DESARROLLO EN VHDL DE UN FILTRO DIGITAL GENÉRICO BASADO EN ESTRUCTURAS CANÓNICAS

A. ALVAREZ, A. MILLAN, M. J. BELLIDO, J. JUAN,
P. RUIZ-DE-CLAVIJO, D. GUERRERO, E. OSTUA Y J. VIEJO
Departamento de Tecnología Electrónica. Universidad de Sevilla
Av. Reina Mercedes, s/n (E. T. S. Ingeniería Informática) - 41012 Sevilla
avealvarez@yahoo.com, {amillan, bellido, jjchico, paulino, guerre, ostua, julian}@dte.us.es
Tel.: +34 954556161 - Fax: +34 954552764
http://www.dte.us.es/gtm/

Este trabajo abarca la realización de un filtro digital a bajo nivel. El diseño propuesto se basa en la utilización de un lenguaje de descripción de hardware (VHDL) así como de una estructura canónica para la implementación del filtro. Con ello, se aporta al alumno un enfoque mucho más práctico del diseño de filtros, desde un punto de vista adecuado a la titulación de Ingeniería en Informática.[1]

1. Introducción

El trabajo que presentamos se incluye dentro de la asignatura «Tratamiento Digital de Señales» de tercer curso de Ingeniería en Informática (Universidad de Sevilla). Hasta ahora, las prácticas de dicha asignatura han cubierto básicamente el diseño de filtros digitales a alto nivel, utilizando la herramienta OCTAVE [1]. Así, el objetivo principal de este trabajo ha sido añadir al programa de prácticas la realización de filtros a bajo nivel. Para ello, se ha pretendido que los alumnos implementen un filtro desde el punto de vista *hardware* siguiendo dos pautas claras. Por un lado, el filtro debía ser totalmente configurable (debía ser posible modificar la precisión de los datos, el orden y los coeficientes del mismo) por lo que se ha elegido un lenguaje de descripción de *hardware* para su desarrollo (en nuestro caso VHDL [2, 3, 4]). Por otro lado, el filtro debía seguir alguna de las estructuras canónicas expuestas en la literatura debido a su mayor eficiencia (en comparación con la forma directa).

2. Interfaz y estructura interna del filtro

Tanto la interfaz como la estructura interna del filtro diseñado se muestran en la Fig. 1 (N es la precisión de los datos de entrada en bits). Las señales CLK y RESET dotan al circuito de reloj y puesta a cero. Los buses X e Y son la entrada y la salida de datos del filtro. Por último, la señal STROBE indica al circuito que hay un nuevo dato válido en la entrada. La estructura interna del filtro es relativamente sencilla y cuenta con los siguientes componentes:

- RAM, almacena los vectores auxiliares necesarios para el cálculo de la salida.
- ROM, almacena los coeficientes del filtro.
- Acumulador (AC), necesario para almacenar el resultado parcial durante cada iteración del filtro.
- ALU, realiza operaciones aritméticas en un solo ciclo de reloj (suma, resta y multiplicación).

Todos los componentes trabajan con datos de $2N$ bits con objeto de mantener la precisión de los resultados.

[1] Este trabajo ha sido financiado parcialmente por el proyecto OFU TIC 1023 de la CONSEJERÍA DE INNOVACIÓN, CIENCIA Y EMPRESA de la JUNTA DE ANDALUCÍA.

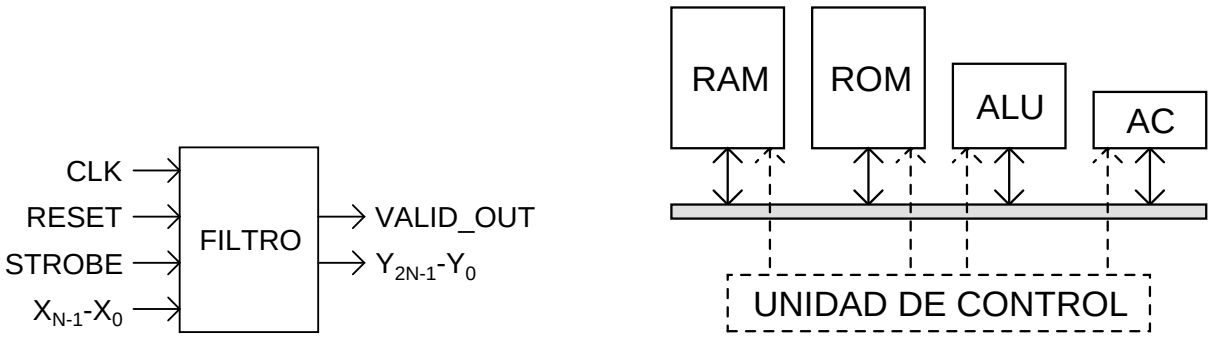


Figura 1: Interfaz y estructura interna del filtro.

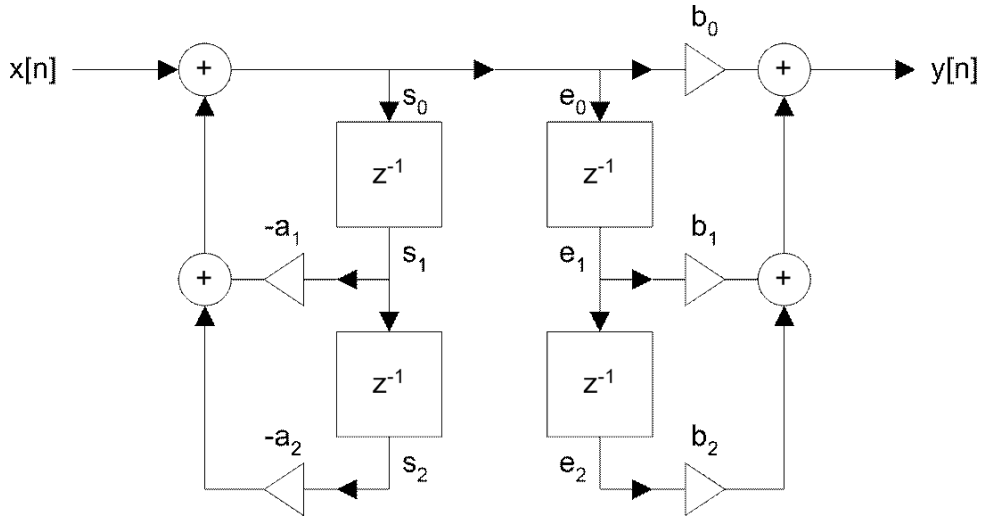


Figura 2: Estructura canónica para filtros recursivos de orden dos.

3. Funcionamiento del filtro

El funcionamiento del filtro sigue la forma canónica cuya estructura para orden dos se muestra en la Fig. 2. Esta estructura permite calcular la salida utilizando $3L$ posiciones de memoria (siendo L la longitud del filtro) frente a las $4L$ posiciones que requiere la forma directa derivada de la ecuación de recurrencia de un filtro genérico [5, 6]:

$$y[n] = \sum_{k=0}^{M} b[k]x[n-k] - \sum_{k=1}^{M} a[k]y[n-k]$$

donde $x[n]$ e $y[n]$ son la entrada y la salida del sistema, $a[k]$ y $b[k]$ son los coeficientes del filtro, y M es el orden del mismo. El código OCTAVE correspondiente a dicha estructura se muestra en la Fig. 3.

4. Diseño en VHDL

El diseño del filtro se ha realizado en lenguaje VHDL mediante la herramienta XILINX ISE WEBPACK v7.1iWP3.0 [7]. En primer lugar se ha diseñado la arquitectura del módulo (Fig. 1) y después se ha desarrollado el código VHDL del mismo a nivel RTL (*register transfer level*). Cabe indicar que el

```
function y = iir1(b, a, x)
  b = b./a(1);
  a = a./a(1);% NORMALIZAR COEFICIENTES
  L = max([length(a), length(b)]);
  M = L-1;
  while length(a) < L
    a = [a, 0];
  end
  while length(b) < L
    b = [b, 0];
  end% IGUALAR LONGITUDES DE VECTORES DE COEFICIENTES
  v = zeros(L, 1);% INICIALIZAR VECTOR DE CALCULOS TEMPORALES
  N = length(x);
  for n = 1:N
    ac = x(n);
    for k = 2:L
      ac = ac-a(k)*v(k);
    end% CALCULO DE LA PARTE RECURSIVA
    v(1) = ac;
    ac = b(1)*v(1);
    for k = L:-1:2
      ac = ac+b(k)*v(k);
      v(k) = v(k-1);
    end% CALCULO DE LA PARTE NO RECURSIVA
    y(n) = ac;
  end
```

Figura 3: Código OCTAVE correspondiente a la estructura canónica empleada en el filtro.

código correspondiente considera como parámetros tanto la precisión de los datos como el orden y los coeficientes del filtro, con objeto de conseguir un sistema totalmente configurable. La verificación del diseño se ha realizado mediante la herramienta MODELSIM XILINX EDITION v6.0a [8] (Fig. 4). Por último, el circuito diseñado ha sido empleado en la implementación de diversos filtros. Se muestran los resultados obtenidos para un filtro de paso de banda a modo de ejemplo (Fig. 5).

5. Conclusiones

Con este trabajo, se ha conseguido completar el programa de prácticas de la asignatura indicada con anterioridad, aportando al alumno un punto de vista mucho más práctico del diseño de filtros digitales, aunque manteniéndolo dentro del ámbito de su titulación.

Referencias

[1] "Octave." http://www.octave.org/.

[2] F. Pardo and J. A. Boluda, *VHDL. Lenguaje para síntesis y modelado de circuitos*. RA-MA, 1999.

[3] P. J. Ashenden, *The designer's guide to VHDL.* Morgan Kaufmann, 2nd ed., 2002.

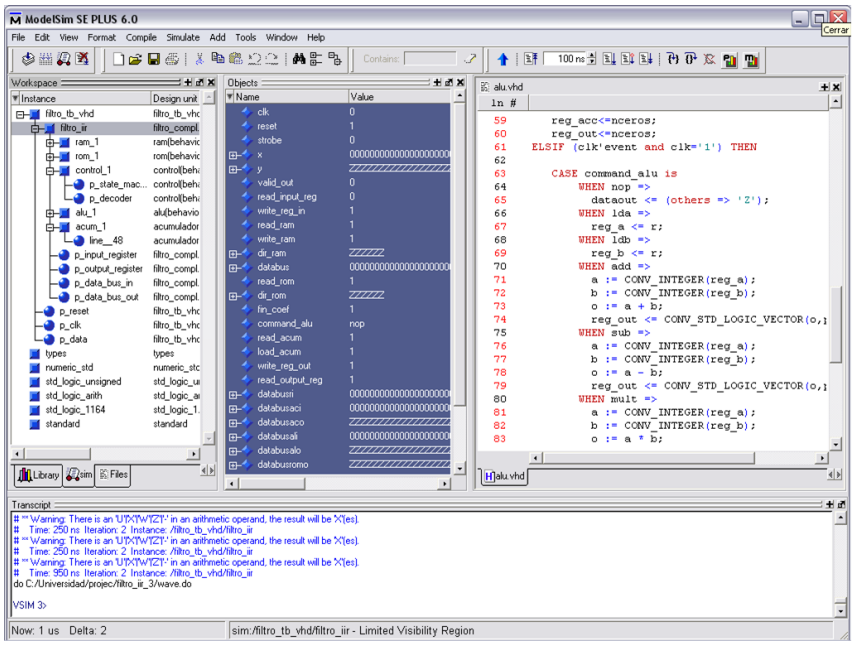
[4] S. A. Pérez, E. Soto, and S. Fernández, *Diseño de sistemas digitales con VHDL.* Thomson, 2002.

[5] J. B. Mariño, F. Vallverdú, J. A. Rodríguez, and A. Moreno, *Tratamiento digital de la señal. Una introducción experimental*. Edicions UPC, 2nd ed., 1996.

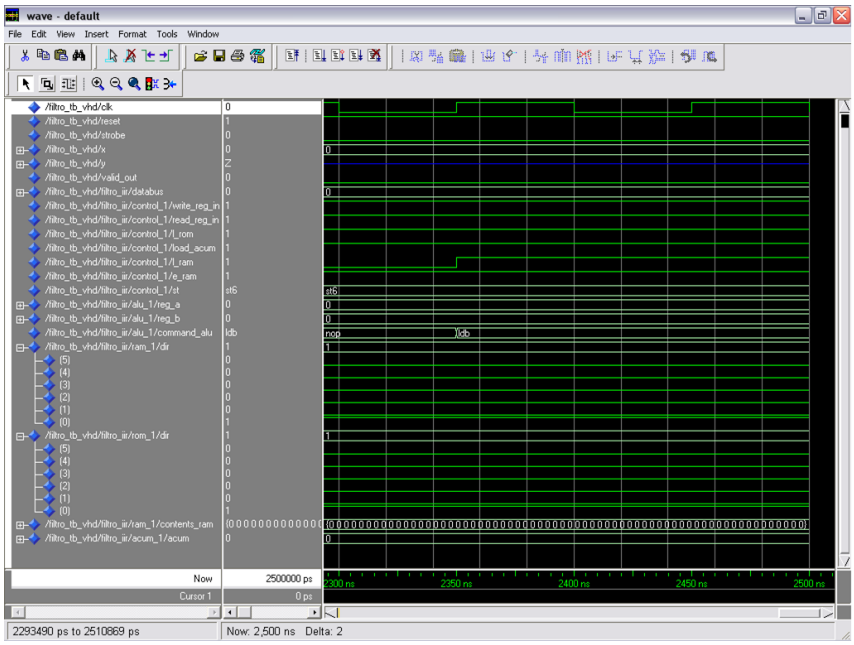
[6] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Tratamiento de señales en tiempo discreto*. Prentice Hall, 2nd ed., 2000.

[7] "ISE WebPACK." http://www.xilinx.com/ise/logic_design_prod/webpack.htm.

[8] "ModelSim Xilinx Edition-III." http://www.xilinx.com/ise/optional_prod/mxe.htm.

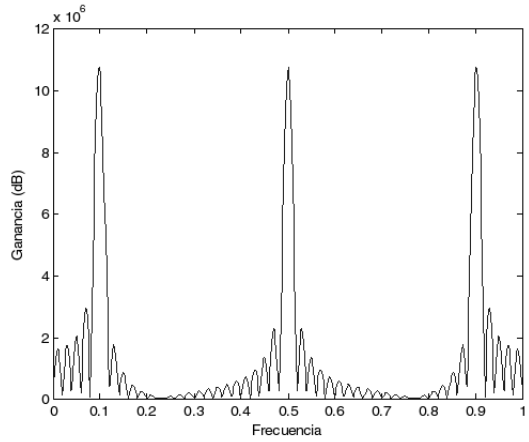


(a)

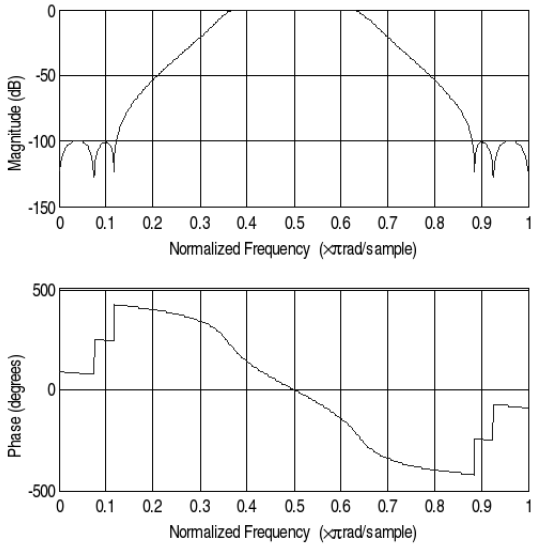


(b)

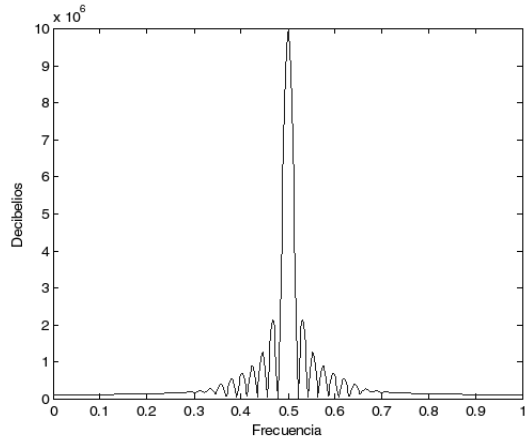
Figura 4: Entorno de diseño de MODELSIM (a) y detalle de simulación del sistema (b).



(a)



(b)



(c)

Figura 5: Implementación de un filtro de paso de banda utilizando el circuito diseñado: (a) espectro de entrada, (b) respuesta en frecuencia del filtro y (c) espectro de salida.