Implementación de Filtros Digitales Tipo FIR en FPGA

Jesús Antonio Álvarez Cedillo, Klauss Michael Lindig Bos, Gustavo Martínez Romero

***Resumen*—En este artículo se hace la descripción del diseño de un filtro digital tipo FIR con ocho bits de ancho de datos. Este sistema ha sido implementado en un FPGA (SPARTAN 3E de XILINX) y posee un software que realiza el cálculo de los coeficientes del filtro y la reconfiguración del hardware. Las pruebas se realizaron usando el programa MATHLAB para verificar su funcionamiento.**

***Palabras clave*—Filtros digitales, tratamiento digital de señales, FPGA, VHDL, FIR.**

IMPLEMENTATION OF DIGITAL FILTERS OF FIR TYPE IN FPGA

***Abstract*—This paper presents the description of development of digital filter of FIR type with eight bits data transmission. This system was implemented in FPGA (SPARTAN 3E by XILINX) and includes the software for calculation of filter coefficients and hardware reconfiguration. The experiments were conducted using simulation in MATHLAB.**

***Index Terms*—Digital filter, digital signal processing, FPGA, VHDL, FIR.**

I. INTRODUCCIÓN

Un filtro es un sistema que, dependiendo de algunos parámetros, realiza un proceso de discriminación de una señal de entrada obteniendo variaciones en su salida. Los filtros digitales tienen como entrada una señal analógica o digital y a su salida tienen otra señal analógica o digital, pudiendo haber cambiado en amplitud, frecuencia o fase dependiendo de las características del filtro.

El filtrado digital es parte del procesado de señal digital. Se le da la denominación de digital más por su funcionamiento interno que por su dependencia del tipo de señal a filtrar, así podríamos llamar filtro digital tanto a un filtro que realiza el procesado de señales digitales como a otro que lo haga de señales analógicas.

El filtrado digital consiste en la realización interna de un procesado de datos de entrada. El valor de la muestra de la entrada actual y algunas muestras anteriores (que previamente habían sido almacenadas) son multiplicadas por unos coeficientes definidos. También podría tomar valores de la salida en instantes pasados y multiplicarlos por otros coeficientes. Finalmente todos los resultados de todas estas multiplicaciones son sumados, dando una salida para el instante actual. Esto implica que internamente tanto la salida como la entrada del filtro serán digitales, por lo que puede ser necesario una conversión analógico-digital o digital-analógico para uso de filtros digitales en señales analógicas. Un elemento de prueba de estos circuitos típicamente es el ruido blanco (Figura 1).

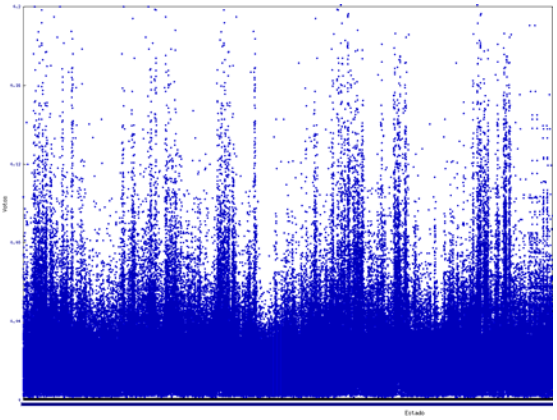

Fig. 1. Gráfica de ruido blanco

II. FILTROS FIR

Los filtros digitales se usan frecuentemente para tratamiento digital de la imagen o para tratamiento del sonido digital.

FIR es un acrónimo en inglés para *Finite Impulse Response* o *Respuesta finita al impulso*. Se trata de un tipo de filtros digitales en el que, como su nombre indica, si la entrada es una señal impulso, la salida tendrá un número finito de términos no nulos.

Para obtener la salida sólo se basan en entradas actuales y anteriores. Su expresión en el dominio **n** es:

Manuscrito recibido el 13 de marzo del 2008. Manuscrito aceptado para su publicación el 16 de junio del 2008.

J. A. Alvarez Cedillo, Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional, México, D. F. (teléfono: 57296000 Ext. 52535; e-mail: jaalvarez@ipn.mx).

K. M. Lindig Bos, Dirección de Cómputo y Telecomunicaciones del Instituto Politécnico Nacional, México, D. F. (teléfono: 57296000 Ext. 52536; e-mail: mlindig@ipn.mx).

G. Martínez Romero, Centro de Investigación e Innovación Tecnológica Unidad Azcapotzalco del Instituto Politécnico Nacional, México, D. F. (teléfono: 57296000 Ext. 52535; e-mail: gumartinezr@ipn.mx).

$$y_n = \sum_{k=0}^{N-1} b_k x(n-k)$$

En la expresión anterior **N** es el orden del filtro, que también coincide con el número de términos no nulos y con el número de coeficientes del filtro. Los coeficientes son **bk.**

La salida también puede expresarse como la convolución de la señal de entrada *x(n)* con la respuesta impulsional *h(n)*:

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k}$$

Aplicando la transformada Z a la expresión anterior:

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} = h_0 + h_1 z^{-1} + \cdots + h_{N-1} z^{-(N-1)}$$

La estructura básica de un FIR se presenta en la Fig. 2.

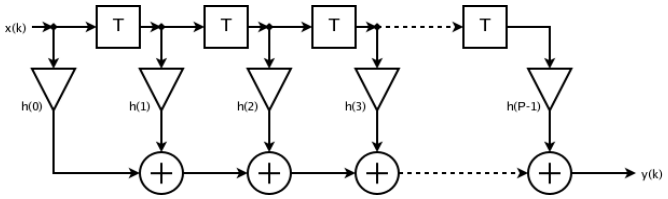

Fig. 2. Estructura básica de un FIR

En la figura 2 los términos **β** son los coeficientes y los **T** son retardos.

Pueden hacerse multitud de variaciones de esta estructura. Hacerlo como varios filtros en serie, en cascada, etc.

Hay tres métodos básicos para diseñar este tipo de filtros:

- Método de las ventanas. Las más habituales son:
  - Ventana rectangular
  - Ventana de Barlett
  - Ventana de Hanning
  - Ventana de Hamming
  - Ventana de Blackman
  - Ventana de Kaiser
- Muestreo en frecuencia.
- Rizado constante,

III. Antecedentes

Como ya se mencionó en líneas anteriores, es gracias al avance de las computadoras modernas que el tratamiento digital de señales se ha expandido y se ha hecho cada vez más fuerte. Han contribuido con su velocidad al montaje de algoritmos de procesamiento para lograr entregar respuestas casi instantáneas, siempre dentro de los límites exigidos por el desarrollo en las diferentes aplicaciones en que han sido utilizadas.

Las computadoras no sólo han influido en el montaje de algoritmos, además se han convertido en una herramienta fundamental para los diseñadores, ya que gracias a éstos se ha conseguido elaborar mejores y variados algoritmos para el diseño de filtros y herramientas de proceso. Una herramienta muy potente para el análisis de imágenes es el MATHLAB.

El diseño de filtros se ha apoyado en los dispositivos lógicos programables, los cuales han jugado un papel muy importante en el montaje de los filtros digitales, puesto que gracias a ellos se ha logrado un adecuado funcionamiento en tiempo real. El FPGA es uno de estos dispositivos, que posee la cualidad de la re-configuración, lo que permite realizar cambios en la arquitectura sin necesidad de producir variaciones en el montaje o en el software que se está operando.

IV. Factores de Implementación

La implementación de estos filtros está determinada por algunos factores que ayudan a la calificación de dichos sistemas, tales como:

1. Complejidad computacional, requisitos de memoria y longitud de palabra.
2. La Complejidad computacional: está determinada por el número de operaciones aritméticas necesarias para el cálculo de la salida, como sumas, multiplicaciones y divisiones.
3. Requisitos de memoria: hacen referencia a la cantidad de posiciones de memoria que son necesarias para almacenar elementos, tales como los coeficientes del sistema, entradas retrasadas, salidas retrasadas y algunos valores internos necesarios para el cálculo de la salida.
4. Longitud de palabra: se refiere a un efecto de precisión que se encuentra dado por la cuantificación, tanto de los coeficientes del filtro como de la señal de entrada. Este elemento se hace presente en filtros implementados en hardware y en software.

Las operaciones realizadas deben ser redondeadas o truncadas para poder ajustarse a las restricciones de operación del ordenador, en el caso del software, o a las características definidas por el diseñador del hardware digital.

V. Tipos de Filtros

Existen dos tipos básicos de filtros digitales:

- no recursivos
- recursivos.

Para los filtros no recursivos la función de transferencia contiene un número finito de elementos, cuya ecuación en diferencias es:

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

Y su equivalente en función de transferencia es:

$$H(z) = \frac{\sum_{k=0}^{M} a_k z^{-k}}{1 - \sum_{k=1}^{N} b_k z^{-k}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{1 - b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}}$$

Esta clase de sistemas se caracteriza por no poseer realimentaciones, de lo cual se observa que la salida se encuentra dada en función de la entrada y de sus respectivos retrasos.

Para los filtros recursivos la ecuación en diferencias se encuentra expresada en función de dos formas polinomiales:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

Esta ecuación nos lleva a encontrar una función de transferencia de la forma:

$$H(z) = \frac{\sum_{k=0}^{M} a_k z^{-k}}{1 - \sum_{k=1}^{N} b_k z^{-k}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{1 - b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}}$$

A los primeros pertenecen los filtros tipo FIR, caracterizados por no poseer realimentación, y a los segundos los filtros tipo IIR, en donde la salida se encuentra dada en función de la entrada y de las salidas en instantes previos.

VI. Implementación en el FPGA

Se creó una herramienta en hardware como filtro tipo FIR, esta fue probada en una tarjeta XILINX SPARTAN 3E. El kit de desarrollo se muestra en la figura 3.

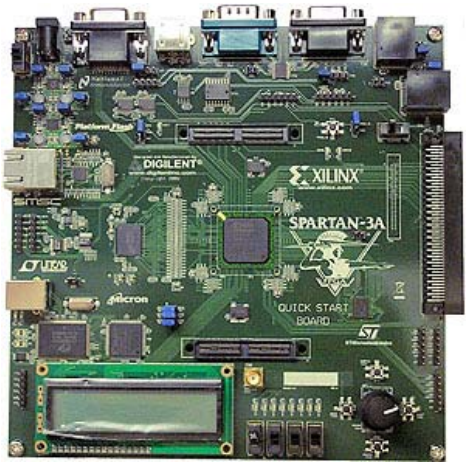

Fig. 3. Kit de desarrollo XILINX SPARTAN 3E

La entidad fue definida de la siguiente forma:

```
ENTITY FIR IS   GENERIC (N : natural := 16  );
PORT (   SIGNAL CLK : IN std_logic;
SIGNAL RES_n : IN std_logic;
SIGNAL X : IN std_logic_vector( N-1 DOWNTO 0 );
SIGNAL Y : OUT std_logic_vector( N-1 DOWNTO 0 )   );
END ENTITY FIR ;
```

Como es posible observar se mantiene la señal de reloj CLK como entrada, dos valores de ingreso donde X es dato de 2 bits y RES_n es un reset activado por pulso. La salida se encuentra definida por un valor de dato de 2 bits. Tal como se muestra en la figura 4.

Fig. 4. Entidad de un filtro Fir en FPGA XILINX

La arquitectura se define de la siguiente manera:

```
ARCHITECTURE RTL OF FIR IS
  TYPE t_operacion IS ARRAY (7 DOWNTO 1) OF std_logic_vector(N-1 DOWNTO 0);

  SIGNAL operacion : t_operacion;
  SIGNAL add_01 : std_logic_vector(N DOWNTO 0);
  SIGNAL add_23 : std_logic_vector(N DOWNTO 0);
  SIGNAL add_45 : std_logic_vector(N DOWNTO 0);
  SIGNAL add_67 : std_logic_vector(N DOWNTO 0);
  SIGNAL add_0123 : std_logic_vector(N+1 DOWNTO 0);
  SIGNAL add_4567 : std_logic_vector(N+1 DOWNTO 0);
  SIGNAL add_all : std_logic_vector(N+2 DOWNTO 0);
 SIGNAL vystup : std_logic_vector(N+2 DOWNTO 0);
 SIGNAL pom    : std_logic_vector(N+2 DOWNTO 0);
 SIGNAL pipe_0123 : std_logic_vector(N+1 DOWNTO 0);
SIGNAL pipe_4567: std_logic_vector(N+1 DOWNTO 0);
BEGIN
```

Como es posible observar las señales de operación son diversas, la idea principal es mostrar el manejo del pipeline, el circuito presentará el esquema de la figura 5.

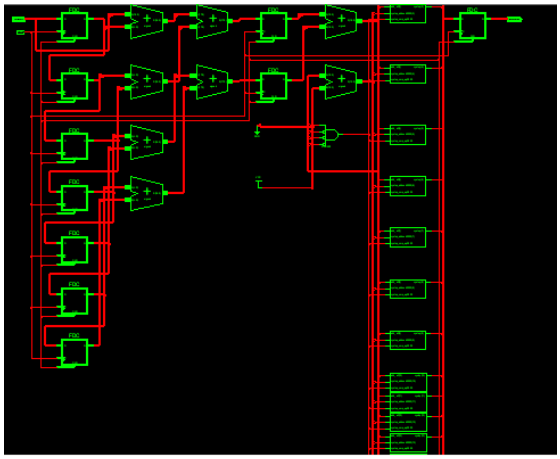

Fig. 5. Esquema RTL

El pipeline característico se codifico de la siguiente manera:
Si es presionado el reset RES_n el pipeline inicia su operación, si existe el evento del reloj y es un pulso positivo, entonces realiza las operaciones retardas y secuenciales.

```
 pipeline: PROCESS (CLK, RES_n)
 BEGIN
 IF RES_n='0' THEN
   pipe_0123 <= (OTHERS => '0');
```

```
    pipe_4567 <= (OTHERS => '0');

    ELSIF CLK='1' AND CLK'EVENT THEN
      pipe_0123 <= add_0123;
      pipe_4567 <= add_4567;
   END IF;
   END PROCESS pipeline;
```

Las siguientes instrucciones muestran el bloque de cada procedimiento u operación del pipeline, cabe resaltar el manejo de los índices en el filtro.

```
    add_01 <= (X(N-1) & X) + (operacion(1)(N-1) & operacion(1));
    add_23 <= (operacion(2)(N-1) & operacion(2)) + (operacion(3)(N-1) &
operacion(3));
    add_45 <= (operacion(4)(N-1) & operacion(4)) + (operacion(5)(N-1) &
operacion(5));
    add_67 <= (operacion(6)(N-1) & operacion(6)) + (operacion(7)(N-1) &
operacion(7));

    add_0123 <= (add_01(N) & add_01) + (add_23(N) & add_23);
    add_4567 <= (add_45(N) & add_45) + (add_67(N) & add_67);
  add_all <= (pipe_0123(N+1) & pipe_0123) + (pipe_4567(N+1) &
pipe_4567);

    pom(3 DOWNTO 0) <= "0100";
    pom(N+2 DOWNTO 4) <= (OTHERS => '0');

    vystup <= add_all WHEN add_all(3 DOWNTO 0) = "0100" else  add_all
+ pom;
```

VII. Pruebas y Resultados

Se creó una herramienta de software que se encarga de brindar una conexión sencilla y amable entre el usuario del filtro y el hardware.

La interfase se encuentra desarrollada sobre Matlab 6.0 y al inicio presenta 3 posibilidades de filtros tipo FIR para escoger con cuál se desea trabajar:

- Filtro con coeficientes estáticos.
- Filtro con coeficientes dinámicos.
- Filtro adaptativo.

Para efectos de este artículo sólo se trabaja con la opción de filtro con coeficientes dinámicos.

Esta sección escogida presenta dos bloques principales así:

Diseño del filtro: esta es la primera sección, le permite al usuario manejar las especificaciones del filtro tales como:

- Cantidad de coeficientes.
- Ancho de banda.
- Atenuación y ganancia del filtro para diferentes frecuencias, como se muestra en la figura 6.

Una vez se han escogido los diferentes parámetros del filtro, tanto para su arquitectura como para su funcionamiento, se procede a diseñar dicho filtro. Proceso que se lleva a cabo mediante la utilización del algoritmo FIR.

La verificación e implementación: se realizan una vez que se han determinado las especificaciones del filtro y se ha diseñado. Se observa una nueva ventana (figura 7), la cual muestra la respuesta en frecuencia del filtro. En esta ventana se presentan tres respuestas espectrales diferentes así:

- La deseada por el usuario.
- La lograda por el algoritmo
- La obtenida luego de redondear los valores de los coeficientes a los valores permitidos por la resolución de 8 bits.

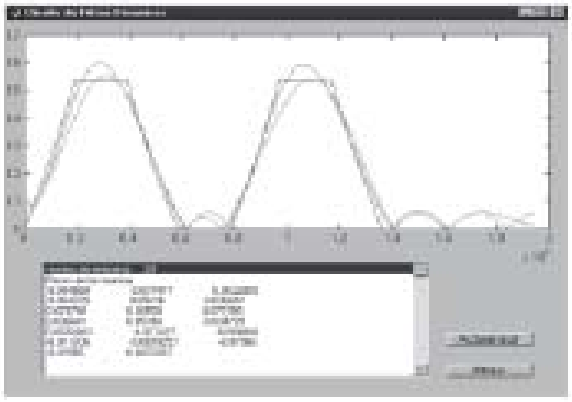

Fig. 6. Muestra de la interfase para el diseño de los filtros

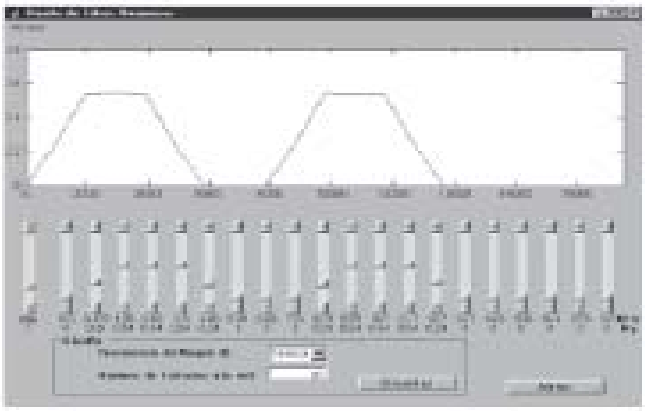

Fig. 7. Ventana de verificación e implementación del filtro

VIII. Conclusiones

El diseño se realizó en forma jerárquica, de tal manera que se dividió en diferentes etapas.

Antes de iniciar el estudio de cada una de estas entidades deberá de ser necesario tener en cuenta algunos aspectos del funcionamiento que presenta este diseño:

- El ancho de palabra es de 8 bits, lo que obliga a las memorias, al sumador y al acumulador a trabajar con esta especificación.
- Debido a que el ancho de palabra es de 8 bits, se toma como mínimo intervalo 1/128, de tal forma que el formato varía desde 0.996 hasta –1. De esta forma se logra reducir el problema de truncamiento que se experimenta cuando se trabaja con números de formato entero.
- Para el reloj del sistema se utilizó el oscilador interno del FPGA, que es de 50 Mhz, al que se le acondiciona un contador para poder dividir la frecuencia y así obtener diferentes posibilidades en el ancho de banda del filtro, como: 77.8 Khz, 39Khz, 19.5 Khz, 9.7 Khz, 4.8 Khz, 1.2Hhz, 600Hz y 150Hz. Estas frecuencias son

seleccionadas por el usuario desde el exterior de la arquitectura.

- La arquitectura presenta una etapa que se encarga de comunicarse con el software y, de esta forma, mediante una interfase de puerto paralelo a la PC.

REFERENCIAS

[1] Willis J. Tompkins. Biomedical digital signal procesing.Prentice Hall, may, 1993.

[2] Xilinx The progamable logic databook. SPARTAN3 , marzo 2 de 2007.

[3] J. G., Proakis, Dimitris G. Manolakis. Tratamiento digital de señales. Prentice Hall, 1997.

[4] Samuel Stearms, Ruth A. David. Signal procesing algoritms". Prentice Hall, 1997.

[5] Binary numbering systems. Altera Corporation, 1997

[6] R. W. Hamming. Digital filters. Prentice Hall, 1989.

[7] Using select-RAM memory in XC4000 series FPGA's". Xilinx aplication note, july 7, 1996.

[8] A CPLD VHDL introduction. Xilinx application note, january 12, 1998.

[9] Digital signal processing toolbox. The MathWorks Inc., 1992-2001.