Implementación De Filtros Digitales Tipo FIR En FPGA's Con Coeficientes Reconfigurables On-Line.

Ing. Linda J. Arcos Pantoja, Ing. Juan Manuel Calderón Chávez

***Abstract*—in this paper, a FIR digital filter is described in hardware (VHDL) with 64 tabs and wide word of 8 bits. This filter has the characteristic of reconfigurate coefficients (On-Line) without the necessity of restarting the system. The filter has eight different velocities of sampling, and they could be determinated for the user from the hardware.**

This filter has been implemented in a FPGA (SPARTAN 10XL), and also it has a software, which calculates the filter coefficients and the hardware reconfiguration.

***Index Terms*— Filtros Digitales, Tratamiento digital de señales, FPGA, VHDL , FIR.**

I. INTRODUCCIÓN

En el transcurso de las ultimas décadas, se han presentado grandes avances en el diseño de computadores, los cuales son cada vez más veloces y eficientes. Esto ha sido utilizado para diferentes fines como: el tratamiento, almacenamiento, estudio y transmisión de información en forma digital. De esta forma el tratamiento digital de señales se ha convertido hoy por hoy en una de las herramientas mas utilizadas para dar solución a algunos problemas de ingeniería.

El procesamiento digital de señales es una herramienta muy útil para la estimación de parámetros y características de sistemas y señales, eliminación o reducción de ruido e interferencias y la transformación de la respuesta espectral de señales entre otras. Los cuales repercuten en aplicaciones como el estudio de señales biomédicas para el diagnostico de enfermedades, la compresión de información para la transmisión de datos y el procesamiento de audio y video son algunas de las aplicaciones mas comunes que se encuentran hoy en día.

De lo anterior se logra vislumbrar parte de la importancia de los filtros digitales, pues estos son una de las herramientas mas utilizadas en el tratamiento digital de señales, para dar solución a un sin fin de problemas del acontecer diario.[4]

II. ANTECEDENTES

Como se menciona en líneas anteriores, es gracias al avance de los computadores modernos, que el tratamiento digital de señales se ha hecho cada vez mas fuerte. Estos han contribuido con su velocidad en el montaje de algoritmos de procesamiento para así lograr entregar respuestas casi instantáneas, permitidas dentro de los limites exigidos en el desarrollo de las diferentes aplicaciones.

Los computadores no solo han influido en el montaje de algoritmos, sino se han convertido en una herramienta fundamental para los diseñadores, los cuales gracias a estos han logrado elaborar mejores y variados algoritmos para el diseño de filtros y herramientas de proceso.

Por otro lado esta sección de la ciencia 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, el cual posee la cualidad de la re-configuración, lo que permite realizar cambios en la arquitectura sin la necesidad de producir variaciones en el montaje o software que se esta operando.

Este dispositivo esta compuesto por CLB's (Bloques Lógicos Configurables), IOB's (Bloques de entrada y salida), Y PSM's (Matrices de switcheo Programable).

En cada CLB hay unidades denominadas LUT's las cuales permiten la implementación de circuitos lógicos combinacionales sin la necesidad de matrices de AND y OR y a su vez posee Flip-Flops tipo D para el montaje de la lógica secuencial.

Los IOB's determinan la configuración de los puertos, ya sea como salida, estradas o bidireccionales. Por ultimo el PSM se encarga de la comunicación entre cada una de las estructuras anteriormente mencionadas. Por esta y algunas razones mas es que se hace posible el montaje de sistemas digitales complejos.

III. MARCO TEORICO

A. *Filtro Digital*

1) *Concepto*

En lo básico de su funcionalidad un filtro digital se comporta de igual manera que un análogo, de lo que se puede deducir que un filtro es un sistema encargado de

alterar el contenido de la información espectral de una señal de entrada X(t), produciendo una señal de salida Y(t).[4]
Los filtros análogos son implementados mediante la utilización de circuitos electrónicos activos o pasivos y operan sobre formas de onda continuas.
Los filtros digitales por otra parte son implementados mediante la utilización de circuitos lógicos o en programas de computador. Estos operan sobre una secuencia de números que son obtenidos por el muestreo de ondas continuas.[1]
Los filtros digitales gozan hoy en día de una gran popularidad y un extendido uso gracias a la facilidad que se presenta para montar estos diseños en los computadores modernos o en otros casos son diseñados e implementados en circuitos lógicos programables como el FPGA y CPLD.

2. VENTAJAS DE LOS FILTROS DIGITALES

Existen bastantes ventajas de los filtros digitales sobre los filtros análogos y estas son algunas de ellas:

- Un filtro digital es altamente inmune al ruido, gracias a la forma en que se implementa (Software o Circuito digital).
- La precisión de este filtro depende exclusivamente del error de redondeo, el cual es determinado directamente por el número de bits que escoge el diseñador para representar las variables en el filtro.
- Es muy fácil y barato cambiar las características de operación del filtro, esto a diferencia de los filtros análogos donde se requiere toda una reestructuración del hardware.
- A diferencia de un filtro análogo, su desempeño no se encuentra dado en función de la precisión o el deterioro de sus componentes, de las variaciones de la temperatura o de las variaciones de la fuente.[6]

3. CARACTERIZACION DE LOS FILTROS DIGITALES

Esta clase de sistemas se encuentra caracterizada por una ecuación lineal en diferencias con coeficientes constantes como se muestra en la ecuación 1.

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

Estos sistemas, mediante la transformada Z se pueden caracterizar como una función de transferencia racional (2) así:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \qquad (2)$$

Al observar las ecuaciones anteriores se puede apreciar que los polos y los ceros del sistema se encuentran dados por los $\{b_k\}$ y $\{a_k\}$, los cuales determinan las características de la respuesta en frecuencia del sistema.
De las ecuaciones anteriores es posible construir un diagrama de bloques conformado por elementos de retardo, multiplicadores y sumadores (Figura. 1).

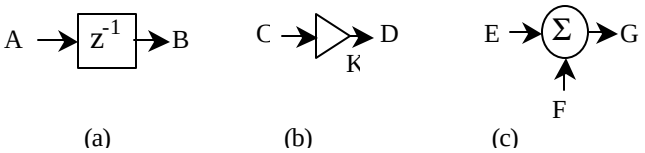

Figura 1. Operadores de un Filtro Digital. (a) Retardo de un tiempo de reloj. (b) Multiplicación por una constante. (c) Suma de dos números.

Este diagrama de bloques puede ser plenamente utilizado para el diseño de un software que rueda sobre un ordenador digital o puede ser base en la configuración de un hardware que sirva para la implementación de dicho sistema.[1]

4. FACTORES DE MPLEMENTACION

La implementación de estos filtros se encuentra enmarcada dentro de algunos factores que ayudan a la calificación de dichos sistemas, tales como: complejidad computacional, requisitos de memoria, longitud de palabra.

Complejidad computacional: Esta determinada por el número de operaciones aritméticas tales como sumas, multiplicaciones y divisiones que son necesarias para el calculo de la salida.

Requisitos de memoria: Se hace referencia a la cantidad de posiciones de memoria que son necesarias para almacenar los coeficientes del sistema, entradas retrasadas, salida retrasadas y algunos valores internos necesarios para el calculo de la salida.

Longitud de la palabra: Este es un efecto de precisión el cual se encuentra dado por la cuantificación, tanto de los coeficientes del filtro como el de la señal de entrada y se hace presente tanto en filtros implementados en hardware y 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 de las características definidas por el diseñador del hardware digital.[3]

5. TIPOS DE FILTROS

Existen dos tipos básicos de filtros digitales, lo cuales son: No-recursivos y recursivos.
Para los filtros recursivos la función de transferencia contiene un número finito de elementos y por consiguiente la ecuación en diferencias es:

$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k) \quad (3)$$

Y su equivalente en función de transferencia es:

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k} \quad (4)$$

Esta clase de sistemas se caracteriza por no poseer realimentaciones de lo cual se aprecia que la salida se encuentra dada en función de la entrada únicamente y de sus respectivos retrasos.
Para los filtros recursivos la ecuación en diferencias se encuentra expresada en función de dos formas polinomiales así:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \quad (5)$$

La cual nos lleva a encontrar una función de transferencia de la forma:

$$H(z) = \frac{\sum_{k=0}^{M} a_k z^{-k}}{1 - \sum_{k=1}^{N} b_k z^{-k}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{1 - b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}} \quad (6)$$

A los primeros pertenecen los filtros tipo FIR, los cuales se caracterizan por no poseer realimentación y a los segundos los filtros tipo IIR en los cuales la salida se encuentra dada en función de la entrada y de las salidas de instantes anteriores.[3]

B. FILTRO FIR

1)CONCEPTO

Este filtro es del tipo no-recursivo y presenta una respuesta finita al impulso, el cual se encuentra limitado por el número de términos, esto es lo opuesto al filtro de impulso infinito (IIR) el cual produce un número infinito de términos de salida cuando se aplica un impulso unitario en su entrada.
La salida de este filtro depende únicamente del presente y pasado de las entradas. Esta característica es de vital importancia para el diseño e implementación de esta clase de filtros.[3],[6]

2. CARACTERÍSTICAS DE LOS FILTROS FIR.

Algunas de las características más importantes que presentan los filtros tipo FIR son:

Respuesta finita al impulso: La respuesta finita al impulso significa que el efecto de transiente o condiciones iniciales en la salida del filtro eventualmente desaparecen.
Este filtro es simplemente un arreglo entre los coeficientes del filtro y los retardos de la entrada del mismo.
La ecuación en diferencias que representa este tipo de filtro es:

$$y(nT) = \sum_{k=0}^{N} b_k x(nT - kT) \quad (7)$$

La función de transferencia se encuentra dada por:

$$H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n} \quad (8)$$

Y el diagrama de bloques se encuentra en la figura 2.

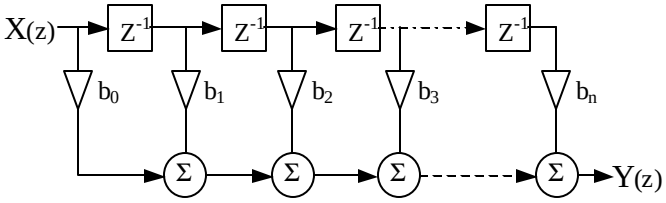

Figura. 2. Diagrama de bloques de un Filtro FIR.

Fase línea: En muchas de las aplicaciones del procesamiento digital de señales, es de suma importancia preservar algunas de las características de la señal de entrada, a través de la operación de filtrado. Un filtro con fase lineal como este, presenta tan solo un simple tiempo de retardo en la respuesta de fase y así la distorsión de fase se hace mínima.

Estabilidad: Puesto que es un filtro no-recursivo, este no presenta realimentación, es por esta razón que no posee polos excepto el que se coloca en Z=0, por lo tanto no hay posibilidad que exista un polo por fuera del circulo unitario. Lo cual significa que este es inherentemente estable.

Longitud de palabra finita: Cuando los valores de los coeficientes para un filtro son calculados, la implementación digital puede solo ser aproximada a algunos valores deseados.
Las limitaciones introducidas por el almacenamiento digital de términos son determinadas por la dimensión finita del registro.

Facilidad de diseño: Todas las propiedades antes mencionadas contribuyen a un proceso de diseño de filtros FIR realmente fácil. Existen muchas formas directas de diseño de filtros tipo FIR, si se conoce la respuesta en frecuencia y fase deseada, para esto hay métodos que utilizan diferentes tipos de aritmética, tales como la distribuida o la on-line, pero además de estos también existen implementaciones clásicas como la que se muestra en el presente articulo.[1]

IV. CARACTERISTICAS DEL SISTEMA IMPLEMENTADO

El sistema que se desarrollo permite la implementación de filtros digitales tipo FIR sobre un FPGA en este caso un SPARTAN 10 XL[2], el cual a su vez hace parte de un proyecto mayor denominado EVA-01.

Las características de este filtro se encuentran especificadas en la tabla 1.

Tipo De Filtro	**Cantidad De Coeficientes**	**Longitud De Palabra**	**Anchos De Banda De Trabajo**	
FIR	3 – 64	8 bits	77.8 Khz,	39 Khz
			19.5 Khz	9.7 Khz
			4.8 Khz	1.2 Khz
			600 Hz	150 Hz.

Tabla 1. Características del filtro.

Para lograr el correcto funcionamiento de este proyecto se desarrollaron dos partes principales para cumplir con este cometido así:

- Interfase gráfica (Software) de diseño y configuración del Hardware.
- Tarjeta de desarrollo EVA-01 (Hardware) en la que es implementado físicamente el filtro.

A. INTERFASE GRAFICA

Se creo una herramienta de software la cual 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 [9] y en su inicio presenta 3 posibilidades de filtros tipo FIR para trabajar así:

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

Para efectos de este articulo, solo se trabaja con la opción de filtro con coeficientes dinámicos.

Esta sección presenta dos bloques principales así:

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

- Cantidad de coeficientes.
- Ancho de banda.
- Atenuación y ganancia del filtro para diferentes frecuencia, tal y como se muestra en la figura 3.

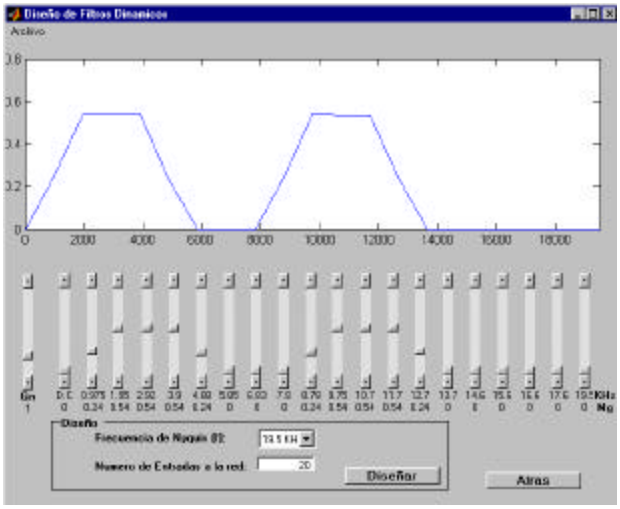

Figura 3. Interfase para el diseño de los filtros.

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 el cual se lleva a cabo mediante la utilización del algoritmo de Remez.

Verificación e implementación: Una vez que se han determinado las especificaciones del filtro y se ha diseñado. Se observa una nueva ventana (figura 4) la cual muestra la respuesta en frecuencia del filtro. En esta se presentan tres respuestas espectrales diferentes así:

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

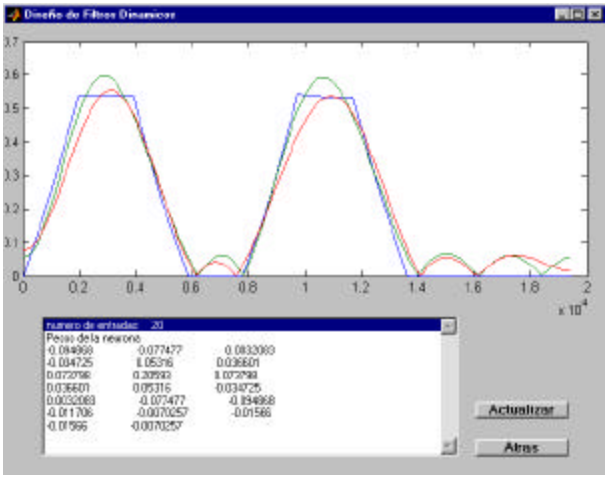

Figura 4. Ventana de verificación e implementación del filtro.

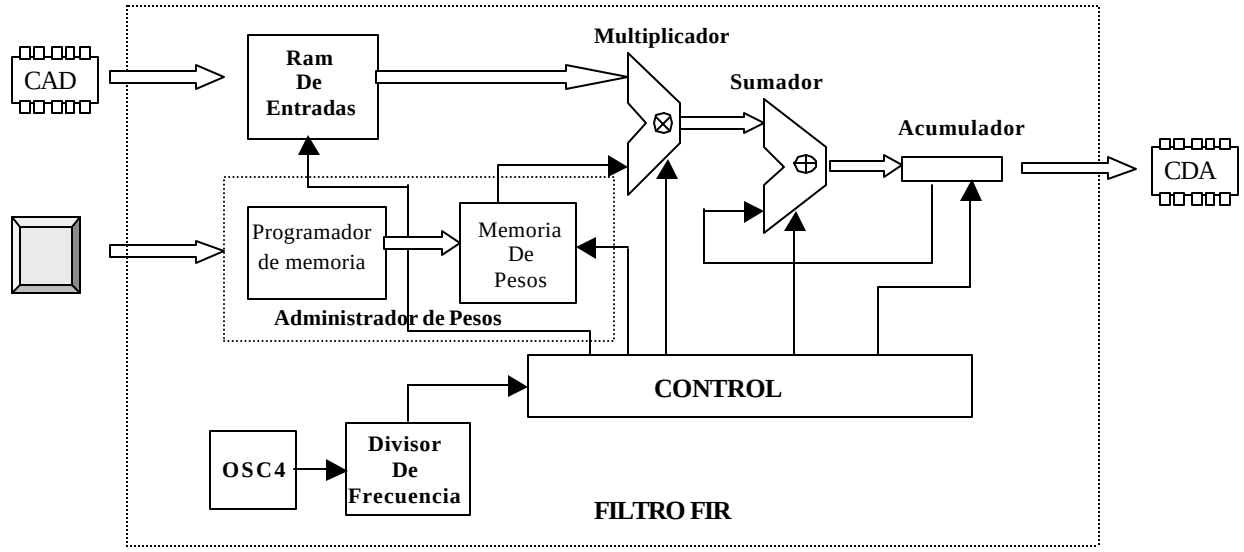

Figura 5. Diagrama de bloque del filtro FIR.

Una vez diseñado y verificada la respuesta en frecuencia del filtro, estas características pueden ser transmitidas al hardware desde el PC, sin la necesidad de reiniciar el sistema, es posible reconfigurar los coeficientes al mismo tiempo en que el filtro se encuentra en pleno funcionamiento (On-Line) gracias a que dentro del FPGA se encuentra implementada una memoria de acceso dual.

B. HARDWARE

Para implementar el hardware, tan solo se hace necesaria la utilización de un FPGA y los conversores Analogo-Digital y Digital-Analogo respectivamente, para adquirir las señales análogas y entregarlas de la misma forma al medio.

De lo anterior se puede deducir, que lo fundamental en el montaje de un filtro digital es la descripción en hardware (VHDL)[8] que se realice, para así poder implementarse sobre un dispositivo lógico programable como lo es el FPGA SPARTAN 10XL, el cual fue el utilizado para nuestro propósito.

C. ETAPAS DEL DISEÑO

El diseño se realizo en forma jerárquica, de tal manera que se dividió en diferentes etapas, tal como se muestra en la figura 5.

Antes de iniciar el estudio de cada una de estas entidades es conveniente tener en cuenta algunos de los aspectos de funcionamiento que presenta dicho diseño así:

- El filtro presenta un máximo de 64 coeficientes, lo cual implica la utilización de una memoria de pesos y otra de entradas con estas especificaciones
- El ancho de palabra es de 8 bits lo cual obliga que las memorias, el sumador y el acumulador trabajen con esta especificación.
- Debido 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 y −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 el cual es de 8 Mhz y al que se le acondiciona un contador para poder dividir la frecuencia y asi obtener diferentes posibilidades en el ancho de banda del filtro asi: 77.8 Khz, 39 Khz, 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 así, mediante una interfase de puerto paralelo al PC, se adquieren los nuevos coeficientes para lograr la reconfiguración del filtro sin la necesidad de reiniciar el sistema.

Ram De Entradas

Esta entidad es la encargada de recibir los datos que provienen del CAD y colocarlos en la posición de memoria según lo indique la unidad de control.

Por otro lado esta entidad se ocupa de transformar el dato de entrada a formato de signo magnitud fraccional, normalizado entre 1 y −1, esto con el fín de facilitar el trabajo del multiplicador.[7]

A continuación se muestra la descripción en VHDL de la memoria de entradas:

```
Library IEEE ;
use IEEE.std_logic_1164.all ;
use IEEE.std_logic_arith.all ;
use IEEE.std_logic_unsigned.all ;
entity ram_ent is
 port (
       dir_ent  : in  std_logic_vector(5 downto 0);
       carg     : in  std_logic ;
       entrada  : in  std_logic_vector(7 downto 0);
      dato_ent : out std_logic_vector(7 downto 0));
end ram_ent ;
architecture synt of ram_ent is
 type ram is array (63 downto 0) of std_logic_vector(7
downto 0) ;
 signal bank : ram ;
 signal entrada_1     : std_logic_vector(7 downto 0);

  begin
  process(entrada)
  begin
  if entrada(7) = '0' then
     entrada_1(7 downto 0) <=not( entrada(7 downto
0));
  else entrada_1(7) <= not(entrada(7));
  entrada_1(6 downto 0) <=( entrada(6 downto 0));
  end if;
  end process;

  process (dir_ent,carg,entrada_1,bank)
  begin
   if rising_edge(carg) then
     bank(dir_ent)<=entrada_1;
   end if ;
  dato_ent <= bank(conv_integer(dir_ent)) ;
  end process ;
end synt ;
```

Administrado De Pesos

Esta entidad tiene como funcion principal, recibir los valores de los pesos, provenientes del computador, administrar la comunicación con el PC, almacenar los valores de los pesos en memoria y posteriormente entregarlos al multiplicador, para que sean procesados junto con el dato de la entrada correspondiente.

Para lograr este cometido la entidad se encuentra dividida en dos componentes mas, los cuales son:

- ***Programador de la memoria***: Este es el encargado de la comunicación con el PC y de recibir los datos para luego hacer todo el manejo pertinente en la programación de la memoria.

- ***Memoria de pesos***: Esta memoria es de acceso dual, lo cual permite que sea grabada mientras se encuentra entregando los datos al mutiplicador, sin la necesidad que un proceso sea interrumpido por otro. La memoria es de 64X8 bits, lo que permite trabajar con un número máximo de 64 coeficientes. Los datos que son entregados por el PC también se encuentran en formato de signo magnitud fraccional y normalizados entre 1 a –1, lo cual facilita el trabajo a la etapa de multiplicación. A continuación se muestra la descripción en VHDL del Administrador de pesos.

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_signed.all;

entity arc_ada is
port (reset_pc, dato_pc,clk_pc  : in std_logic;
   dir_pes      : in std_logic_vector(5 downto 0);
   dir_memo1    : out std_logic_vector(5 downto 0);
   dato_pes,dato_memo1    :  out  std_logic_vector(7
downto 0));
end arc_ada;

architecture arquitec of arc_ada is

component pro_memo
port (clk_pc,reset_pc, dato_pc : in  std_logic;
   write                  : out std_logic;
   dir_memo           : out std_logic_vector (5 downto
0);
   dato_memo  : out std_logic_vector (7 downto 0));
end component;

component ram_ada
port  (dir_pes,dir_memo    :  in    std_logic_vector(5
downto 0);
   write    : in  std_logic ;
   dato_memo : in  std_logic_vector(7 downto 0);
   dato_pes  : out std_logic_vector(7 downto 0));
end component;

signal write: std_logic;
signal dir_memo: std_logic_vector(5 downto 0);
signal dato_memo: std_logic_vector(7 downto 0);

begin

dato_memo1<=dato_memo;
dir_memo1<=dir_memo;

graba:
pro_memo       port       map       (clk_pc,reset_pc,
dato_pc,write,dir_memo,dato_memo);

memoria:
ram_ada                                          port
map(dir_pes,dir_memo,write,dato_memo,dato_pes);

end arquitec;
```

Multiplicador

Esta entidad es la encargada de recibir los datos de la memoria de entradas y la de pesos y llevar a cabo la multiplicación entre los dos valores y entregar el resultado al sumador en formato de complemento base dos fraccional. Este multiplicador se encuentra contruido en forma combinacional en su totalidad, tal y como lo muestra a continuación la descripción en VHDL del multiplicador

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity mult is
   port (dato_pes,dato_ent  : in  std_logic_vector(7
downto 0);
        mul  : out  std_logic_vector(7 downto 0));
end mult ;

architecture behavior of mult is
signal data_out_1 : std_logic_vector(13 downto 0);
signal sig_d     : std_logic;
begin

process (dato_ent, dato_pes,data_out_1,sig_d)
begin
sig_d <= dato_pes(7) xor dato_ent(7);
data_out_1 <= std_logic_vector(unsigned(dato_pes(6
downto 0)) * unsigned(dato_ent(6 downto 0)));

if sig_d ='1' then
if data_out_1(13 downto 7) = 0 then
  mul(7 downto 0)<=("00000000");
else MUL(6 DOWNTO 0) <= (not data_out_1(13
downto 7))+1;
  mul(7)<=sig_d;
end if;
else MUL(6 DOWNTO 0) <= data_out_1(13 downto
7);
  mul(7)<=sig_d;
end if;
end process;

end behavior;
```

Sumador Acumulador

Es la entidad encargada de recibir los datos entregados por el multiplicador y por orden de la entidad de control, realiza la suma de cada uno de los productos entre los coeficientes y las entradas. Dicha suma es guardada en en un acumulador en cada paso, para finalmente ser entregada al CDA en formaro de entero.

Descripción en VHDL de la etapa del Sumador Acumulador:

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY sum IS
PORT(
   res_sum,clk_sum,sal: in STD_LOGIC;
   mul: in STD_LOGIC_vector(7 downto 0);
   salida: out STD_LOGIC_vector(7 downto 0);
   satura: out STD_LOGIC);
END sum;

ARCHITECTURE estruct OF sum IS
signal sumt: std_logic_vector(7 downto 0);
signal sat_1,sat_2,sat_3: std_logic;
begin

PROCESS (mul,clk_sum,res_sum,sumt,sal)
begin
if  res_sum='1' then
  sumt <= (others => '0');
  sat_1<='0';
  sat_2<='0';
  sat_3<='0';
  satura<='0';
elsif rising_edge(clk_sum) then
  sumt<= sumt + mul;
  sat_1<=sumt(7);
  sat_2<=mul(7);
if (sat_1 ='0' and sat_2 ='0' and sat_3 ='1') or (sat_1
='1' and sat_2 ='1' and sat_3 ='0') then
  satura <='1';
end if;
elsif falling_edge(clk_sum) then
  sat_3 <=sumt(7);
end if;

if  sal='1' then
  salida(6 downto 0) <= sumt(6 downto 0);
  salida(7)<=not(sumt(7));
else null;
end if;
end process;
end estruct;
```

Control

Esta presenta como objetivo unico, la sincronización de las demas entidades. Para esto se construyo una maquina de estados, la cual se encarga de controlar el proceso de los datos desde la adquisición de las entradas, el almacenamiento en memoria, el proceso de multiplicación, suma y acumulación para finalmente entregar el resultado al CAD.

El unico proceso en el que no toma parte es el de comunicaión al PC y almacenamiento en memoria de los pesos, el cual se lleva a cabo sin comunicación alguna con esta entidad. Esta caracteristica es la que permite que los coeficientes sean alterados sin

interrumpir el ritmo de procesamiento de la imformación.
Descripción en VHDL de la etapa de control:

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity temp_global is
port (clk,reset: in std_logic;
      dir_pes, dir_ent: out std_logic_vector (5 downto
0);
    sal,adc,clk_sum,carg, res_sum: out std_logic);
end temp_global;

architecture flow of temp_global is

type estado is (st0, st1, st2, st3, st4);
signal estado_actual,siguiente_estado: estado;
signal cont_pes, cont_ent : std_logic_vector (5 downto
0);
signal res_1:          std_logic;
type rom_type is array(0 to 15) of std_logic;
signal rom_adc: rom_type:= ('0', '1', '1', '1', '1', '1', '1',
'1', '1','1','1','1','0','0','0','0' );
begin
process (clk, res_1, reset, estado_actual, cont_pes,
cont_ent)
begin
if reset ='0' then
  estado_actual <= st0;
elsif rising_edge(clk) then
  estado_actual <= siguiente_estado;
end if;

if estado_actual = st2 or estado_actual=st3 then
  clk_sum<= clk;
else clk_sum<='1';
end if;

if estado_actual = st4 and clk ='0'  then
  sal <='1';
  carg <='1';
else sal <='0';
  carg<='0';
end if;

if (estado_actual = st2 and clk ='1' and res_1 ='1')or
estado_actual = st1 then
  res_sum <='1';
else res_sum <='0';
end if;
if falling_edge(clk) then
if estado_actual=st4  then
  res_1<='1';
else res_1<='0';
end if;
if estado_actual=st2 then
  cont_ent <= cont_ent;
  cont_pes <= cont_pes;
elsif estado_actual =st3 then
  cont_ent <= cont_ent + "000001";
  cont_pes <= cont_pes + "000001";
elsif  estado_actual = st4 then
  cont_pes <= cont_pes + "000001";
  cont_ent <= cont_ent;
else cont_pes <= "000000";
  cont_ent <= "000000";
end if;
end if;

dir_ent <= cont_ent;
dir_pes <= cont_pes;
end process;
adc<=rom_adc(conv_integer(cont_pes(3 downto 0)));

process(estado_actual,cont_pes)
begin
case estado_actual is
when st0 =>
  siguiente_estado <= st1;
when st1 =>
  siguiente_estado <= st2;
when st2 =>
  siguiente_estado <= st3;
when st3 =>
if cont_pes = "111111" then
  siguiente_estado <= st4;
else siguiente_estado <= st3;
end if;
when st4 =>
  siguiente_estado <= st2;
end case;
end process;
end flow;
```

FILTRO FIR

Esta es la arquitectura general y es la encargada de unir todo el resto de entidades, tomandolas como componentes y realizando la interconección entre cada una de ellas . Ademas de esto se ocupa de unir el OSC4 (oscilado interno del FPGA) con el resto del diseño, mediante un contador que hace las veces de divisor de frecuencia, para asi variar la frecuancia del reloj que alimenta la etapa de control y de esta forma poder ofrece diferentes velocidades de prcesamiento y a su vez varaiar el ancho de banda en el manejo del filtro.

Descripción en VHDL del FILTRO:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

```
entity adaline is

port (
    reset,dato_pc,clk_pc,reset_pc : in std_logic;
    entrada   : in std_logic_vector(7 downto 0);
    sel_clk   : in std_logic_vector(2 downto 0);
    dato_memoria  : out std_logic_vector(7 downto 0);
    dir_memo      : out std_logic_vector(10 downto 0);
    mux_in        : out std_logic_vector(4 downto 0);
    mux_sal       : out std_logic_vector(3 downto 0);
    satura,adc,oe,we,pin_s,re_su  : out std_logic;
    salida        : out std_logic_vector(7 downto 0));
end adaline;

architecture arquitec of adaline is

component temp_global
port (
   clk,reset: in std_logic;
   dir_pes, dir_ent: out std_logic_vector (5 downto 0);
   sal,adc,clk_sum,carg, res_sum : out std_logic );
END component;

component ram_ent
  port (
     dir_ent  : in  std_logic_vector(5 downto 0);
     carg     : in  std_logic ;
     entrada  : in  std_logic_vector(7 downto 0);
     dato_ent : out std_logic_vector(7 downto 0));
end component;

component arc_ada is
  port (reset_pc, dato_pc,clk_pc : in std_logic;
     dir_pes     : in std_logic_vector(5 downto 0);
     dir_memo1   : out std_logic_vector(5 downto 0);
     dato_pes,dato_memo1 : out std_logic_vector(7
downto 0));
end component;

component mult
  port (
     dato_pes,dato_ent : in  std_logic_vector(7 downto
0);
      mul    : out  std_logic_vector(7 downto 0));
end component;

component sum
  port(
     res_sum,clk_sum,sal : in STD_LOGIC;
     mul  : in STD_LOGIC_vector(7 downto 0);
     salida: out STD_LOGIC_vector(7 downto 0);
     satura: out STD_LOGIC);
end component;

Component OSC4
  port(
     F8M   : out  std_logic:= 'Z';
     F500K : out  std_logic:= 'Z';
     F16K  : out  std_logic:= 'Z';
     F490  : out  std_logic:= 'Z';
     F15   : out  std_logic:= 'Z');
end component;

signal sal,clk_sum,carg,res_sum,clk,clk_con:
std_logic;
signal dir_pes,dir_ent,dir_memo1: std_logic_vector(5
downto 0);
signal dato_pes,dato_ent,mul,dato_memo1:
std_logic_vector(7 downto 0);
signal con_clk: std_logic_vector(8 downto 0);

begin

dato_memoria<=dato_memo1;
dir_memo<="00000" & dir_memo1;
mux_in <="10000";
mux_sal<="0000";
we<='0';
oe<='1';
pin_s<=clk_sum or (not(clk));
re_su<=res_sum;

process(clk_con, con_clk)
begin
if rising_edge(clk_con) then
  con_clk<=con_clk + 1;
else con_clk<=con_clk;
end if;
end process;

process(clk_con, con_clk, sel_clk)
begin
case sel_clk is
  when "000" => clk <= clk_con;
  when "001" => clk <= con_clk(0);
  when "010" => clk <= con_clk(1);
  when "011" => clk <= con_clk(2);
  when "100" => clk <= con_clk(3);
  when "101" => clk <= con_clk(5);
  when "110" => clk <= con_clk(6);
  when others => clk <= con_clk(8);
end case;
end process;

relojes:
OSC4 port map (F8M =>clk_con, F500K =>open ,
F16K =>open, F490=>open, F15=>open);

control:
temp_global port map(clk,reset,dir_pes,
dir_ent,sal,adc,clk_sum,carg, res_sum );



pesos:
```

arc_ada port map (reset_pc, dato_pc,clk_pc,dir_pes,dir_memo1,dato_pes,dato_memo1);

entradas:
ram_ent port map(dir_ent,carg,entrada,dato_ent);

multiplicador:
mult port map (dato_pes,dato_ent,mul);

sumadores:
sumport map (res_sum, clk_sum, sal, mul, salida, satura);

end arquitec;

V CONCLUSIONES

Luego de realizar un estudio sobre las principales características de los filtros digitales, en especial del tipo FIR y de realizar su descripción en hardware para un FPGA, es claro observar algunas caracteristicas que se pueden tomar de este trabajo asi:

- Los filtros digitales presentan un buen número de vetajas sobre los filtros analogos, tales como: Alta inmunidad al ruido, Precisión, Facil reconfiguración de sus caracteristicas y un desempeño independiente la precisión o deterioro de sus componentes.

- Las características del filtro son excepcionales, sobre todo cuando se desea un respuesta en frecuencias con pendintes muy elevadas o un filtrdo bastante selectivo.

- La utilización del FPGA para la implementación de filtros digitales, genera algunas ventajas tales como:

    1. Se logra obtener una velocidad de muestreo mayor que los implementados en software.
    2. Al implementar el filtro con este dispositivo, solo se hace necesaria la utilización de un conversor análogo-digital y uno digital-análogo.
    3. La capacidad de reconfiguracion y la posibilidad de utilizar memorias de acceso dual, ayudan a cambiar las características del sistema sin tener que reiniciarlo.

- Por último, es bueno observar que la utilización de un adecuado formato en la caracterización de las entradas y de los coeficientes. Ayuda, para que el sistema trabaje eficientemente.

REFERENCIAS

[1] Willis J. Tompkins, "*Biomedical Digital Signal Procesing*". Ed. Prentice Hall may, 1993.

[2] "*XILINX THE PROGAMABLE LOGIC DATABOOK*". DS060, Marzo 2 de 2000.

[3] J G, Proakis, Dimitris G. Manolakis "*Tratamiento Digital De Señales*". Ed. Prentice Hall International., 1997.

[4] Samuel . Stearms, Ruth A. David, "*Signal Procesing Algoritms*". Ed. Prentice Hall International., 1997.

[5] " *Binary Numbering Systems*". ALTERA CORPORATION 1997

[6] R. W. Hamming, " *DIGITAL FILTERS*", Ed. Prentice Hall International 1989.

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

[8] "*A CPLD VHDL introdution*". Xilinx application note. January 12, 1998.

[9] "*digital signal processing toolbox*". The MathWorks Inc. 1992-2001

L. J. Arcos Pantoja Ingeniero Electrónico de la Universidad Santo Tomas de Aquino, Bogotá D.C. Cursos realizados: Curso de Inteligencia computacional: teoría y aplicaciones, Universidad Nacional de Colombia, Bogotá D.C., Octubre de 2002.

J. M. Calderón Chávez Ingeniero Electrónico de la Universidad Santo Tomas de Aquino, Bogotá D.C. Cursos realizados: Diseño Digital y Lenguajes de Descripción de Hardware, IEEE, Universidad Santo Tomas de Aquino, Bogotá D.C, *Octubre 2000-Enero 2001*. Curso de Inteligencia computacional: teoría y aplicaciones, Universidad Nacional de Colombia, Octubre de 2002, Bogotá D.C.