IMPLEMENTACIÓN HARDWARE DE UN FILTRO FIR CON ARITMÉTICA DISTRIBUIDA USANDO ARQUITECTURAS SERIE Y PARALELO

FIR FILTER HARDWARE IMPLEMENTATION WITH DISTRIBUTED ARITHMETIC USING SERIAL AND PARALLEL ARCHITECTURES

Luis F. Gaona C.
Tecnológico Nacional de México en Celaya
m1803017@itcelaya.edu.mx

Luis E. Ortega G.
Tecnológico Nacional de México en Celaya
m1803049@itcelaya.edu.mx

Israel M. Lemus
Tecnológico Nacional de México en Celaya
m1803018@itcelaya.edu.mx

Carlos E. García A.
Tecnológico Nacional de México en Celaya
m1803052@itcelaya.edu.mx

Agustín Ramírez A.
Tecnológico Nacional de México en Celaya
agustin.ramirez@itcelaya.edu.mx

Resumen

En este artículo se reporta el resultado de una comparativa entre las implementaciones hardware de dos arquitecturas basadas en aritmética distribuida. Una implementación utiliza una arquitectura completamente paralela y la otra una arquitectura serie. Para ello, las implementaciones de la aritmética distribuida se aplican a un filtro FIR de cuarto orden. Las implementaciones se realizan sobre un FPGA de la familia Spartan 3E de Xilinx. La comparativa se lleva a cabo tomando en consideración la ocupación de recursos, la máxima frecuencia de operación y la cadencia de funcionamiento para los dos sistemas desarrollados.
Palabra(s) Clave: Filtro de respuesta de impulso finito, aritmética distribuida, Implementación hardware, FPGA.

Abstract

This paper exposes a comparison between the hardware implementations of two distributed arithmetic based architectures. The first implementation uses a completely parallel architecture and the second a sequential architecture. Both implementations are applied in a fourth order FIR filter. The implementations are developed over a Xilinx Spartan 3E family FPGA. The comparison is made taking in account resource occupation, maximum operating frequency, and the throughput of both systems.

Keywords**:** *Finite Impulse Response Filter, Distributed Arithmetic, Hardware implementation, FPGA.*

1. Introducción

Originalmente, la aritmética distribuida fue propuesta como una técnica de procesamiento en serie de los datos con el fin de implementar algoritmos basados en la suma de productos con un reducido costo en área. La misma técnica permite aumentar la velocidad de cómputo de los datos, procesando dígitos (agrupaciones de bits), en serie o el dato completo en paralelo, a costa de un incremento en el área de los circuitos que lo implementan. El propósito de la aritmética distribuida es el de realizar las operaciones de multiplicación sin utilizar directamente multiplicadores.

Los filtros FIR se utilizan en la mayoría de las aplicaciones de tratamiento digital de la señal. Para su implementación en FPGA, una alternativa es utilizar aritmética distribuida debido principalmente a que los dispositivos poseen abundantes recursos de almacenamiento distribuidos por todo el chip y a la facilidad de automatizar el proceso de implementación.

Los filtros adaptativos son ampliamente utilizados para procesamiento de señales. Los filtros FIR basados en el algoritmo de media cuadrática mínima (LMS) son de los más utilizados por su simplicidad y desempeño satisfactorio, pero no es el más apropiado para su implementación cuando la velocidad de muestreo es alta [1].

En [2] se propone una técnica de optimización del diseño en dos pasos de una arquitectura VLSI para un filtro de interpolación con la cual se reduce inicialmente

el número de multiplicaciones y sumas por muestreo en un 83%, en comparación con una implementación individual de un filtro estándar. Además se propone un algoritmo para la eliminación de sub-expresiones comunes binarias de 2 bits, logrando reducir el área y consumo de potencia en 41% y 38%, respectivamente.

Para la implementación de la Transformada Wavelet Discreta (DWT) se utiliza la arquitectura basada en aritmética distribuida (DAA), es decir, los filtros FIR de paso bajo (LPF) y los filtros FIR de paso alto (HPF) utilizados en la DWT se implementan usando DAA.

En [3] se realiza una comparación entre el sistema DWT basado en la arquitectura con multiplicador convencional (CMA) y el sistema DWT basado en DAA; aunque este último es de bajo costo, ya que no requiere el módulo DSP, se recomienda su uso en aplicaciones de baja velocidad, en tanto que la velocidad puede verse comprometida.

En [4] se analiza el diseño de filtros FIR de baja potencia y las implementaciones de éstos en FPGA para aplicación en el procesamiento de señales ECG, se concluye que la implementación basada en aritmética distribuida permite lograr una eficiencia computacional importante al reducir el número total de compuertas entre un 50 y 80%, lo que también disminuye el consumo de potencia total.

En [5] se realiza la aplicación de una arquitectura de hardware eficiente basada en aritmética distribuida para la implementación de un filtro de respuesta al impulso finita adaptativa (FIR) de mínimos cuadrados medios (LMS). Se concluye que debido a la implementación sin multiplicadores, la arquitectura es más eficiente en lo relativo a la utilización del área, consumo de potencia y velocidad.

En [6] se presenta una arquitectura que utiliza aritmética distribuida en paralelo para un filtro FIR. La arquitectura se basa en compresores 4:2 que se implementan eficientemente en FPGAs de Xilinx. En promedio se obtiene una reducción del 17.5% en el uso de recursos y una mejora de 20.7% en cuanto a desempeño en comparación con filtros FIR con arquitectura directa. También obtienen una reducción en el uso de recursos del 57.9% y un mejor desempeño del 23% en comparación con filtros generados con la herramienta Coregen de Xilinx.

Muchos esquemas han sido desarrollados para implementar filtros basados en arquitecturas de DA. Uno de estos esquemas es llamado, Offset Binary Coding (OBC), donde la característica principal es el tamaño de la LUT es reducido a la mitad, justamente para reducir el área utilizada y el consumo de energía, también se ha propuesto un método de compartimiento de LUTs para generar la salida del filtro para actualizar la ponderación del filtro en filtros adaptativos, los cuales significativamente reducen el área total del filtro adaptativo. También se ha propuesto reducir un cierto nivel las LUT en algunas menores, pero esto se traduce en que se requieren sumadores adicionales para combinar estas pequeñas LUTs, lo que también incrementa la potencia dinámica del diseño [7].

Un filtro FIR reconfigurable cuyos coeficientes de filtrado cambian dinámicamente durante el tiempo de ejecución juega un papel muy importante en los sistemas de radio definidos por software, filtros multicanales, y convertidores digitales de subida/bajada. La implementación convencional de la DA usada para la implementación del filtro FIR, asume que los coeficientes de respuesta al impulso son fijos y justo este comportamiento hace posible el uso de LUTs basadas en ROM. La memoria requerida para una implementación de un filtro FIR basado en DA, crece exponencialmente con el orden del filtro. Para eliminar el problema de los requerimientos de grandes cantidades de memoria, la técnica de descomposición sistólica es sugerida para la implementación de bloques de convolución de filtros FIR de alto orden basados en DA [8].

2. Desarrollo

La técnica de aritmética distribuida

Un filtro FIR con coeficientes constantes de orden N está caracterizado por la siguiente ecuación 1.

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot s(n-k) \qquad (1)$$

Siendo, respectivamente, $h(k)$ y $s(n-k)$; $k = 0, 1, \cdots N-1$ los N coeficientes del filtro y las N muestras más recientes de la señal de entrada, respectivamente.

Para simplificar la derivación de la técnica de aritmética distribuida, en la ecuación (1) se omite el parámetro temporal n, dando lugar a la ecuación 2.

$$y = \sum_{k=0}^{N-1} h(k) \cdot x(k) \tag{2}$$

Donde $x(k) = s(n-k)$.

Si $x(k)$ se representa mediante un número binario con signo en complemento a 2 de D bits, la ecuación 2 se puede reescribir en ecuación 3.

$$y = \sum_{k=0}^{N-1} h(k)[-2^{D-1}x_{D-1}(k) + \sum_{d=0}^{D-2} 2^d x_d(k)] \tag{3}$$

$$= -2^{D-1} \sum_{k=0}^{N-1} h(k)(x_{D-1}(k) + \sum_{d=0}^{D-2} 2^d \sum_{k=0}^{N-1} h(k)\, x_d(k)$$

Siendo $x_d(k)$ el bit d del valor de la muestra $x(k)$, para $d = 0, 1, \ldots D-2$ y $x_{D-1}(k)$ el bit más significativo de esa misma muestra, es decir el bit de signo.

En tanto que los valores posibles para $x_d(k)$ son sólo *1* o *0*, las multiplicaciones se convierten en sumas de los valores de los coeficientes del filtro desplazados ciertas posiciones para incorporar el factor 2^d.

Por lo tanto, una manera eficiente de implementar filtros FIR sobre un FPGA es utilizar aritmética distribuida, en tanto que ésta se basa en almacenar los productos parciales de las multiplicaciones en tablas, evitando así el uso de multiplicadores, los cuales representan una reducción de la velocidad y un incremento del área. Es decir, la aritmética distribuida, de acuerdo con la ecuación 3, se basa en el cálculo de una suma de productos sin utilizar multiplicadores (figura 1). El tiempo de computación es de D ciclos de reloj, donde D es el tamaño de palabra utilizado para las señales de entrada. Esta memoria se implementa en la FPGA de manera distribuida, utilizando las LUT.

De acuerdo con lo anterior, las operaciones a realizar son: una secuencia de búsquedas en tablas, sumas y desplazamientos. Todas estas operaciones se pueden implementar en un FPGA de forma eficiente. El tamaño de la tabla puede llegar a ser muy grande. Para mejorar la velocidad se puede aumentar el número

de bits por entrada. De este modo se trabaja con Digit-Serial y tamaño de dígito WD. En este caso se obtiene una salida cada D/WD ciclos de reloj. La máxima velocidad se obtiene cuando WD es igual a D, que es el caso paralelo.

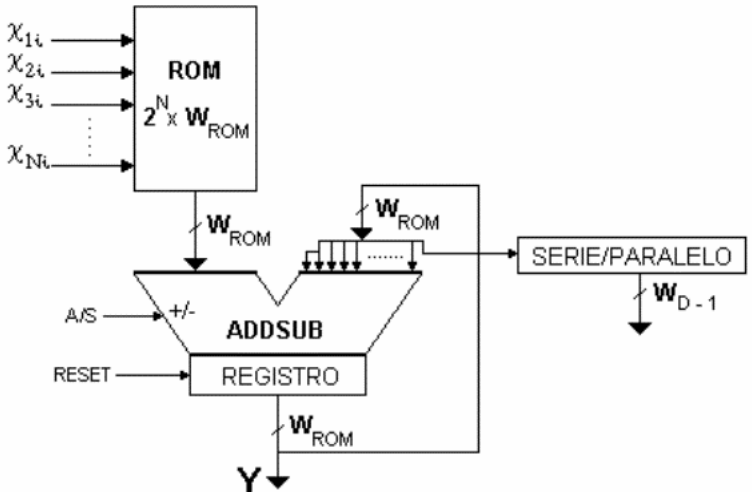

Figura 1 Suma de productos con aritmética distribuida.

Las dos implementaciones desarrolladas se realizaron para longitud de palabra $D = 8$. Asimismo, en tanto que el filtro es de orden $N = 4$, las 16 diferentes combinaciones que se pueden formar con las sumas de los cuatro coeficientes del filtro se almacenan en una memoria ROM, de la manera que muestra tabla 1.

Tabla 1 Memoria ROM con suma de coeficientes.

Address	Data	Address	Data
0	0	8	$h(3)$
1	$h(0)$	9	$h(3)+h(0)$
2	$h(1)$	10	$h(3)+h(1)$
3	$h(1)+h(0)$	11	$h(3)+h(1)+h(0)$
4	$h(2)$	12	$h(3)+h(2)$
5	$h(2)+h(0)$	13	$h(3)+h(2)+h(0)$
6	$h(2)+h(1)$	14	$h(3)+h(2)+h(1)$
7	$h(2)+h(1)+h(0)$	15	$h(3)+h(2)+h(1)+h(0)$

Estructura secuencial

Para la estructura secuencial se utilizó la arquitectura que se muestra en la figura 2. Se implementó la máquina de estados de la figura 3 para experimentación.

La síntesis produjo el diagrama RTL (Nivel de Transferencia de Registro) de primer nivel (figura 4) y el RTL de segundo nivel (figura 5).

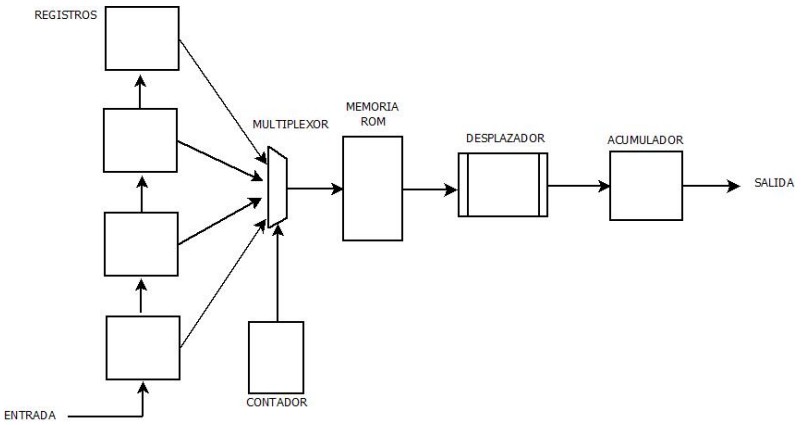

Figura 2 Arquitectura de la implementación secuencial.

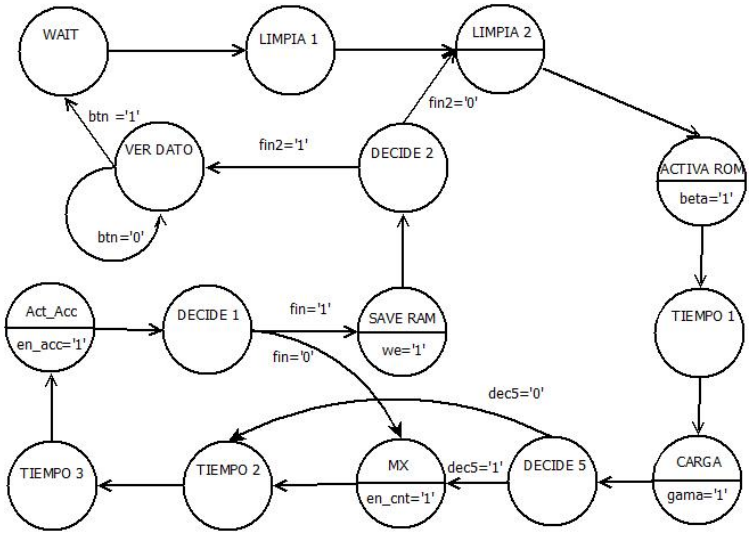

Figura 3 Diagrama de transición de estados para experimentación (estructura secuencial).

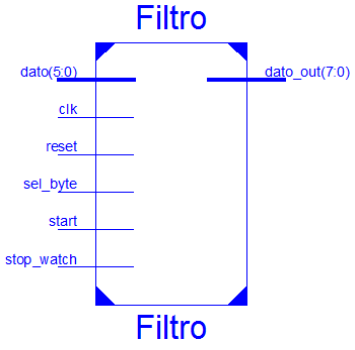

Figura 4 RTL nivel 1, estructura secuencial.

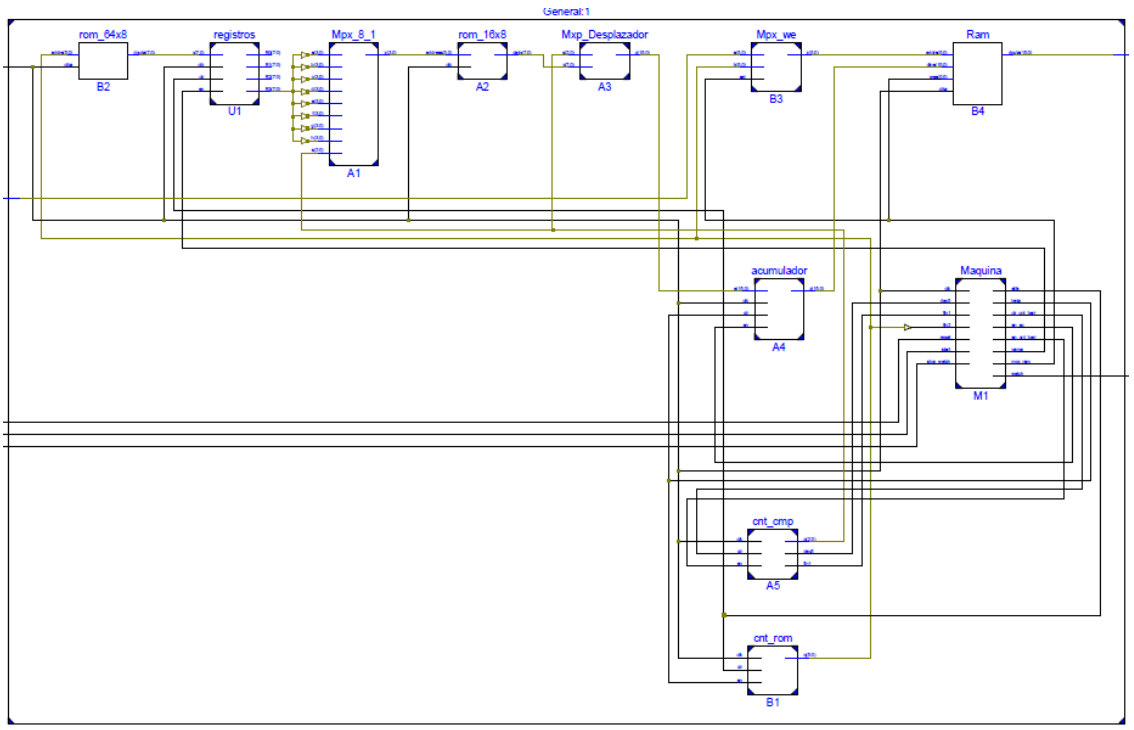

Figura 5 RTL nivel 2, estructura secuencial.

La arquitectura secuencial sólo utiliza una memoria para almacenar las 16 diferentes combinaciones de sumas de los coeficientes y además incorpora los siguientes bloques:

- Registros, almacenan los cuatro valores más recientemente muestreados de la señal de entrada, de manera que los valores se van desplazando de uno a otro a medida que se obtienen las muestras.
- Contador de 0 a 7, proporciona el índice del bit de los datos que se están procesando (d).
- Multiplexor, en este bloque, por medio del contador se canalizan los bits de igual índice de los datos de entrada para formar la dirección que se proporciona a la memoria ROM.
- Memoria ROM, entrega el valor previamente calculado de la suma de los coeficientes del filtro en función del valor (‘0’ o ‘1’) de cada uno de los bits del mismo orden en las 4 muestras más recientes de la señal de entrada.
- Acumulador, en este bloque, para cada uno de los bits de las muestras (0 a 7) se van acumulando las sumas desplazadas y así se obtiene la salida.

- Registro de desplazamiento, la suma de coeficientes se desplaza la cantidad de posiciones requeridas para realizar la multiplicación por el factor 2^d presente en la ecuación 3.

Como se puede observar en las figuras 3 a 5, los resultados que se presentan se obtienen para propósitos de experimentación, pero para el esquema de funcionamiento el diagrama de transición de estados de la figura 3 cambia como se muestra en la figura 6.

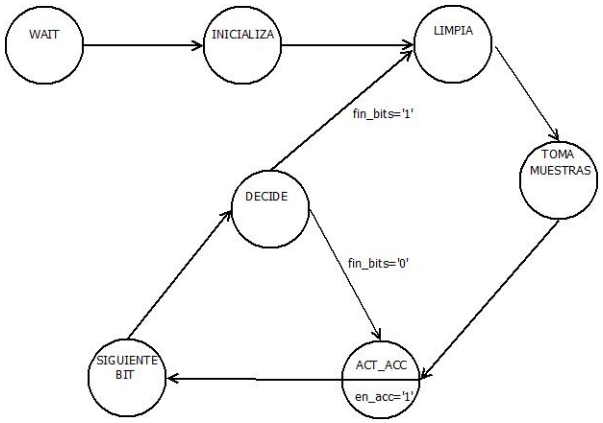

Figura 6 Transición de estados esquema de funcionamiento, estructura secuencial.

Estructura paralelo

Para la estructura en paralelo se utiliza la arquitectura que se muestra en la figura 7. El control se realiza mediante la máquina de estados de la figura 8 para propósitos de experimentación, después de la síntesis se obtuvieron los diagramas RTL de primer nivel (figura 9) y segundo nivel (figura 10). En esta estructura se utilizan ocho bloques de memoria ROM y ocho registros de desplazamiento, esto debido a que los 8 bits de las muestras se procesan simultáneamente. Asimismo, en lugar del acumulador se tiene un sumador que en paralelo suma los 8 valores proporcionados por los registros de desplazamiento. Como se puede observar en las figuras 8 a 10, los resultados que se presentan se obtienen para propósitos de experimentación, pero para esquemas de funcionamiento el diagrama de transición de estados de la figura 8 cambia como se muestra en la figura 11.

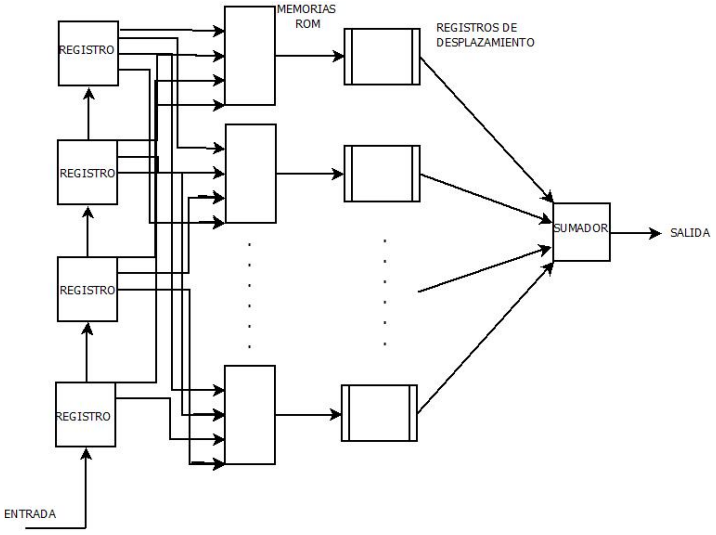

Figura 7 Estructura para procesamiento en paralelo.

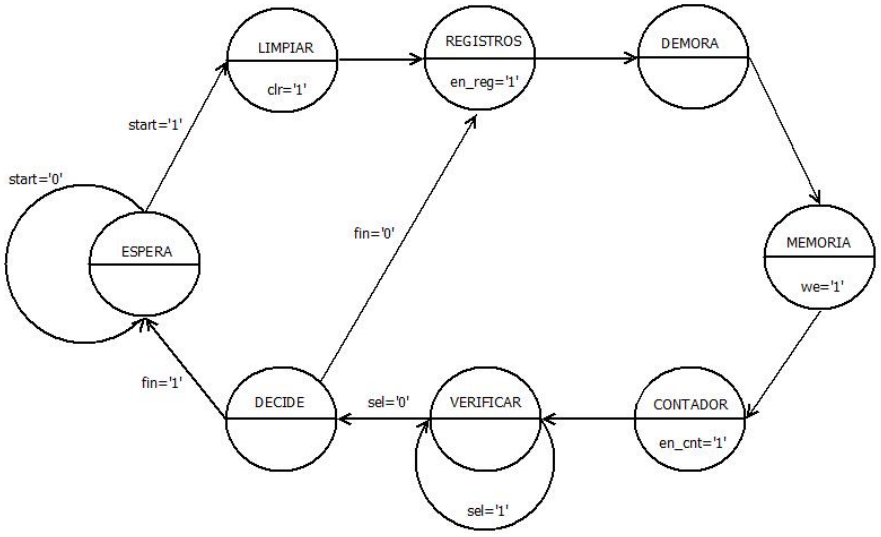

Figura 8 Transición de estados, esquema de experimentación, estructura paralelo.

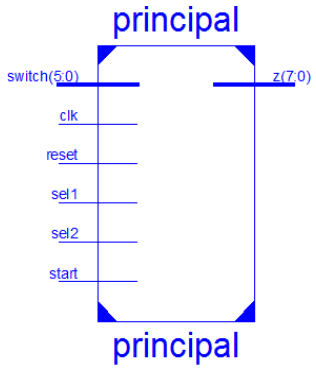

Figura 9 RTL nivel 1, estructura paralelo.

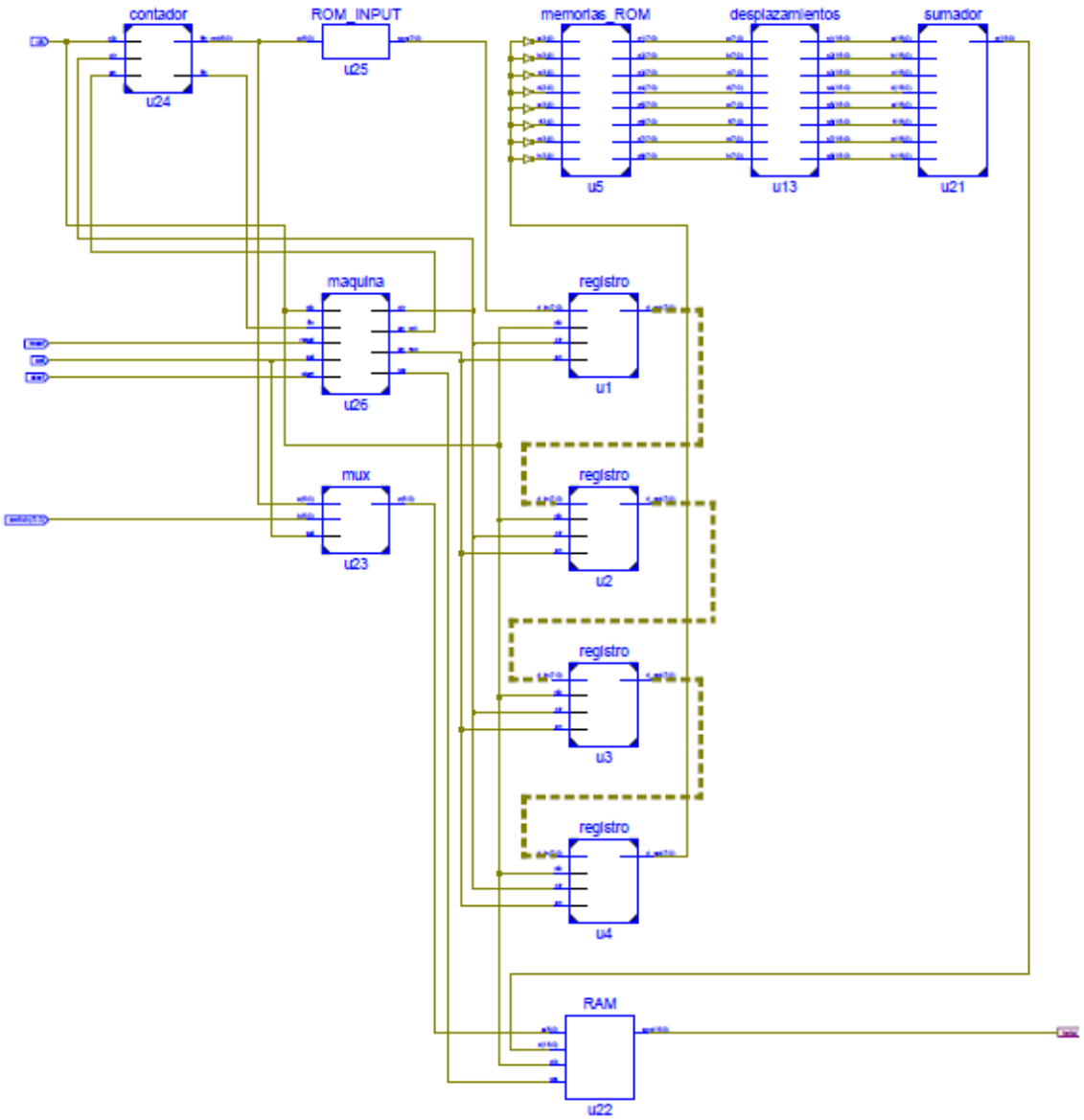

Figura 10 RTL nivel 2, estructura paralelo.

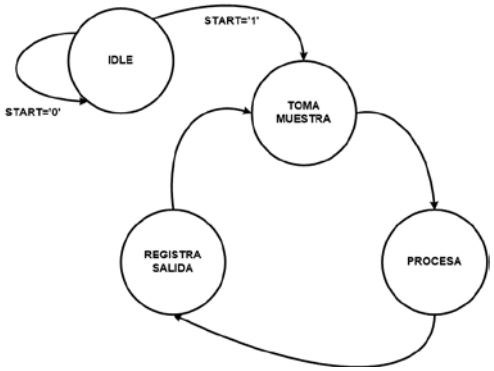

Figura 11 Transición de estados, procesamiento paralelo (esquema de funcionamiento).

3. Resultados

Se experimentó con las implementaciones hardware de un filtro FIR con las siguientes características, orden 4, pasa bajas, frecuencia de corte 1000 Hz frecuencia de muestreo 10,000 Hz; para el diseño del filtro se utilizó la técnica de ventaneo de Káiser utilizando una valor de beta de 3.5, así la respuesta del filtro queda como se muestra en la figura 12.

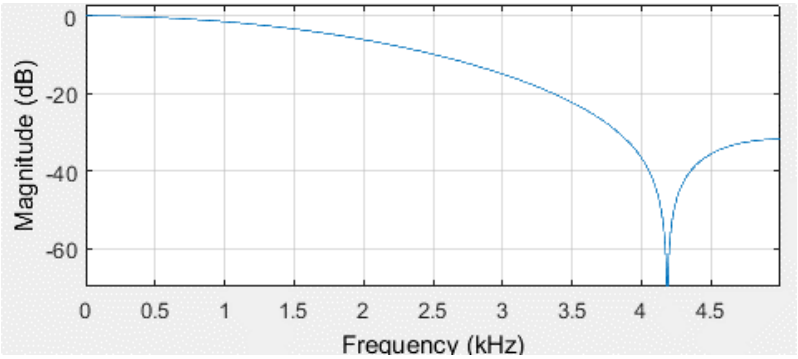

Figura 12 Filtro pasa bajas de orden 4.

La operación del filtrado se simuló utilizando simulink como se puede apreciar en la figura 13, utilizando el bloque llamado “chirp signal”, el cual proporciona una señal senoidal que realiza un barrido en un rango de frecuencias, como se puede observar en la figura 14.

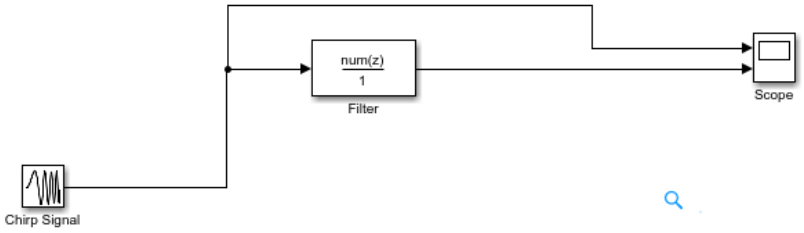

Figura 13 Prueba del filtro utilizando un barrido senoidal.

Como se puede observar, el filtro funciona correctamente en simulación. Para poder probarlo de manera física y por simplicidad se utilizó una señal de entrada de 1 kHz y otra de 5 kHz, de las cuales los resultados se pueden observar en las figuras 15 a 17. En la figura 15 se puede apreciar la señal de entrada a 1 kHz, la

cual se implementó en el FPGA. La figura 16 muestra la comparación de las señales de entrada y de salida en concordancia con la respuesta observada en la figura 14, ya que a una frecuencia de 1 kHz la atenuación de la señal es muy baja. En la figura 17 podemos observar el mismo fenómeno, pero ya que la frecuencia aumenta a 5 kHz, la atenuación es mucho mayor, por lo tanto podemos decir que el filtro funciona correctamente en su implementación en el FPGA.

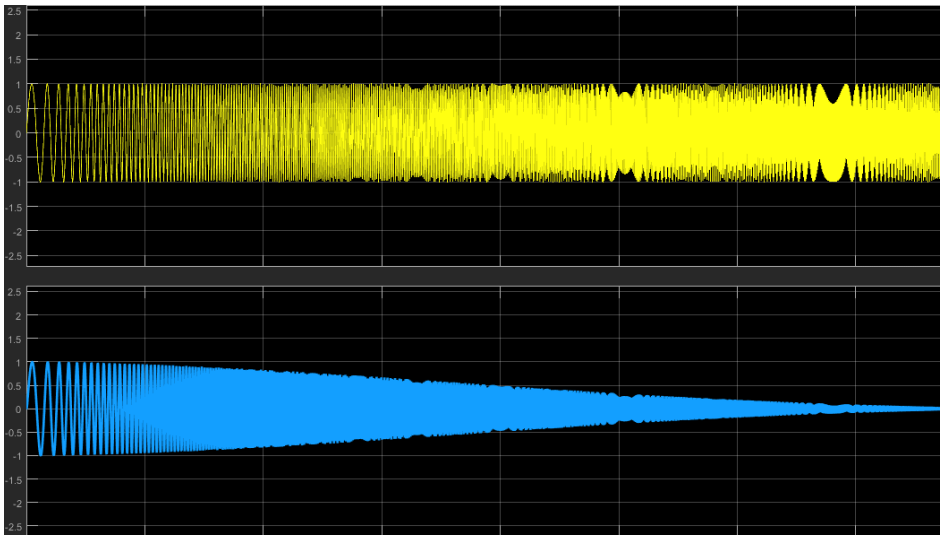

Figura 14 Prueba filtro a diferentes frecuencias (salida filtro azul, señal entrada amarillo).

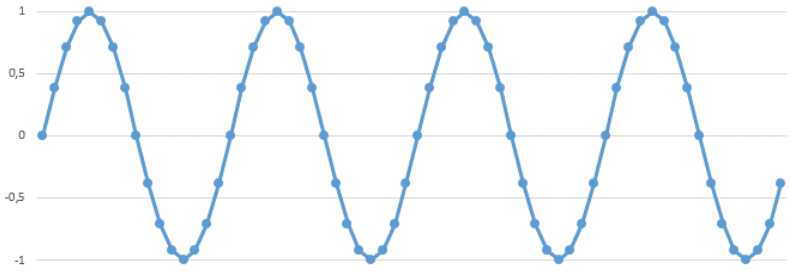

Figura 15 Señal de entrada utilizada (1 kHz).

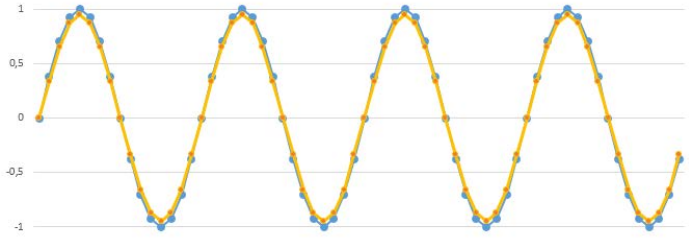

Figura 16 Comparación de señales de entrada y salida del filtro (1 kHz).

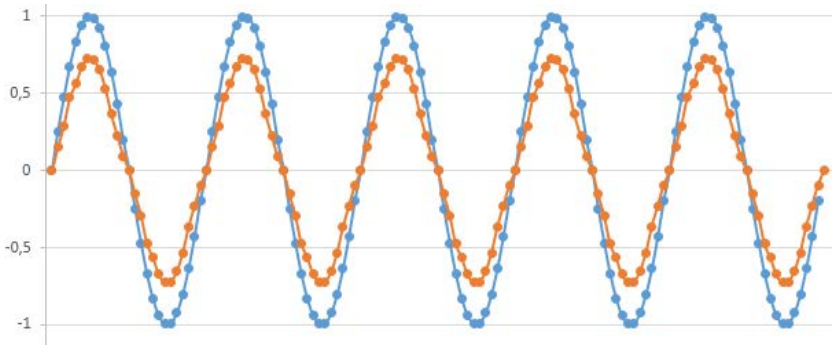

Figura 17 Comparación de señales de entrada y salida del filtro (5 kHz).

En cuanto a su velocidad se refiere, comparando las 2 estructuras empleadas para la aritmética distribuida, se obtienen los valores que se muestran en la tabla 2.

Tabla 2 Comparación de estructuras, velocidad.

Parámetro	Estructura paralelo	Estructura serie
Latencia	7 ciclos	30 ciclos
Cadencia	3 ciclos	26 ciclos
Frecuencia máxima de operación	84 MHz	158 MHz

En la tabla 3 se realiza la comparación de recursos del FPGA utilizados por cada una de las implementaciones.

Tabla 3 Comparación de estructuras, ocupación de recursos.

Recurso	E. Paralelo	E. Serie
Flip-flops	46	85
LUTs	251	135
LUTs para lógica	161	115
LUTs de interconexión	26	1
LUTs para memoria	64	19

En la tabla 4 se incluye un análisis de error efectuado para comprobar la correcta operación de la aritmética distribuida de manera estática, es decir con valores de entrada constantes.

Para generalizar, sea N el orden del filtro y D la longitud de palabra utilizada para los coeficientes del filtro, la tabla 5 muestra la cadencia para las dos arquitecturas.

Tabla 4 Análisis de error.

Multiplicación directa	Multiplicación con aritmética distribuida	Error relativo %
-0.806152344	-0.8053	-0.11
-0.945678711	-0.9459	0.02
-0.984436035	-0.9863	0.19
-0.937927246	-0.9352	-0.29
-0.790649414	-0.7929	0.28

Tabla 5 Cadencia para un filtro de orden N y longitud de palabra D (ciclos de reloj).

Paralelo	Serie
3	3 D + 2

4. Conclusiones

Se realizó la implementación hardware de un filtro digital de orden 4 utilizando aritmética distribuida para obtener los productos incluidos en la ecuación 1, la aritmética distribuida se implementó con dos estructuras. Revisando las dos estructuras mediante los diagramas de transición de estados, es posible advertir que la estructura en serie tiene una cadencia mayor, aunque puede operar a mayor frecuencia. Por otra parte, se advierte al revisar la ocupación de recursos del FPGA que la estructura en paralelo ocupa más recursos que la estructura en serie. En tanto que el consumo de potencia es mayor en la configuración en paralelo, la relación área-potencia es más favorable al utilizar la estructura en serie.

En la tabla 3 se aprecia que los errores son muy pequeños al utilizar la aritmética distribuida, y al graficar la señal de salida se obtiene la respuesta deseada del filtro.

5. Revisores

Revisor 1

Nombre:	Adolfo Rafael López Núñez
Institución:	TecNM / Instituto Tecnológico Superior de Irapuato
Cédula Profesional:	11088915
Área de conocimiento:	Electrónica de Potencia
Correo electrónico:	adolfo.lopez@itesi.edu.mx

Revisor 2

Nombre:	Leonel Estrada Rojo
Institución:	Instituto Tecnológico Superior del Sur de Guanajuato
Cédula Profesional:	6533069
Área de conocimiento:	Electrónica
Correo electrónico:	l.estrada@itsur.edu.mx

6. Bibliografía y Referencias

[1] Mohanty B. K., Meher P. K. & Patel S. K., LUT optimization for distributed arithmetic-based block least mean square adaptive filter, 2015.

[2] Hatai I., Chakrabarti I. & Banerjee S., An Efficient VLSI Architecture of a Reconfigurable Pulse-Shaping FIR Interpolation, 2015.

[3] Zhang X., Tu L., Chen D., Yuan Y., Huang K. & Wang Z., Parallel Distributed Arithmetic FIR Filter Design Based on 4:2 Compressors on Xilinx FPGAs, 2017.

[4] Avinash C.S. & Sahaya J., FPGA Implementation of Discrete Wavelet Transform using Distributed Arithmetic Architecture, 2015.

[5] Narsale R.M., Gawali D. & Kulkarni A., FPGA Based Design & Implementation of Low Power FIR Filter for ECG Signal Processing, 2014.

[6] Pitchaiah T. & Sridevi P. V., Low Power Area Efficient High Speed Implementation of LMS Adaptive Filter using Distributed Arithmetic, 2015.

[7] Murali L., Chitra D. & Manigandan T., Low Power Distributed Arithmetic Based Fir Filter, 2014.

[8] Park S.Y. & Meher P. K., Efficient FPGA and ASIC Realizations of a DA-Based Reconfigurable FIR Digital Filter, 2014.