Programación Matemática y Software (2021) 13 (1): 21-30. ISSN: 2007-3283

Implementación en FPGA de filtros digitales IIR selectivos en frecuencia con fines didácticos

FPGA implementation of frequency-selective digital IIR filters for educational purposes

Mariana N. Ibarra Bonilla

División de Ingeniería Mecatrónica, Instituto Tecnológico Superior de Atlixco. Prolongación Heliotropo No.1201, Colonia Vista Hermosa, Atlixco, Puebla, México. C.P. 74210.
Correo-e: mariana.ibarra@itsatlixco.edu.mx

PALABRAS CLAVE:

filtro digital, IIR, FPGA, máquina de estados finita.

RESUMEN

El presente artículo presenta la metodología para implementar un filtro digital de tipo IIR de segundo orden en un FPGA. El chip es un XC6SLX16-Spartan-6 integrado en una tarjeta de desarrollo Avanxe, la cual incluye un PSoC con un convertidor analógico-digital y digital-analógico. El filtro digital es programado en lenguaje VHDL usando una Máquina de Estados Finita (FSM). El desempeño del filtro se comprobó en una aplicación didáctica para filtrar señales por arriba de los 100 Hz, en donde en comparación con una simulación con el software Matlab, la propuesta presentó una exactitud del 99.8%.

KEYWORDS:

digital filter, IIR, FPGA, finite state machine.

ABSTRACT

This article presents the methodology to implement a second-order digital IIR filter in an FPGA. The chip is an XC6SLX16-Spartan-6 embedded in an Avanxe development board, which includes a PSoC with an analog-digital and digital-analog converter. The digital filter is programmed in VHDL language using a Finite State Machine (FSM). The filter performance was verified in a didactic application to filter signals above 100 Hz, which compared to a simulation with Matlab software, the proposal presented an accuracy of 99.8%.

Recibido: 30 de agosto de 2020 • Aceptado: 15 de noviembre de 2020 • Publicado en línea: 26 de febrero de 2021

1 INTRODUCCIÓN

Actualmente las nuevas tecnologías y aplicaciones en diferentes campos de la industria están aprovechando los algoritmos de procesamiento digital de señales (DSP, del inglés digital signal processing) para tratar con imágenes, voz, señales biomédicas, robótica, instrumentación, entre otros [1]. Una de las principales razones de su éxito en la industria es el desarrollo y uso de software y hardware de bajo costo. Esto conducirá a una mayor demanda de ingenieros con experiencia en DSP y por lo tanto, es necesario hacer de DSP una parte integral de cualquier plan de estudios de ingeniería eléctrica, electrónica y afín.

Filtro es el nombre genérico que se refiere a un sistema lineal invariante en el tiempo (LTI, del inglés *linear time invariant*) diseñado para un trabajo específico de selección de frecuencia o discriminación de frecuencia [1]. Por lo tanto, un sistema LTI en tiempo discreto es denominado filtro digital. En las aplicaciones del DSP, los filtros digitales desempeñan un papel importante y han sido implementados en una amplia gama de aplicaciones en ingeniería [2, 3, 4, 5].

Los FPGAs (*Field Programmable Gate Array*) son una alternativa para la implementación en hardware de los filtros digitales. La capacidad de procesamiento paralelo de los procesos lógicos del FPGA hace que los sistemas sean rápidos y robustos. En el FPGA, el ancho de los datos es definido por el desarrollador, de acuerdo a la conveniencia de la lógica de cada proceso que se implemente, mientras que en un procesador está función está limitada por el fabricante. En consecuencia, los filtros son de alto rendimiento y de rápida respuesta [6]. En este trabajo se presenta el diseño e implementación de un filtro digital de respuesta al impulso infinito (IIR) en un FPGA. El propósito del filtro es discriminar un intervalo de frecuencias de una señal eléctrica y así proporcionar un método de diseño e implementación a los estudiantes de cursos de Control Digital en ingeniería. El diseño del filtro se realizó usando el software Matlab usando la caja de herramientas de procesamiento de señales y la programación en lenguaje VHDL usando Design Suite para su implementación en un FPGA Spartan 6.

2. MÉTODO PROPUESTO

Un sistema LTI es denominado filtro de respuesta al impulso infinito o filtro IIR cuando su respuesta al impulso es infinita. La función de transferencia de un sistema discreto IIR está dado por:

$$H(z)=\frac{B(z)}{A(z)}=\frac{\sum_{n=0}^{M} b_n z^{-n}}{\sum_{n=0}^{N} a_n z^{-n}}=\frac{b_0+b_1 z^{-1}+\cdots+b_M z^{-M}}{1+a_1 z^{-1}+\cdots+a_N z^{-N}};\quad a_0=1 \qquad (1)$$

donde a_n y b_n son los coeficientes del filtro. El orden de cada filtro IIR es denominado N si $a_n \neq 0$. La representación de la ecuación de diferencias de un filtro IIR es expresada como:

$$y(n)=\sum_{m=0}^{M} b_m\, x(n-m)-\sum_{m=1}^{N} a_m\, y(n-m) \qquad (2)$$

La estructura del filtro que se implementará es la forma Directa. En esta forma, la ecuación de diferencias (2) es implementada directamente. Hay dos partes en un filtro IIR denominadas la parte de media móvil (MA, *moving average*) y la parte autorecursiva (AR, *autoregressive*), o equivalentemente la parte del numerador y del denominador.

Para describir la estructura de los filtros digitales IIR se requieren tres elementos: punto de suma, multiplicador y retardo unitario, los cuales se presentan en la Figura 1. El punto de suma es un elemento de dos entradas y una única salida, si se requiere la suma de tres o más elementos, esta se implementa con puntos de suma conectados en cascada. El multiplicador es un elemento de una-entrada, una-salida y representa el valor de una ganancia. El elemento de retardo unitario retrasa la señal un periodo de muestreo, este elemento es implementado usando un registro de corrimiento.

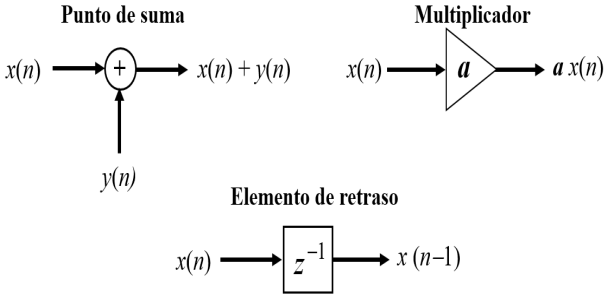

Figura 1. Elementos básicos en la implementación de filtros digitales.

2.1 DISEÑO DEL FILTRO IIR

El proceso de diseño del filtro digital IIR requiere en primer lugar establecer las especificaciones del filtro, tales como: tipo de filtro selectivo, la frecuencia de corte y la frecuencia de muestreo. Posteriormente se debe obtener la función de transferencia y el diagrama a bloques. El diagrama a bloques es útil para determinar los elementos de implementación en hardware, tales como los retrasos unitarios, multiplicadores y puntos de suma.

El diseño del filtro se realizó en el software Matlab usando la caja de herramientas de procesamiento digital de señales. Para efectos prácticos se seleccionó un filtro de segundo orden pasa-bajas Butterworth con una frecuencia de corte de 100 Hz y frecuencia de muestreo de 1 kHz. Para ello, se calculó la frecuencia de corte normalizada (ω/π). La Figura 2 presenta el código en Matlab para obtener la función de transferencia del filtro. En Matlab, la estructura de la forma directa del filtro es descrita por dos arreglos; donde la variable b contiene los coeficientes b_n y a contiene los coeficientes a_n.

Entonces, de acuerdo con los resultados obtenidos en Matlab, la función de transferencia del filtro es:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.0675 + 0.1349\,z^{-1} + 0.0675\,z^{-2}}{1 - 1.143\,z^{-1} + 0.4128\,z^{-2}} \qquad (3)$$

entonces la ecuación de diferencias es:

$$y(n) = 0.0675\,x(n) + 0.1349\,x(n-1) + 0.0675\,x(n-2) + 1.143\,y(n-1) - 0.4128\,y(n-2) \qquad (4)$$

El diagrama a bloques de la estructura directa del filtro en la forma tipo I se presenta en la Figura 3.

```
>> fc = 100;                    % frecuencia de corte
>> fs = 1000;                   % frecuencia de muestreo
>> ts = 1/fs;                   % periodo de muestreo
>> n = 2;                       % orden del filtro
>> w = 2*fc*ts;                 % frecuencia de corte normalizada
>> [b,a] = butter(n,w,'low')    % filtro pasa-baja de orden 2
   b =
       0.0675    0.1349    0.0675
   a =
       1.0000   -1.1430    0.4128
```

Figura 2. Código de diseño del filtro en Matlab.

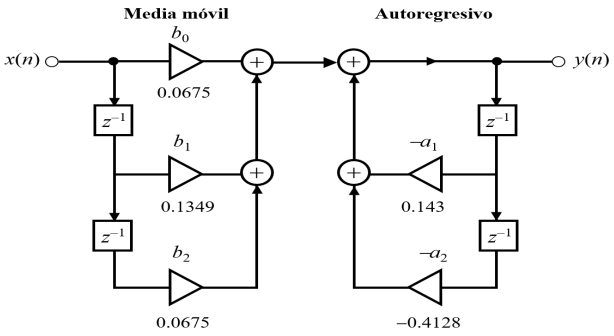

Figura 3. Estructura Directa tipo I del filtro IIR de segundo orden.

2.2 PROGRAMACIÓN DEL FILTRO IIR EN LENGUAJE VHDL POR FSM

Un FPGA está diseñado para trabajar con números enteros binarios (0 y 1), por lo que los coeficientes del filtro tienen que representarse en binario. Para ello se seleccionó una resolución de 32 bits, donde 16 bits son para la parte entera y 16 bits para la parte decimal. Mientras más bits se utilicen mejor será la aproximación del valor del coeficiente. La Tabla 1 presenta los resultados de la conversión a binario de los coeficientes del filtro.

Tabla 1. Valor de los coeficientes del filtro II de segundo orden en decimal y binario.

coeficiente	valor decimal	valor binario	
		parte entera	parte decimal
b_0	0.0675	0000 0000 0000 0000	0001 0001 0100 0111
b_1	0.1349	0000 0000 0000 0000	0010 0010 1000 1000
b_2	0.0675	0000 0000 0000 0000	0001 0001 0100 0111
a_1	1.143	0000 0000 0000 0001	0010 0100 1001 1011
a_2	0.4128	0000 0000 0000 0000	0110 1001 1010 1101

La ecuación de diferencias (4) es la que se implementó en el FPGA. En esta ecuación, los términos $x(n)$ y $y(n)$ representan los datos de entrada y salida actuales del filtro digital, y los términos $x(n-1)$ y $y(n-1)$ representan los datos de entrada y salida procesados un tiempo de muestreo anterior con respecto al actual, y finalmente $x(n-2)$ y $y(n-2)$ son los datos que fueron procesados dos instantes de muestreo antes del tiempo actual. Por lo tanto, es necesario guardar estos datos para que puedan ser procesados junto con la muestra actual $x(n)$ y $y(n)$. Para ello, se crean dos registros de corrimiento REGISTROX y REGISTROY ambos de longitud de 64 bits. El REGISTROX almacena los 32 bits de $x(n-1)$ (X1) y los 32 bits de $x(n-2)$(X2). Esto mismo se aplica para el REGISTROY que

almacena los datos de $y(n-1)$ (Y1) y de $y(n-2)$(Y2). Este almacenamiento se hace por corrimiento de datos, pues el lenguaje VHDL se ejecuta de manera concurrente y no secuencial.

La máquina de estados de la Figura 4 representa los procesos para la ejecución del filtro pasa-bajas IIR de segundo orden en VHDL.

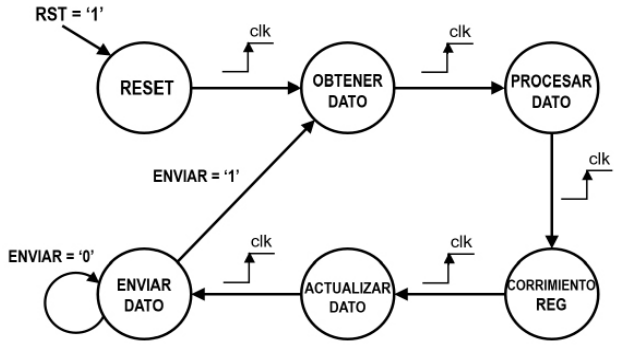

Figura 4. Máquina de estados finitos del filtro digital.

El estado RESET es el que borra el contenido de los registros REGISTROX, REGISTROY y las señales X1, X2, Y1 y Y2 cuando una señal de entrada RST se encuentre activa en el estado lógico 1. La Figura 5 presenta el código en VHDL del proceso RESET.

```
IF RISING_EDGE (CLK) THEN
    IF RST = '1' THEN
       EDO <= RESET;
    ELSE
       CASE EDO IS
           WHEN RESET =>
              Y  <= (OTHERS => '0');
              X1 <= (OTHERS => '0');
              X2 <= (OTHERS => '0');
              Y1 <= (OTHERS => '0');
              REGISTROX <= (OTHERS => '0');
              REGISTROY <= (OTHERS => '0');
              EDO <= OBTENER_DATO;
```

Figura 5. Código del proceso RESET en VHDL.

En el estado OBTENER DATO se captura la muestra de entrada obtenida de un convertidor analógico-digital (ADC) para almacenarla en X. El ADC que se usó en este trabajo entrega un dato de 8 bits, y la resolución de X es de 32 bits se concatenan los 24 bits restantes de la manera en que se presenta en la Figura 6.

```
WHEN OBTENER_DATO =>
       --|----entera----||----decimal-----|
    X <= ("00000000"&ADC&"00000000000000000");
    EDO <= PROCESAR_DATO;
```

Figura 6. Código del proceso OBTENER DATO en VHDL.

En el estado PROCESAR DATO se ejecuta la ecuación de diferencias (4) para obtener el valor de Y, que representa a $y(n)$. El resultado de esta operación genera un registro de 64 bits. Este proceso se presenta en la Figura 7.

```
WHEN PROCESAR_DATO =>
    Y <= (b0*X)+(b1*X1)+(b2*X2)+(a1*Y1)-(a2*Y2);
    EDO <= CORRIMIENTO_REG;
```

Figura 7. Código del proceso PROCESAR DATO en VHDL.

El estado CORRIMIENTO REG es el que se encarga de ejecutar el corrimiento a la izquierda de los bits de los registros REGISTROX y REGISTROY. El corrimiento a la izquierda se hace por paquetes de 32 bits, es decir, para el REGISTROX los 32 bits menos significativos (31 downto 0) se recorren a los 32 bits más significativos (63 downto 32), de tal manera que los 32 menos significativos se actualizan con el valor de X, que es el dato de entrada que se lee del ADC. La lectura del ADC se ejecutó en el estado OBTENER DATO. Para el caso del REGISTROY, de igual manera los 32 bits menos significativos (31 downto 0) se desplazan a los 32 bits más significativos (63 downto 32), entonces los 32 bits menos significativos se actualizan con el valor de Y que se ejecutó en el estado PROCESAR DATO. Dado que Y es un registro de 64 bits, para actualizar los 32 bits menos significativos de REGSTROY solo se consideran los 32 bits centrales del registro Y (16 bits para la parte entera y 16 bits para la parte decimal), el resto de los bits se desprecian para que en la siguiente multiplicación Y se mantenga como un vector de 32 bits. Este proceso se presenta en el código de la Figura 8.

```
WHEN CORRIMIENTO_REG =>
    --X2 = X1
    REGISTROX(63 DOWNTO 32) <= REGISTROX(31 DOWNTO 0);
    --X1 = X
    REGISTROX(31 DOWNTO 0) <= X;
    --Y2 = Y1
    REGISTROY(63 DOWNTO 32) <= REGISTROY(31 DOWNTO 0);
    --Y1 = Y(K)
    REGISTROY(31 DOWNTO 0) <= Y(47 DOWNTO 16);
    EDO <= ACTUALIZAR_DATOS;
```

Figura 8. Código del proceso CORRIMIENTO REG en VHDL.

En el estado ACTUALIZAR DATOS se separa el contenido de REGISTROX y REGISTROY en paquetes de 32 bits para ser almacenados en los registros X1, X2, Y1 y Y2, tal como se presenta en la Figura 9.

```
WHEN ACTUALIZAR_DATOS =>
    X2 <= REGISTROX(63 DOWNTO 32);
    X1 <= REGISTROX(31 DOWNTO 0);
    Y2 <= REGISTROY(63 DOWNTO 32);
    Y1 <= REGISTROY(31 DOWNTO 0);
    EDO <= MANDAR_DATO;
```

Figura 9. Código del proceso ACTUALIZAR DATOS en VHDL.

El último estado ENVIAR DATO se encarga de enviar el dato se salida Y al DAC. Para respetar la frecuencia de muestreo de diseño de 1 kHz, los datos se envían al DAC cada 1 ms. Para esto se ejecuta un contador en un proceso adicional, el cual activa la señal ENVIAR cada que transcurre ese tiempo. El proceso del estado ENVIAR DATO se presenta en la Figura 10.

```
WHEN ENVIAR_DATO =>
    IF ENVIAR = '1' THEN
       EDO <= OBTENER_DATO;
       DAC <= Y(39 DOWNTO 32);
    ELSE
       EDO <= ENVIAR_DATO;
    END IF;
```

Figura 10. Código del proceso ENVIAR DATO en VHDL.

3 RESULTADOS

3.1 Implementación en hardware del filtro digital IIR

La implementación en hardware del filtro IIR se realizó en la tarjeta de desarrollo Avanxe [7], que se presenta en la Figura 11. Esta tarjeta contiene el chip FPGA XC6SLX16 Spartan-6 de Xilinix y un PSoC 3 de Cypress y trabaja con un circuito de reloj de 50 MHz. El PSoC integra un ADC y un DAC de 8 bits y usa el protocolo de comunicación SPI. El esquema estructural de la implementación del filtro IIR en la tarjeta Avanxe se presenta en la Figura 12. Para usar el PSoC es necesario incorporar una Interfaz Analógica que obtenga los datos del ADC y envíe los datos al DAC.

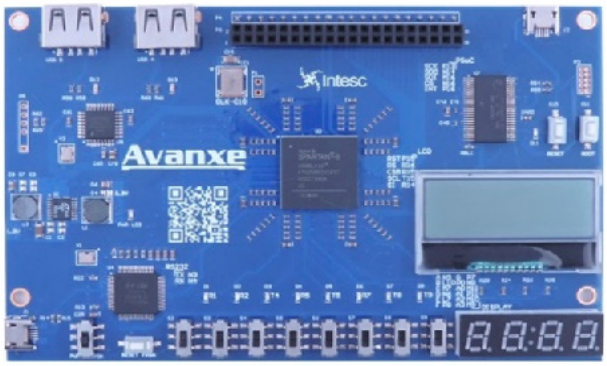

Figura 11. Tarjeta de desarrollo Avanxe [7].

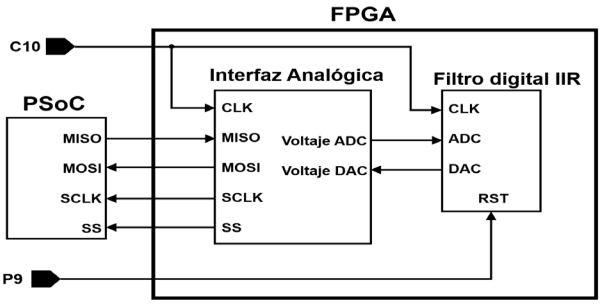

Figura 12. Esquema estructural del filtro IIR en la tarjeta de desarrollo Avanxe.

Los puertos voltajeDAC y voltajeADC son de 10 bits aunque sólo se toman los 8 bits que contienen la información, los bits restantes se utilizan como bits de interrupción. Para calcular la velocidad de transmisión-recepción se hizo la simulación con una frecuencia de reloj (CLK) a 50MHz. El periodo de SCLK es de 220ns, esto significa que cada bit se envía y se recibe cada 220ns. Como el protocolo utiliza 10 bits entonces cada paquete de bits tarda $2.2\mu s$ (254 Khz). Se deben considerar estos tiempos para futuras aplicaciones en Avanxe.

La Figura 13 presenta el mapa de conexiones de la tarjeta Avanxe, en donde se muestra que el circuito de reloj se conecta al pin C10, la entrada física RST (reset) se conectará al interruptor en P9, y los pines del PSoC3. La señal de entrada al ADC se conecta al canal PQ[0] y la señal del DAC se obtiene por el canal PQ[1].

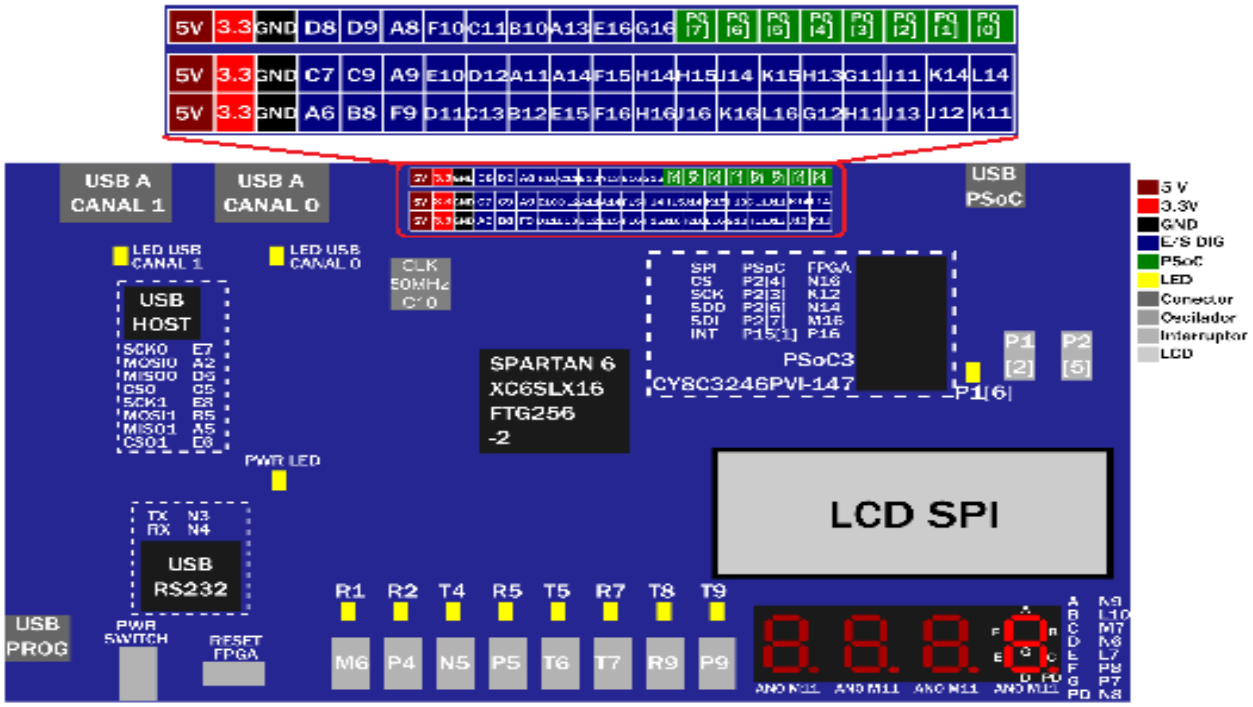

Figura 13. Mapa de conexiones de la tarjeta Avanxe [7].

3.2 Prueba 1: respuesta en frecuencia

Para comprobar el desempeño del filtro digital se realizó una primera prueba experimental, que consistió en comprobar la respuesta en frecuencia del filtro. Para esto se conectó directamente la salida de un generador de funciones a la entrada del ADC de la tarjeta y se observó en un osciloscopio la salida del filtro, tal como ilustra el esquema de la Figura 14. Como se trata de un filtro pasa-bajas de segundo orden, con frecuencia de corte de 100 Hz, de manera ideal todas las señales de frecuencias por debajo de 100 Hz deben salir igual a la señal de entrada, pero señales con frecuencias superiores a 100 Hz son atenuadas. El comportamiento real de la señal de salida tiene que ser similar al que se observa en la Figura 15, en la que se presenta la gráfica de la magnitud de la respuesta en frecuencia en Matlab. La prueba inició configurando el generador a un $1V_{pico\text{-}pico}$ (Vpp) a una frecuencia de 20 Hz y se comenzó a subir hasta 200 Hz. Ejemplos de 4 imágenes observadas en un osciloscopio DS-212 de dos canales simultáneos se observan en la Figura 16. La señal de color amarillo corresponde a la salida del filtro digital (CH-2), proveniente del DAC, y en cada caso se observa como esta señal se va atenuando conforme se aumenta la frecuencia, por lo que se comprueba el comportamiento del filtro pasa-bajas.

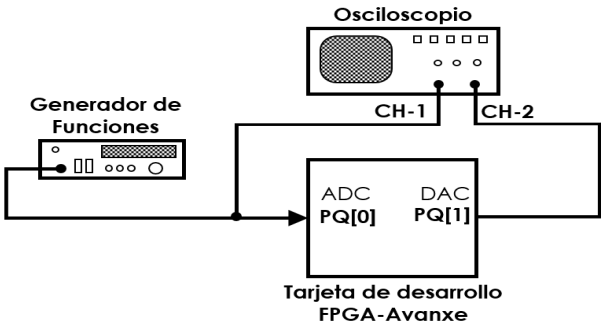

Figura 14. Esquema de conexión de la prueba 1.

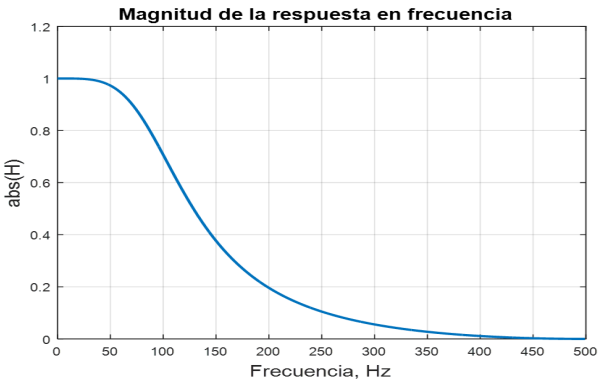

Figura 15. Gráfica de la magnitud del filtro IIR pasa-bajas de orden 2.

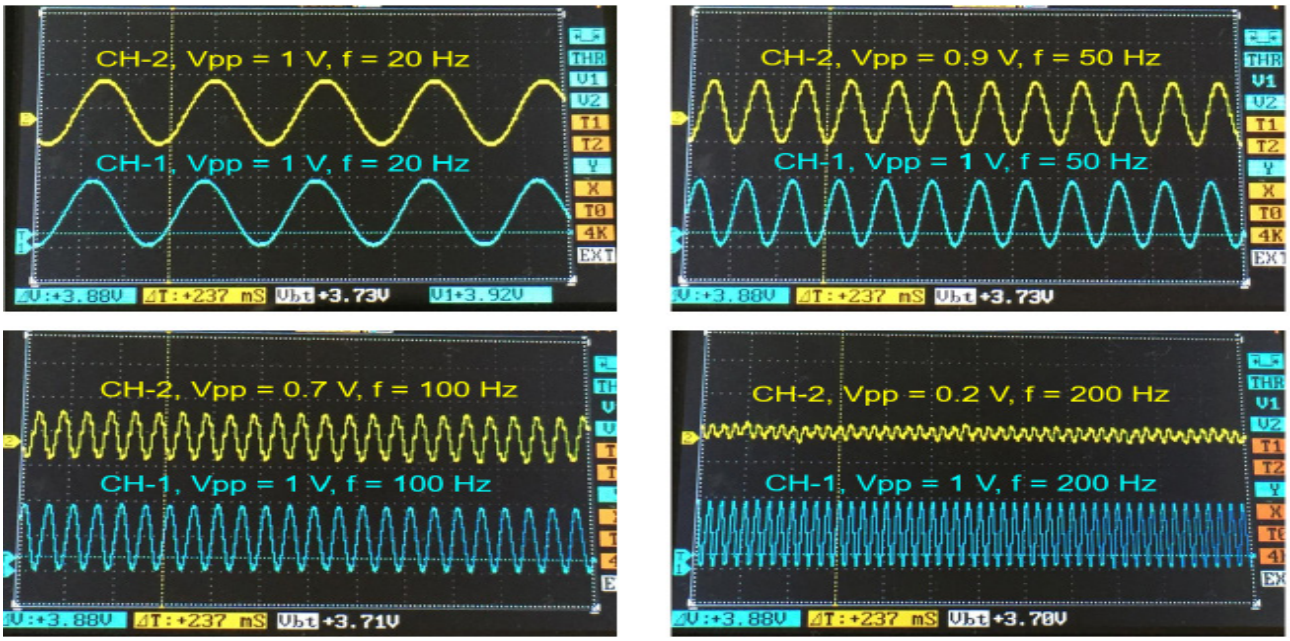

Figura 16. Resultados de la prueba 1 del filtro IIR en el osciloscopio.

3.2 Prueba 2: aplicación didáctica

Se realizó una segunda prueba para comprobar una práctica experimental, en la que se sumaron dos señales tipo senoidales a diferentes frecuencias, tal como se presenta en el esquema de la Figura 17. La Figura 18 muestra los resultados de la simulación en Matlab y la Figura 19 presenta los resultados de las señales visualizadas en un osciloscopio Tektronix de 2 canales simultáneos. La suma analógica de las señales se observa en color azul y se obtuvo de la salida de un amplificador operacional $R = 1\ k\Omega$. configuración de sumador no inversor, con R = 1 k La señal de color amarillo es la salida del filtro proveniente del DAC.

Para verificar los resultados obtenidos se realizó una comparación de los datos obtenidos en Matlab y los datos que entrega el código del filtro en VHDL. Para ello, en Matlab se ajustaron los datos de la señal de entrada a la resolución del ADC de 8 bits, donde el valor máximo de la conversión es un dato de 255. La Tabla 2 presenta una comparación de los 7 primeros datos de salida obtenidos en la simulación de Matlab (ydig) y en ISE-Design (dac) para los mismos valores de entrada ADC (0-255). La Figura 20 presenta evidencias de estos resultados en el simulador de ISE-Design. Considerando los datos de Matlab como los valores reales, a partir de una muestra de 100 datos se calculó que los resultados obtenidos por la implementación del filtro IIR digital en VHDL tiene un 99.8% de exactitud.

Tabla 2. Comparación entre los datos de salida en la simulación de Matlab e ISE.

Simulación	Valores de entrada						
	155	186	160	169	219	200	189
Matlab (ydig)	10	45	94	134	162	184	197
ISE-Design (dac[7:0])	10	45	93	134	162	184	197

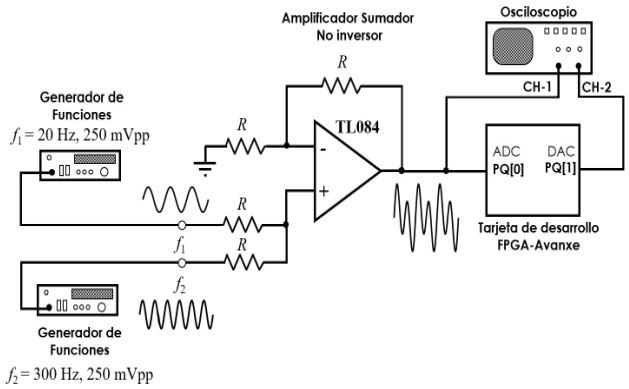

Figura 17. Esquema del ejemplo práctico del filtro digital IIR.

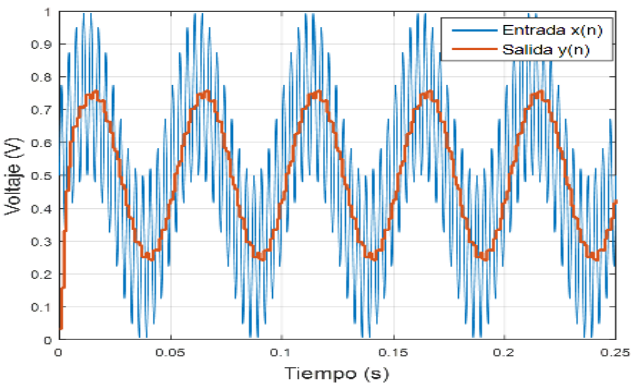

Figura 18. Resultados del ejemplo práctico del filtro IIR en simulación en Matlab.

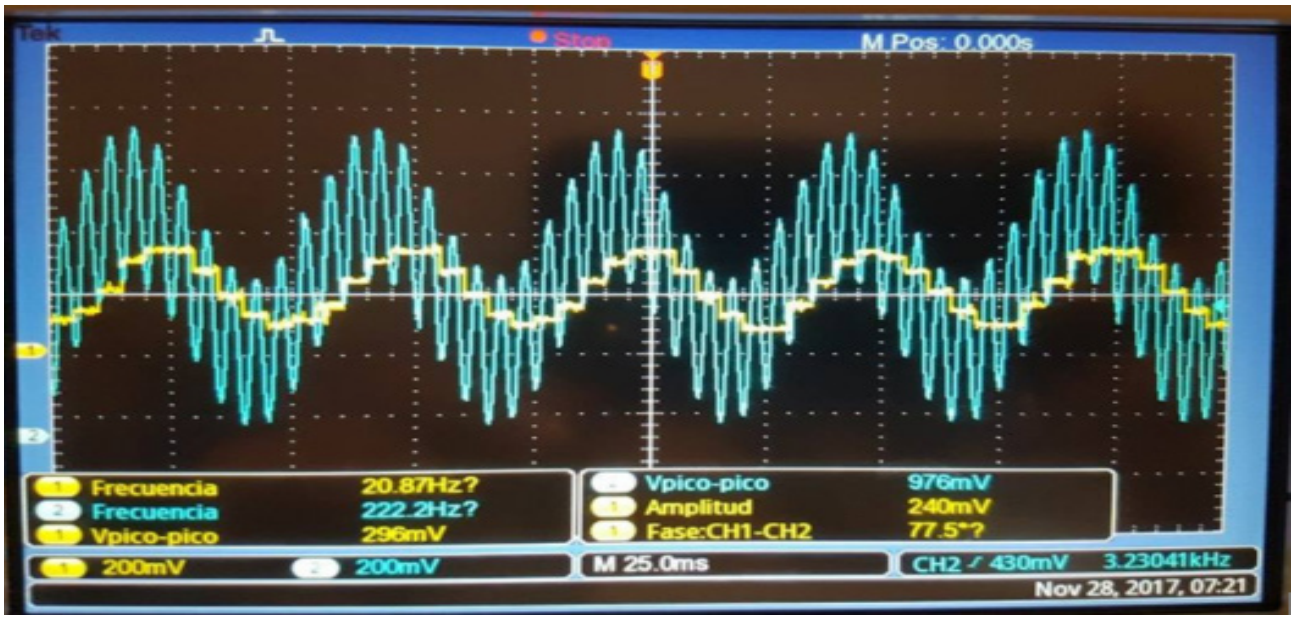

Figura 19. Resultados del ejemplo práctico del filtro IIR en el osciloscopio.

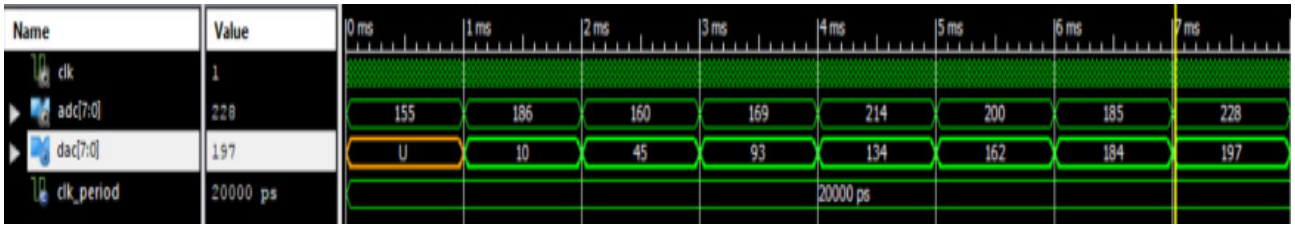

Figura 20. Comparación entre los datos de salida en la simulación de Matlab e ISE.

CONCLUSIONES

Se presentó la descripción de la implementación de un filtro digital IIR pasa bajas de segundo orden en un FPGA usando una máquina de estados finitos. El uso de un FPGA permite diseñar y definir las características del hardware a implementar, lo cual lo vuelve una opción muy viable para las aplicaciones didácticas en ingeniería, que promueven el uso de las tecnologías digitales. El desempeño del filtro se comprobó con dos pruebas en la que los estudiantes pueden verificar de manera experimental los conceptos teóricos en la práctica. La comparación de los resultados entre las simulaciones realizadas en el software Matlab y el simulador de ISE-Design comprueban que el método presentado por máquina de estados finitos es confiable con un 99.8% de exactitud.

Esta propuesta puede extenderse a otras aplicaciones del procesamiento digital de señales o controladores digitales, pues se puede aplicar a cualquier sistema expresado mediante una ecuación de diferencias, tales como filtros pasa-altas, pasa-banda, rechazo de banda de diferente orden y a controladores digitales como los PID.

REFERENCIAS

1. Weeks, M. Digital signal processing using MATLAB & wavelets. Canada: Jones & Bartlett Publishers, 2010.
2. Toledo, D. C., Martínez, M. A., Rodríguez, J., Arriaga, S. T., Márquez, M. A. IIR digital filter design implemented on FPGA for myoelectric signals. In: 2017 XIII International Engineering Congress (CONIIN). Santiago de Queretaro-México, 2017, 1-7.
3. Zhao, C., Zhang, Z. Digital filter design and performance analysis of dynamic temperature signal denoise based on FPGA. In: 2016 10th International Conference on Sensing Technology (ICST). Nanjing, China, 2016, 1–7.
4. Costa, D., Páez, C. S. A comparative analysis of hardware techniques for implementation of IIR digital filter on FPGA. In: 2015 XVI Workshop on Information Processing and Control (RPIC). Cordoba, Argentina, 2015, 1–6.
5. Seshadri, R., Ramakrishnan, S. FPGA implementation of fast digital FIR and IIR filters. Concurrency and Computation: Practice and Experience, 2019, e5246, doi: 10.1002/cpe.5246.
6. Rengaprakash, S., Vinesh, M., Syed, N., Pragadheesh, M., Senthilkumar, E., Sandhya M., Manikandan, J. FPGA implementation of fast running FIR filters. In: 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). Chennai, India, 2017, 1282-1286.
7. Intesc Electrónica y Embebidos. Avanxe [en línea]. México: Intesc. Recuperado el 26 de noviembre de 2020 de https://www.intesc.mx/productos/avanxe/

Acerca de los autores

La Dra. Mariana Natalia Ibarra Bonilla recibió el título de Ingeniería en Electrónica en 2006 por el Instituto Tecnológico de Veracruz, en 2009 obtuvo el grado de Maestría en Ciencias y en 2015 el grado de Doctorado en Ciencias en la especialidad de Electrónica, ambos por el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), en donde trabajó en el área de Instrumentación con proyectos de control y procesamiento de señales. Actualmente labora como profesora de Tiempo Completo en la división de Ingeniería Mecatrónica del Instituto Tecnológico Superior de Atlixco, en donde desempeña principalmente actividades de docencia e investigación, siendo líder de la línea de investigación y del cuerpo académico "Automatización de Procesos". Las áreas de especialidad que desempeña hoy en día son: sistemas de control, sistemas inteligentes, lógica difusa y sistemas digitales embebidos con microcontroladores y FPGAs.