

DISEÑO DE FUNCIONES DSP USANDO VHDL Y CPLDs-FPGAs

Mario Vera-Lizcano, Gustavo Vejarano, Jaime Velasco-Medina

Grupo de Bioelectrónica y Nanoelectrónica, EIEE, Universidad del Valle
A.A. 25360, Cali, Colombia

E-mail: mario@univalle.edu.co, jvelasco@univalle.edu.co

ABSTRACT

In this paper we present a simple methodology to design basic DSP building blocks, and the state-of-the-art in architectures for DSP microprocessors and CPLDs-FPGAs oriented to DSP design. The design methodology is initialized with the design at the software level using MATLAB, later the architecture is designed and captured using VHDL, finally the simulations are carried out using MAX+plus II and the design is implemented in a CPLD circuit. As test vehicle some simple FIR filters are designed, and the simulation and experimental results allow to validate the design methodology proposed.

RESUMEN

En este artículo presentamos una metodología simple para diseñar bloques funcionales DSP básicos, y el estado del arte en arquitecturas para procesadores DSP y CPLDs-FPGAs orientados al diseño DSP. La metodología de diseño se inicia con el diseño a nivel de software usando MATLAB, luego se concibe la arquitectura y se captura el diseño del hardware usando VHDL, finalmente se llevan a cabo las respectivas simulaciones usando MAX+plus II y se implementa el diseño en un circuito CPLD. Como vehículo de test se diseñaron filtros FIR simples, y los resultados de simulación y experimentales validan la metodología de diseño propuesta.

DISEÑO DE FUNCIONES DSP USANDO VHDL Y CPLDs-FPGAs

Mario Vera-Lizcano, Gustavo Vejarano, Jaime Velasco-Medina

Grupo de Bioelectrónica y Nanoelectrónica, EIEE, Universidad del Valle
A.A. 25360, Cali, Colombia

E-mail: mario@univalle.edu.co, jvelasco@univalle.edu.co

RESUMEN

En este artículo presentamos una metodología simple para diseñar bloques funcionales DSP básicos, y el estado del arte en arquitecturas para procesadores DSP y CPLDs-FPGAs orientados al diseño DSP. La metodología de diseño se inicia con el diseño a nivel de software usando MATLAB, luego se concibe la arquitectura y se captura el diseño del hardware usando VHDL, finalmente se llevan a cabo las respectivas simulaciones usando MAX+plus II y se implementa el diseño en un CPLD-FPGA. Como vehículo de test se diseñaron filtros FIR simples, y los resultados de simulación y experimentales validan la metodología de diseño propuesta.

1. INTRODUCCION

Los procesadores DSP son usados para implementar muchas de las aplicaciones de DSP, tales como: protocolos de voz para Internet (VoIP), comunicaciones inalámbricas (3G wireless), sistemas de radar y satélite, procesamiento de imágenes, sistemas multimedia, etc. Sin embargo, aunque los procesadores son programables a través de software, la arquitectura del hardware del procesador DSP no es flexible. Por lo tanto, los procesadores DSP son limitados por la arquitectura fija del hardware tal como: el desempeño del bus, un número fijo de bloques multiplicador-acumulador (MAC), memoria fija, número fijo de bloques aceleradores de hardware y ancho de datos fijo. Un procesador DSP con arquitectura fija del hardware no es adecuado para ciertas aplicaciones que podrían requerir implementaciones de funciones DSP dedicadas.

Los circuitos CPLDs-FPGAs suministran una solución reconfigurable y eficiente para implementar aplicaciones DSP. Estos circuitos pueden alcanzar un más alto throughput y una mayor potencia de procesamiento de

datos (*raw data processing power*) que los procesadores DSP [1].

Debido a que los circuitos CPLDs-FPGAs pueden ser reconfigurados en hardware, estos ofrecen un hardware completamente dedicado o específico (*hardware customization*) para implementar varias aplicaciones DSP. Por lo tanto, sistemas DSP implementados en FPGAs pueden tener una arquitectura específica, estructura de bus específica, memoria específica, bloques aceleradores de hardware específicos y un número variable de bloques MAC [1][2]. Entonces, los FPGAs ofrecen una oportunidad para acelerar una aplicación DSP hasta 1000 veces más que un microprocesador DSP tradicional [3].

Por ejemplo, en muchos sistemas DSP el filtro FIR es usado para llevar a cabo diferentes tareas tales como el pre-acondicionamiento de la señal, antialiasing, selección de la banda, interpolación y convolución. Sin embargo, un número limitado de circuitos filtros FIR (*off-the-shell FIR filters*) está disponible en el mercado y estos circuitos frecuentemente limitan el desempeño del sistema. Por lo tanto, los circuitos CPLDs-FPGAs son una alternativa ideal para implementar filtros FIR. En este caso, los dispositivos FLEX de Altera, incluyendo FLEX 10K y FLEX 8000 permiten fácilmente implementar filtros FIR. Un circuito FLEX permite implementar una o más funciones de filtrado crítico para una aplicación basada en un microprocesador DSP, liberando al microprocesador para que este lleve a cabo las operaciones algorítmicamente complejas. Un microprocesador DSP puede implementar un filtro FIR de 8 etapas (TAP) a 5 millones de muestras por segundo (MSPS: *million samples per second*), mientras un circuito off-the-shell FIR filter puede trabajar a 30 MSPS. En contraste, un dispositivo FLEX puede implementar el mismo filtro por encima de 100 MSPS [4].

En este artículo presentamos una metodología simple para diseñar bloques funcionales DSP, los cuales pueden ser

incorporados en una librería VHDL para aplicaciones DSP. En este contexto, el artículo describe y muestra la importancia de considerar una librería VHDL para funciones DSP. Como vehículo de test se diseñaron filtros FIR, es decir, como transferir (MAP) a nivel de hardware las operaciones matemáticas del filtro FIR usando un circuito CPLD. Entonces, el artículo presenta el diseño de filtros FIR convencionales usando una arquitectura específica, la cual es descripta estructuralmente usando VHDL. Los detalles de la implementación, incluyendo los compromisos entre desempeño y área del circuito son también presentados.

El artículo está organizado de la siguiente manera, las secciones 2 y 3 presentan el estado del arte en arquitecturas para procesadores DSP y CPLDs-FPGAs orientados al diseño DSP. La sección 4 describe las sub-librerías de una librería VHDL para aplicaciones DSP. La sección 5 presenta una metodología simple de diseño para bloques funcionales DSP, es decir, el diseño a nivel de software usando MATLAB, el diseño de la arquitectura, la simulación y verificación del diseño, y la circuitería de test para probar el funcionamiento del circuito. Finalmente, la sección 6 presenta algunas conclusiones y el trabajo futuro.

2. TENDENCIAS EN LA ARQUITECTURA DE PROCESADORES DSP

Existen varias diferencias entre un procesador DSP y un procesador genérico, sin embargo la diferencia fundamental entre estos procesadores es: a) El bloque de hardware MAC (multiplicador-acumulador) del procesador DSP y b) La memoria y estructura del bus específicas para facilitar el acceso frecuente de datos comúnmente encontrados en aplicaciones DSP.

La operación MAC es usualmente el cuello de botella (desempeño) del procesador DSP en muchas aplicaciones. Algunos procesadores DSP incorporan múltiples bloques MAC en su arquitectura para minimizar el problema del desempeño relacionado con la multiplicación (*to boost the overall multiplier bandwidth*). Por ejemplo, el TMS320C6411 de Texas Instruments puede calcular hasta ocho multiplicaciones de 8X8 en un solo ciclo de reloj.

Sin embargo, mientras la adición de más bloques MACs podría suministrar un más alto *throughput*, en algunos casos la potencia de procesamiento de datos del procesador podría ser reducida (*falls behind in raw data processing power*) para ciertas funciones DSP intensiva en datos tales como decodificador-codificador Viterbi y filtros FIR

Para solucionar este problema, los fabricantes de procesadores DSP, también han tratado de incorporar un bloque acelerador de hardware (*coprocessor*) tal como el coprocesador Viterbi, coprocesador turbo, y el coprocesador filtro (*enhanced-filter-coprocessor*). Mientras los bloques coprocesadores suministran un alto desempeño, la mayoría de las aplicaciones DSP no se pueden beneficiar de los procesadores DSP con bloques aceleradores de hardware prediseñados. Adicionalmente, los bloques aceleradores de hardware son fijos, no permiten un nivel de implementación dedicada (*customization*) para diseños con necesidades específicas, y pueden llegar a ser rápidamente obsoletos [1].

3. TENDENCIAS EN ARQUITECTURAS PARA CPLDs-FPGAs ORIENTADOS A DSP

Los circuitos FPGAs consisten de elementos lógicos y memoria que pueden ser configurados para operar en modos diferentes para una funcionalidad diferente. La flexibilidad del hardware permite al diseñador implementar el diseño usando un adecuado lenguaje de descripción de hardware (HDL) tal como VHDL o Verilog. Entonces, el mismo FPGA puede implementar un enrutador DSL, un modem DSL, un codificador JPEG o un sistema de transmisión digital.

Con la introducción de FPGAs de alta densidad, tal como la familia Stratix FPGA de Altera, la cual incorpora varios bloques funcionales embebidos de alto nivel (*embedded silicon*), los diseñadores pueden implementar sistemas completos dentro de un FPGA, creando un sistema completo sobre un chip programable (*SOPC: System-On-a-Programmable-Chip*). Los fabricantes de FPGAs, han comenzado a incorporar *embedded silicon* ideales para aplicaciones DSP, tal como memoria embebida, bloques DSP, y procesadores embebidos, los cuales son bien adecuados para implementar funciones DSP tales como filtros FIR, FFTs, correlatores, ecualizadores, decodificadores, codificadores, y funciones aritméticas.

La Figura 1 muestra varios bloques funcionales embebidos en los circuitos FPGAs de Altera, los cuales están disponibles para aplicaciones DSP.

□ Bloques DSP embebidos

Los bloques DSP embebidos en los circuitos FPGAs, también suministran otras funciones tales como acumulación, adición-substracción y sumatoria, las cuales son operaciones aritméticas comunes en funciones DSP. Por ejemplo, los bloques DSP del dispositivo Stratix ofrecen hasta 224 multiplicadores que pueden llevar a cabo 224 multiplicaciones en un solo ciclo de reloj.

Comparado los FPGAs con los procesadores DSP que solo ofrecen un número reducido de multiplicadores, estos ofrecen un mayor ancho de banda para la multiplicación (*multiplier-bandwidth*).

Debido a que un factor determinante del ancho de banda DSP total (*overall-DSP-bandwidth*) es el *multiplier-bandwidth*, el *overall-DSP-bandwidth* para los FPGAs puede ser mucho mayor que para los procesadores DSP. Por ejemplo, los bloques DSP del dispositivo Stratix pueden entregar 70 GMACS de throughput mientras los mas avanzados procesadores DSP disponibles hoy en día, pueden entregar solamente hasta 4.8 GMACS [1].

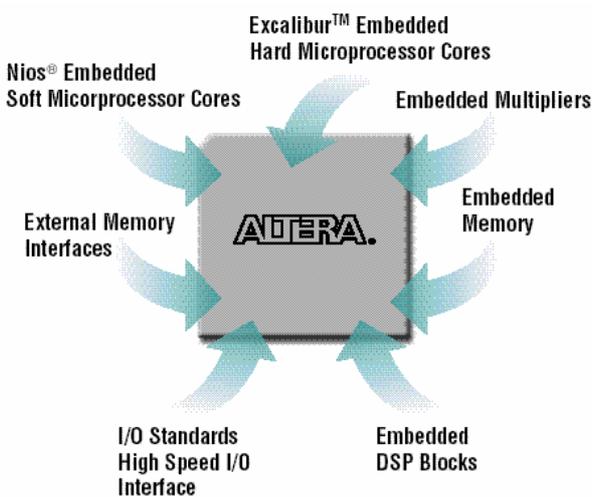


Figura 1. Embedded silicon para aplicaciones DSP

❑ Memoria embebida

De otro lado, varias aplicaciones DSP usan memoria externa para manejar una gran cantidad de datos para el procesamiento. La memoria embebida en los FPGAs reúne estos requerimientos y también elimina la necesidad de memoria externa en ciertos casos. Por ejemplo, la familia de dispositivos Stratix ofrecen hasta 10Mbits de memoria embebida.

4. LIBRERÍA VHDL PARA APLICACIONES DSP

En esta sección del artículo, el objetivo básico es ilustrar las diferentes sub-librerías que se pueden diseñar usando VHDL para aplicaciones DSP. En este contexto, las nuevas metodologías de diseño de sistemas electrónicos integrados adoptan estrategias basadas en usar módulos de propiedad intelectual (*IP: Intellectual Property*), es decir, los diseñadores concentran sus esfuerzos en diseñar y validar módulos IP para ser re-usados en diferentes aplicaciones. Entonces, en nuestro caso, el objetivo es diseñar algunos módulos IP para ser utilizados en

aplicaciones DSP. La experiencia ha demostrado que muchos bloques típicos requeridos en tales aplicaciones pueden ser implementados desde una librería de bloques funcionales pre-diseñados y parametrizados. Los bloques pueden ser usados para implementar funciones tales como procesadores FFT, transformada DCT-IDCT, filtros FIR, filtros IIR, decodificador-codificador Viterbi, etc.

Esta aproximación suministra un medio eficiente para trabajar a nivel de silicio, la cual permite a los ingenieros no especialistas diseñar chips DSP avanzados y permite a los mas experimentados ingenieros de sistemas en silicio crear nuevos diseños para rigurosas especificaciones.

Una librería VHDL estructurada y jerárquica puede ser implementada usando cinco sub-librerías [5], tal como se muestra en la Figura 2.

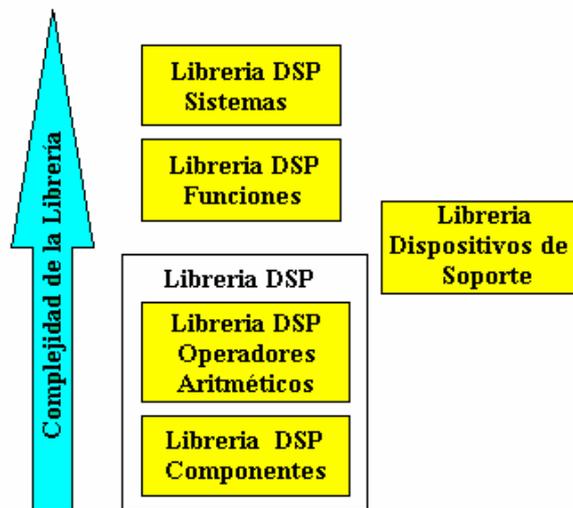


Figura 2. Jerarquía de una librería VHDL

Los bloques funcionales de los niveles superiores de la librería han sido diseñados usando bloques funcionales de los niveles inferiores. Esto significa que la optimización de los bloques en los niveles inferiores es llevada a cabo para diseños de alto desempeño por unidad de área y bajo consumo de potencia.

❑ Librería DSP: Componentes básicos

Esta librería consiste de bloques funcionales básicos que tienen en cuenta la longitud y el formato de la palabra de datos. Por ejemplo, tipo de aritmética, longitud de la palabra, tamaño del contador. Algunos componentes básicos son:

- contadores
- comparadores

- elementos de retardo
- formateadores de datos
- convertidores de datos
- memorias
- desplazadores (shifters)

□ **Librería DSP: Operadores aritméticos**

La librería de operadores aritméticos contiene operadores tales como sumadores, multiplicadores, divisores, procesadores de raíz cuadrada, etc. En este caso, el grado de parametrización es principalmente en términos de longitud de la palabra y diferentes tipos de estructuras aritméticas. Por ejemplo, los sumadores *ripple-carry*, *carry-ski*, *carry-look-ahead*, *carry-save*, *carry-select* [6]; y multiplicadores secuenciales (algoritmo de Booth) y multiplicadores paralelos (*array-multiplier*, *Wallace-tree*, *Baugh-Wooley*, *redundant-binary-adder-tree*) [6][7]. Los diseños de estos circuitos aritméticos para operándos de 32 bits son presentados en [8] y [9].

Diferentes arquitecturas para cada bloque funcional pueden ser consideradas con el propósito de cubrir los formatos de la aritmética y la palabra de datos (IEEE punto flotante, complemento a dos). Algunos operadores aritméticos son:

- multiplicadores
- acumuladores
- bloques MAC
- operador raíz cuadrada
- divisores
- sumadores

□ **Librería DSP: Funciones**

Esta librería tiene un amplio rango de aplicaciones. En este caso, una variedad de parámetros son típicamente usados con los detalles de la función específica. Algunos parámetros son:

- Longitud de la palabra de datos
- Formato de la palabra de datos
- Nivel de truncamiento
- Nivel de pipelining
- Taps del filtro
- Tamaño de la transformada

Esto permite un alto grado de flexibilidad para adaptar diseños con requerimientos de desempeño muy específicos. En particular, los bloques permiten varias alternativas de diseño para ser exploradas. Por ejemplo, es posible diseñar rápidamente filtros FIR y filtros IIR parametrizables usando diferentes intercambios de bloques multiplicador-acumulador de la librería de

operadores aritméticos. De otro lado, también es posible usar uno u otro filtro pre-diseñado de la librería, en este caso los detalles son transparentes al usuario. Los bloques típicos son:

- filtros FIR - IIR
- DFT
- DCT
- FFT
- filtros LMS
- Reed-Solomon
- decodificador Viterbi

□ **Librería DSP: Sistemas**

Esta librería se encuentra en el nivel mas alto de la jerarquía de la librería VHDL, y contiene bloques a nivel de sistema. Algunos sistemas típicos en este nivel son:

- MPEG
- JPEG
- H.261
- ADPCM
- PRML
- Detección-reconocimiento de objetos

□ **Librería dispositivos de soporte**

Esta librería suministra las interfaces para los diferentes componentes externos. Algunos dispositivos son:

- controladores de memoria
- interfaces para microprocesadores
- controladores de bus
- controladores gráficos

5. METODOLOGÍA DE DISEÑO PARA FILTROS FIR SIMPLES

Esta sección describe una metodología simple para diseñar filtros FIR dedicados para aplicaciones no críticas, y las estrategias usadas para simular e implementar el diseño en un simple circuito CPLD-FPGA. La metodología de diseño consiste en:

- Diseño a nivel de software usando MATLAB
- Diseño de la arquitectura a nivel de hardware
- Descripción del hardware usando VHDL
- Síntesis del hardware
- Simulación y verificación del hardware usando MAX+plus II
- Implementación del diseño usando un circuito CPLD-FPGA

- Test del circuito diseñado usando un sistema DSP *front-end*

□ Diseño de filtros usando MATLAB

La primera etapa de la metodología de diseño consiste en diseñar el filtro FIR a partir del diseño de un filtro análogo a nivel de software usando MATLAB. En este caso se diseñaron filtros Butterworth paso bajo de segundo y tercer orden con $F_c = 2.5\text{kHz}$ y periodo de muestreo $T = 13.2\mu\text{s}$. La Figura 3 muestra la respuesta al impulso $g(t)$ para el filtro Butterworth de tercer orden, el cual tiene 32 coeficientes.

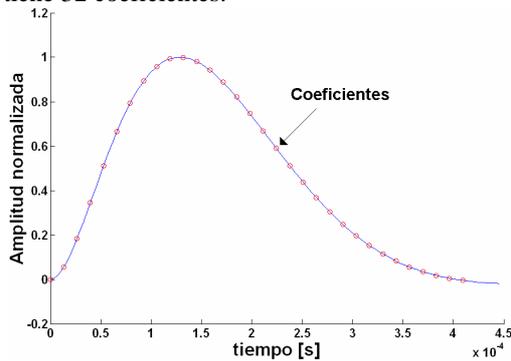


Figura 3. Respuesta al impulso $g(t)$ para el filtro Butterworth paso bajo de tercer orden

Desde la Figura 3 se observa que la respuesta al impulso $g(t)$ se acerca asintóticamente a cero a medida que el tiempo transcurre, entonces se asumen valores $g(kT) = 0$ para $k > 31$. El tiempo entre cada uno de los coeficientes es $T = 13.2\mu\text{s}$, lo cual permite una frecuencia de muestreo $f = 1/T = 75.7\text{kHz}$ y una frecuencia máxima para señal de entrada $f_{\text{max}} = f/2.2 = 34.32\text{kHz}$.

Con el propósito de verificar el funcionamiento de los filtros, varias simulaciones en MATLAB fueron llevadas a cabo. Una señal senoidal fue usada como señal de entrada, la cual trabaja a diferentes frecuencias: 1kHz, 2.5kHz, 5kHz, 7.5kHz y 75kHz. La última frecuencia no cumple con la condición de $f_{\text{max}} = 34.32\text{kHz}$. La Figura 4 muestra las señales de salida para el filtro Butterworth paso bajo de tercer orden con $F_c = 2.5\text{kHz}$.

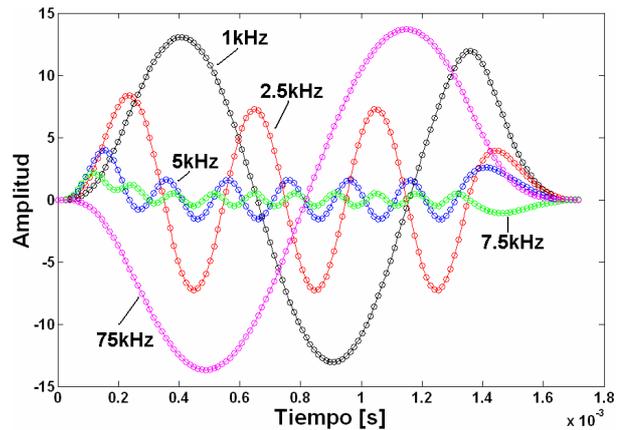


Figura 4. Señales de salida para el filtro Butterworth paso bajo de tercer orden

□ Arquitectura del filtro FIR programable

Con el propósito de diseñar la arquitectura del filtro, la idea inicial es ilustrar una arquitectura a nivel de hardware para un filtro FIR típico. La Figura 5 muestra la arquitectura de un filtro FIR típico de 8-bit, 8-tap [2][4].

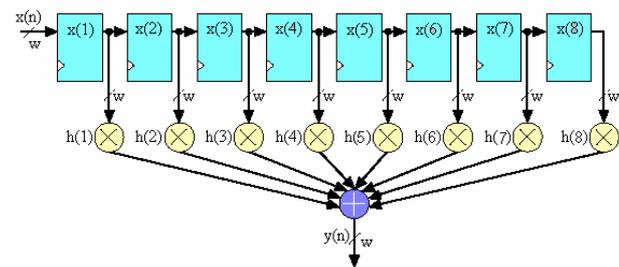


Figura 5. Arquitectura típica para un filtro FIR

En este caso, el filtro tiene ocho multiplicadores, ocho registros de 8 bits organizados en una configuración de registros de desplazamiento (*shift-registers*), y un sumador completo de 8 datos de entrada cada uno de 8 bits. Cada registro contiene la muestra anterior (*unit sample delay*), y la salida de cada registro es una etapa (TAP) y es representada por $x(n)$, donde n es el número del TAP. Cada señal de salida de la etapa (delay inputs) es multiplicada por un coeficiente $h(n)$ y todos los productos son sumados para producir la salida del filtro. La operación del filtro FIR puede ser representada por la ecuación:

$$y(n) = \sum_{n=1}^8 x(n)h(n)$$

Donde $x(n)$ y $h(n)$ son los valores de la n -ésima muestra de la señal de entrada y los coeficientes del filtro.

Debido a que la arquitectura anterior emplea una área considerable para el hardware, la idea es diseñar una arquitectura que use la menor área posible puesto que la velocidad no es un parámetro crítico. El diseño de la arquitectura a nivel de hardware para el filtro FIR con coeficientes programables es mostrado en la Figura 6, y esta es obtenida a partir de la implementación en forma directa de la siguiente expresión [10][11]:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$$

La arquitectura consiste de dos bloques principales, los cuales son: la unidad operativa y la unidad de control. La unidad operativa realiza el procesamiento de la información, es decir, decodificación del código de entrada (convertor ADC), almacenamiento de los coeficientes $h(k)$, almacenamiento de la historia de la señal de entrada $x(n-k)$, realiza las multiplicaciones y sumas, y codificación de la salida (convertor DAC). La unidad de control ejecuta la secuencia del procesamiento de la unidad operativa y hace que esta opere en forma concurrente con los convertidores, permitiendo que durante el cálculo de $y(n)$ se realicen de forma simultánea las conversiones A/D y D/A para obtener $x(n+1)$ y $y(n-1)$ respectivamente.

La arquitectura del FIR es programable debido a que dispone de un puerto COEFF (8 bits) para ingresar los 32 coeficientes $h(k)$ y de la RAM (32 posiciones 8 bits) para

almacenar los $h(k)$. La señal de entrada $x(n)$ del FIR se ingresa a través del puerto D_IN, la cual es codificada en Signo-Magnitud (SM) por el bloque CONVERTOR_IN. El valor actual de la entrada $x(n)$ y la historia $x(n-k)$, correspondiente a los 31 valores muestreados con anterioridad a $x(n)$, son almacenados en el bancos de REG_DESP_INST.

Para realizar el cálculo de $y(n)$, se realizan 32 multiplicaciones de las parejas ($h(k)$, $x(n-k)$), estas multiplicaciones se realizan en el bloque MULTIPLICADOR de forma secuencial. La k -ésima multiplicación se realiza direccionando la posición k del bloque RAM y la posición k de entrada del bloque MUX, cuyo resultado se obtiene en la salida MULT_OUT. Los dos bloques SUMADOR y REG_A corresponden a un circuito acumulador, el cual almacena los 32 resultados calculados por el bloque MULTIPLICADOR. El registro acumulador almacena $y(n)$ con un formato de 19 bits, el cual se convierte a un formato de 8 bits usando el bloque REDONDEAR. Entonces, $y(n)$ es codificado para ser operado por el convertor DAC. Este resultado de $y(n)$ se almacena en el bloque REG_Y y corresponde al puerto de salida del FIR. Una vez finalizado el cálculo del valor $y(n)$, el FIR está listo para recibir un nuevo dato $x(n)$, repitiéndose el proceso anterior, pero desplazando los datos almacenados en el bloque REG_DESP_INST, según el flujo de datos del banco de registros (ver Figura 6).

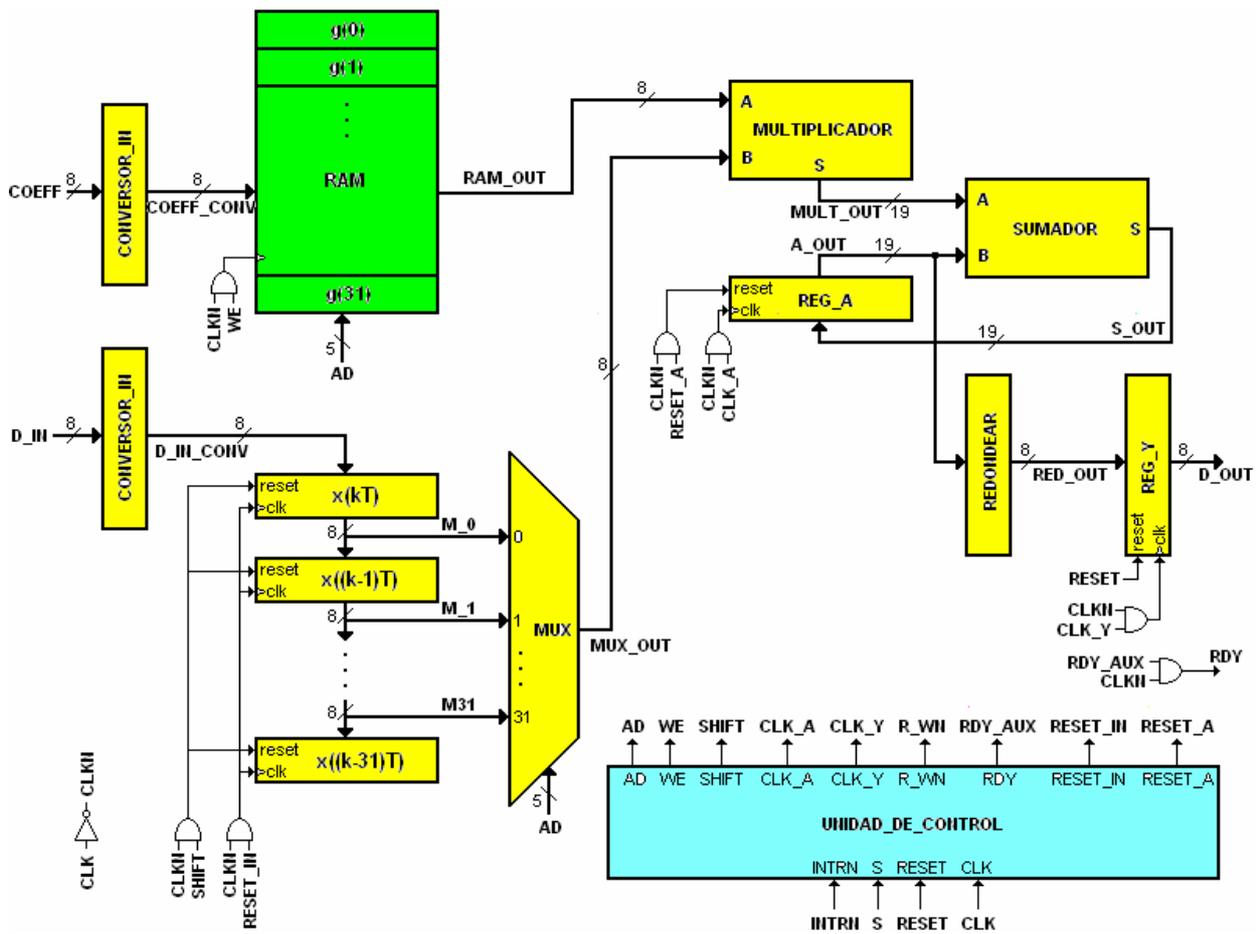


Figura 6. Arquitectura dedicada para un filtro FIR programable

Resultados de Simulación

Con el propósito de verificar el funcionamiento del hardware del filtro FIR (descrito en VHDL e implementado sobre el FLEX EPF10K70RC240-4), se utilizó el simulador MAX+plus II de Altera. Sin embargo, este es un simulador digital, por lo cual no es práctico generar una señal sinusoidal como señal de entrada para simular un filtro digital. Entonces, la idea es simular el comportamiento del filtro FIR para una entrada tipo escalón y observar su comportamiento. Los resultados de simulación entregados por el MAX+plus II son

adecuados como código de entrada para un convertidor DAC, entonces con el propósito de obtener la gráfica de la respuesta al escalón del filtro FIR programable, los datos deben ser normalizados y procesados usando MATLAB. La Figura 7 muestra los respectivos resultados de la simulación obtenidos en el MAX+plus II y la Figura 8 muestra la gráfica de la respuesta al escalón del filtro FIR desde los datos normalizados del MAX+plus II y la respuesta al escalón del filtro análogo simulado con MATLAB.

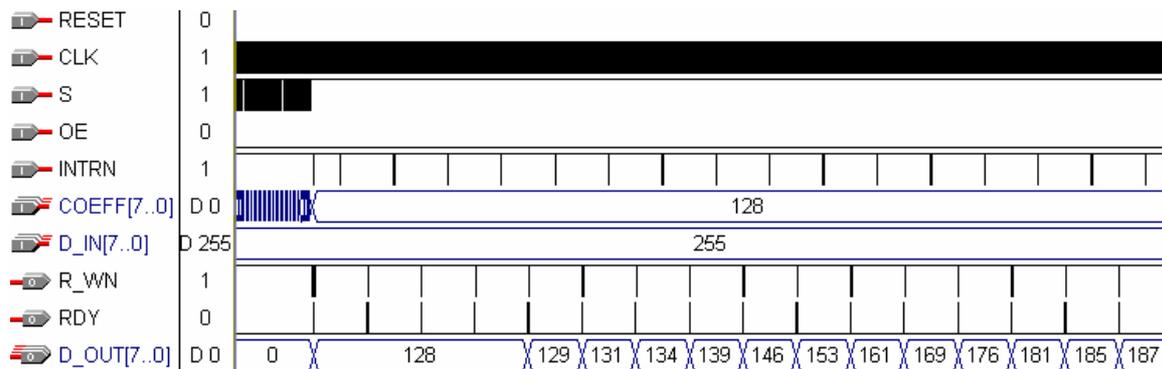


Figura 7. Resultados de simulación del filtro FIR

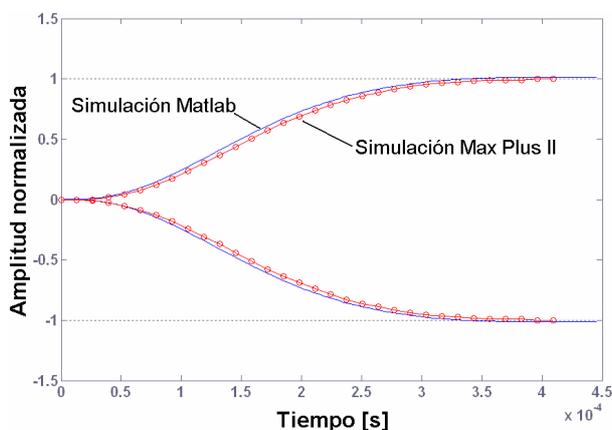


Figura 8. Grafica de la respuesta al escalón para el filtro FIR y el filtro análogo

En este caso, los resultados de simulación verifican el correcto funcionamiento del filtro FIR diseñado.

□ Resultados Experimentales

Para realizar el test del filtro FIR, se implemento un sistema DSP front-end, es decir, se utilizaron los respectivos conversores ADC y DAC, y la tarjeta UP-1X de Altera (ver Figura 9).

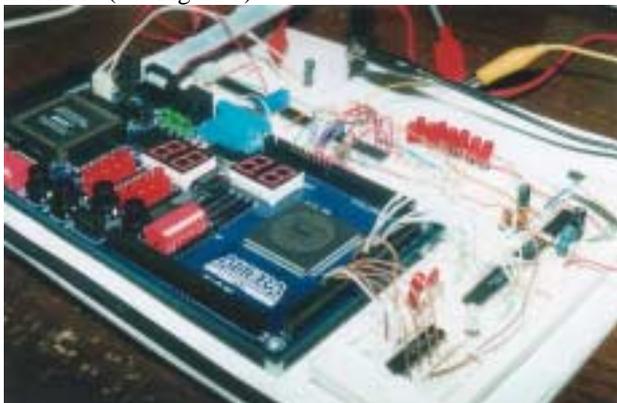


Figura 9. Sistema DSP front-end para el test del filtro FIR

Con el propósito de verificar el comportamiento del filtro usando señales en tiempo real, se observaron las respectivas señales de entrada y salida en un osciloscopio.

El filtro FIR configurado sobre el CPLD se conectó a los conversores A/D y D/A usando los puertos de entrada y salida, es decir, el puerto D_IN se conectó a un conversor ADC 0804 y el puerto D_OUT se conectó a un conversor DAC 0808. Debido a la baja frecuencia de muestreo del conversor análogo digital comparada con la del FIR, no fue posible usar señales de frecuencia superior a 34.32 KHz.

En las Figuras 10, 11 y 12 se muestran las señales observadas en el osciloscopio, donde la línea continua corresponde a la entrada $x(t)$ del FIR y la línea escalonada corresponde a la salida $y(t)$. En la Figura 10 se muestra como el FIR permite el paso de una señal de entrada $x(n)$ con frecuencia por debajo de la frecuencia de corte (f_c) y en las Figuras 11 y 12 se observa como el FIR impide el paso de una señal de entrada $x(n)$ con frecuencia superior a la f_c .

Las señales de salida observadas en el osciloscopio verifican el correcto funcionamiento del filtro FIR diseñado y de la metodología de diseño.



Figura 10. Señales de entrada y salida observadas en el osciloscopio con $f < f_c$

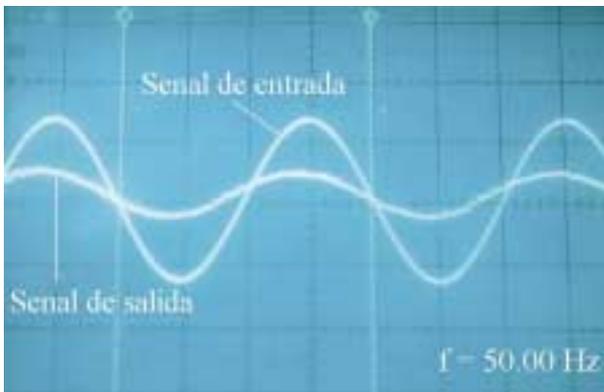


Figura 11. Señales de entrada y salida observadas en el osciloscopio con $f > f_c$

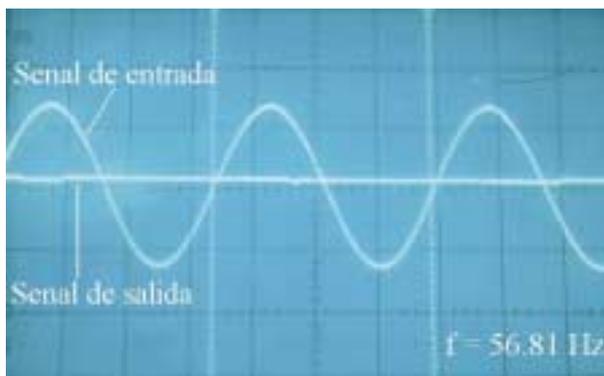


Figura 12. Señales de entrada y salida observadas en el osciloscopio con $f > f_c$

6. CONCLUSIONES Y TRABAJO FUTURO

Una metodología simple para diseñar bloques funcionales DSP básicos para aplicaciones no críticas es presentada en este trabajo. En este caso, el proceso de diseño se

inicia con el diseño a nivel de software usando MATLAB, luego se concibe la arquitectura y se captura el diseño del hardware usando VHDL, posteriormente se realizan las respectivas simulaciones usando MAX+plus II y se implementa el diseño en un simple circuito CPLD-FPGA. Finalmente, se realiza el test de los diseños usando un sistema DSP front-end.

Como vehículo de test se diseñaron filtros FIR simples y los resultados de simulación y experimentales permiten validar la metodología de diseño propuesta.

La idea básica de este trabajo fue concebir y validar una metodología simple y económica para diseñar bloques funcionales DSP básicos usando MAX+plus II (versión estudiantil) y circuitos CPLDs. En este orden de ideas, el trabajo futuro debe ser concentrado en el diseño bloques funcionales DSP complejos usando los programas DSP Builder y Quartus II, y los circuitos Stratix FPGA de Altera. Estos bloques deben ser diseñados usando mínima área, máxima velocidad y arquitecturas parametrizables con el propósito de obtener módulos de propiedad intelectual para ser incorporados en una librería VHDL. Por ejemplo, los bloques DSP para diseñar en el futuro son: filtros FIR, filtros IIR, transformada DCT-IDCT, transformada FFT, filtros LMS y decodificador-codificador Viterbi.

7. AGRADECIMIENTOS

Este trabajo ha sido patrocinado por Altera Corporation a través del Programa Universitario. Los autores dan especial agradecimientos a Mrs. Ralene Marcoccia de Altera Corporation.

8. REFERENCIAS

- [1.] Altera Corp, "FPGAs Provide Reconfigurable DSP Solutions", White Paper, August 2002, version 1.0.
- [2.] Altera Corp, "Using PLDs for High-Performance DSP Applications", White Paper, February 2002, version 1.0.
- [3.] Andraka Consulting Group, "DSP with FPGAs", web page, www.andraka.com/dsp.htm
- [4.] Altera Corp, "Implementing FIR Filters in FLEX Devices", Application Note 73, February 1998, version 1.01
- [5.] McCanny J.V, Ridge D, Hu Y, and Hunter J, "Hierarchical VHDL Libraries for DSP ASIC Design", Proc. IEEE International Conference On Acoustic Speech and Signal Processing, Munich, April 1997.
- [6.] Wai-Kai Chen, "The VLSI Handbook", CRC press and IEEE press, 2000.
- [7.] Neil Weste and Kamran. Eshraghian, "Principles of CMOS VLSI Design: A Systems Perspective", 2nd ed. Addison Wesley, Copyright 1993 by AT&T.

- [8.] Vera L. M.E, “Curso Diseño Digital con VHDL”, Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Junio 2002.
- [9.] Velasco M. J, “Curso Diseño de Arquitecturas Digitales RISC-DSP”, Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali, Junio, 2002.
- [10.] Proakis J. y Manolakis D., “Tratamiento digital de señales”, Prentice Hall, 1998.
- [11.] Poularikas A., Chapter 14:“Digital Signal Processing” “The Electrical Engineering Handbook” Ed. Richard C. Dorf, CRC Press, 2000

9. AUTORES



Mario E. Vera Lizcano. Ingeniero Electricista y Magister en Automática de la Universidad del Valle. Profesor Asistente de la Escuela de Ingeniería Eléctrica y Electrónica de la Universidad del Valle



Gustavo A. Vejarano. Estudiante de octavo semestre del programa académico de Ingeniería Electrónica de la Universidad del Valle.



Jaime Velasco Medina. Ingeniero Electricista de la Universidad del Valle. Doctor en Microelectrónica del Instituto Nacional Politécnico de Grenoble, Francia, 1999. Profesor Titular de la Escuela de Ingeniería Eléctrica y Electrónica de la Universidad del Valle.