseño de la interfaz gráfica usando el *LCD Touch Panel*; al estudiante Joaquín Andrés Alarcón por su colaboración en la verificación de los efectos *chorus* y *reverb*.

REFERENCIAS

Altera (2011). Documentation: SOPC Builder [en línea] Disponible en: http://www.altera.com/literature/lit-sop.jsp [consultado el 4 de octubre de 2013].

Altera (2011). FIR Compiler user guide [en línea]. Disponible en: http://www.altera.com/literature/ug/fircompiler_ug.pdf [consultado el 4 de octubre de 2013].

Altera (2009). Quartus II Development Software Handbook v9.0 [en línea] Disponible en:http://www.altera.com/literature/hb/qts/archives/quartusii_handbook_9.0.pdf [consultado el 4 de octubre de 2013].

Berdahl, E.; Smith, J. O. (2006). Some Physical Audio Effects. *International Conference on Digital Audio Effects.* Montreal, Canada. (Septiembre 18-20), pp. 165-168.

Byun, K.; Kown, Y.-S.; Koo, B.-T.; Eum, N.-W.; Jeong, K.-H.; Koo, J.-E. (2009). Implementation of Digital Audio effect SoC. I*EEE Interntational Conference Multimedia and Expo*. New York, U.S.A. (Junio 28 – Julio 2), pp. 1194-1197.

Fernández, P.; Casajús, J. (2000). Multiband Approach to Digital Audio FX. *IEEE International Conference Multimedia and Expo*. Nueva York, U.S.A. (Julio 30 – Agosto 2), pp. 1747-1750.

Guillemard, M.; Ruwwe, C.; Zölzer, U. (2005). J-DAFX - Digital Audio Effects in JAVA. *International Conference on Digital Audio Effects*. Madrid, España. (Septiembre 20-22), pp. 1-6.

Holters, M.; Zölzer, U. (2006). Parametric Higher-Order Shelving Filters. *European Signal Processing Conference.* Florence, Italy. (Septiembre 4-8).

Karjalainen, M.; Penttinen, H.; Välimäki, V. (2000). Acoustic Sound from the Electric Guitar Using DSP Techniques. *IEEE International Conference on Acoustics, Speech and Signal Processing*. Istanbul, Turkey. (Junio 5-9), pp. 773-776.

Khosravi, P. (2007). On the Design of Spectral Tools in Blue. Csound Journal [en línea] (7). Disponible en: http://www.csounds.com/journal/issue7/onTheDesignOfSpectralToolsInBlue.html [consultado el 4 de octubre de 2013].

Liévano, P. P.; Espinosa-Duran, J. M. y Velasco-Medina, J (2013). Implementación de algoritmos para efectos de audio digital con alta fidelidad usando *hardware* programable. *Revista Ingeniería y Universidad*, 17(1) (Enero), pp. 93-108.

Ling, F.; Khuen, F.; Radhakrishnan, D. (2000). An Audio Processor Card for Special Sound Effects. *IEEE Midwest Symposium on Circuits and Systems*. Michigan, U.S.A. (Agosto 8-11), pp. 730-733.

Oboril, D.; Barik, M.; Schimmel, J.; Smekal, Z.; Krkavec, P. (2000). Modelling Digital Musical Effects for Signal Processors, Based on Real Effect Manifestation Analysis. *Cost G-6 Conference on Digital Audio Effects.* Verona, Italia. (Diciembre 7-9), pp. 1-6.

Pérez, A. *Estudio de efectos de audio para guitarra, e implantación mediante DSP* Tesis de pregrado, Escuela Técnica Superior de Ing., U. P. Comillas, Madrid, España, 2006.

Pfaff, M.; Malzner, D.; Seifert, J.; Traxler, J.; Weber, H.; Gerhard, W. (2007). Implementing Digital Audio Effects Using Hardware/Software Co-Design Approach. *International Conference on Digital Audio Effects.* Bordeux, Francia. (Septiembre 10-15), pp. 1-8.

Schimmel, J.; Smekal, Z.; Krkavec, P. (2002). Optimizing Digital Musical Effect Implementation for Multiple Processor DSP Systems. *International Conference on Digital Audio Effects*. Hamburg, Germany. (Septiembre 26-28), pp. 81-84.

Sound laboratory Zoom (3013). Zoom70II guitar operation manual. [en línea]. Disponible en: http://www.zoom.co.jp/downloads/707ii/software/ [consultado el 4 de octubre de 2013].

Tsai, P.-Y.; Wang, T.-M.; Su, A. (2010). GPU-Based Spectral Model Synthesis for Real-Time Sound Rendering. *International Conference on Digital Audio Effects*. Graz, Austria. (Septiembre 6-10), pp. 1-5.

Verfaille, V.; Zölzer, U.; Arfib, D. (2006). Adaptive Digital Audio Effects (A-DAFx): A New Class of Sound Trans-Formations. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5) (septiembre), pp. 1817-1831.

Zölzer, U. *DAFX - Digital Audio Effects*. Chichester, England: J. Wiley & Sons, 2002.

PARA CITAR ESTE ARTÍCULO / TO REFERENCE THIS ARTICLE / PARA CITAR ESTE ARTIGO /

Espinosa-Durán,J. M.; Liévano-Torres, P. P.; Rentería-Mejía, C. P.; Velasco-Medina, J. (2014). Diseño de un Microsistema Programable para Efectos de Audio Digital Usando FPGAs. Revista EIA, 11(22) julio-diciembre, pp. 131-144. [Online]. Disponible en: http://dx.doi.org/10.14508/reia.2014.11.22.133-146