ALGORITMO DE DEMODULACIÓN HETERODINA EN UN FPGA PARA DATOS DE INTERFEROGRAMAS

HETERODYNE DEMODULATION ALGORITHM IN AN FPGA FOR INTERFEROGRAM DATA

Jesús Enrique Valenzuela de la Cruz
Universidad Autónoma de Ciudad Juárez, México
al148715@alumnos.uacj.mx

Abimael Jiménez Pérez
Universidad Autónoma de Ciudad Juárez, México
abimael.jimenez@uacj.mx

Ángel Sauceda Carvajal
Universidad Autónoma de Ciudad Juárez, México
angel.sauceda@uacj.mx

José Antonio Muñoz Gómez
Universidad de Guadalajara, México
Jose.munoz@cucsur.udg.mx

Recepción: 7/noviembre/2021 **Aceptación:** 21/diciembre/2021

Resumen

En este trabajo se propone el diseño de un sistema digital embebido basado en bloques de hardware específico que implementa el algoritmo de demodulación heterodina. Con esto será posible demodular la señal obtenida de diferentes sensores ópticos interferométricos. El sistema embebido basado en FPGA se implementó satisfactoriamente en la tarjeta Nexys 4 DDR con un dispositivo FPGA Artix 7 y utilizando Vivado® Design Suite. Se lograron integrar distintos periféricos en un sistema embebido para realizar el procesamiento de interferogramas en 1-dimensión. Cada módulo de hardware desarrollado fue probado obteniendo respuestas similares a las obtenidas mediante la simulación del algoritmo de demodulación en Matlab.

Palabras Claves: Demodulación heterodina, FPGA, hardware, Interferogramas, sistema embebido.

Abstract

This paper proposes the design of an embedded digital system based on specific hardware blocks that implements the heterodyne demodulation algorithm. So, it will be possible to determine the physical variables of different interferometric optical sensors. The FPGA-based embedded system was successfully implemented on the Nexys 4 DDR board with an Artix-7 FPGA device and using Vivado® Design Suite. The portable embedded system integrates different peripherals for 1-dimensional interferogram processing. Each hardware module developed was verified, obtaining responses very similar to those obtained by the simulation of demodulation algorithm in Matlab.

Keywords**:** *Embedded system, FPGA, hardware, heterodyne demodulation, interferograms.*

1. Introducción

La tecnología de FPGAs está compuesta de un arreglo de bloques lógicos configurables, los cuales tienen mayor eficiencia debido a que no son de uso general ni configuración fija [Sundararajan, 2010]. Esto permite una gran versatilidad y flexibilidad en el diseño de sistemas digitales complejos, como la demodulación de señales de FM. Para implementar el sistema de demodulación de interferogramas en un FPGA, se revisó el estado del arte de todos aquellos trabajos que implementan algún sistema de interrogación electrónica de interferómetros y que han utilizado un FPGA como unidad de procesamiento. En la tabla 1 se muestra un resumen de los sistemas digitales reportados en la literatura que han sido diseñados para obtener la información de diferentes interferómetros. Se puede observar que únicamente el sistema propuesto por [Vera, 2011] es portable debido a que fue implementado en un FPGA.

Interferometria

El fenómeno de interferencia ha sido la piedra angular de una gran cantidad de aplicaciones en diferentes ámbitos científicos y tecnológicos, como la metrología óptica, la instrumentación científica y desde hace algunas décadas en biosensores

ópticos. Una de las principales ventajas de utilizar la luz como herramienta de medición estriba en que ésta puede ser considerada como una técnica no invasiva de medición; por ejemplo, en la estimación de objetos tridimensionales o de superficies. Además, es una técnica altamente precisa que está basada en la adquisición y procesamiento de señales asociadas a las ondas luminosas, tanto unidimensionales como bidimensionales [Bulut, 2005].

Dentro de las diversas técnicas de perfilometría se encuentra la interferometría basada en el fenómeno de interferencia de la luz, que se basa en el análisis de cambios de fase ocasionados por el fenómeno de interferencia entre dos ondas [Monroy, 2014]. La interferometría fue utilizada inicialmente para mediciones espaciales, y fue popularizada por Albert Michelson desde 1880 [Klempner, 2006]. En esta técnica, la señal obtenida de un interferómetro recibe el nombre de interferograma. El procesamiento de estos para extraer la información de interés se basa en la obtención, procesamiento e interpretación de una serie de imágenes o datos. Este proceso recibe el nombre de demodulación, haciendo una analogía con la nomenclatura utilizada en telecomunicaciones electrónicas [Malacara, 2005]. El instrumento utilizado para producir interferogramas es conocido como interferómetro y existen diversas configuraciones, tales como: Newton, Fizeau, Haldinger, Michelson, Twyman-Green, entre otras [Malacara, 2007].

Tabla 1 Comparación de distintos sistemas de interrogación implementados en FPGA.

Referencia	Sistema	FPGA	LUTs	DSP	ADC
[Cui, 2017]	Controlador electrónico de interrogación.	Stratix III	8867 (7%)	47896(6%)	N/A
[Vera, 2011]	Sensor inteligente basado en FPGA.	Spartan 3E XC3S1600E	N/A	N/A	14 bits
[Ehtesham, 2017]	Método de desenvolvimiento de fase directo.	Virtex 6 XC6VLx75T	5154 (7%)	3 (1%)	N/A
Este proyecto	Subsistema MMIO para el procesamiento de interferogramas.	Artix 7 (XC7A100 TCSG324)	-- de 95100	-- de 240	12 bits

Interferómetro

Un interferómetro consiste en una fuente de luz que emite haces de luz que se propagan en algún medio (espacio libre, fibra óptica o cualquier otro medio que sirva

de guía para las ondas luminosas) dependiendo de la aplicación. Al pasar el haz de luz por un divisor de haz, éste se separa en dos nuevos haces que toman diferentes trayectorias antes de volver a unirse.

En la figura 1 se observa que uno de los haces recorre una trayectoria hasta ser reflejado en una superficie de referencia. El camino recorrido por este haz se considera conocido. El otro haz, llamado haz de prueba, recorre una trayectoria distinta y es reflejado en otra superficie que puede estar en movimiento o en la que cambia alguna de sus propiedades físicas, como temperatura, índice de refracción, espesor, etc.

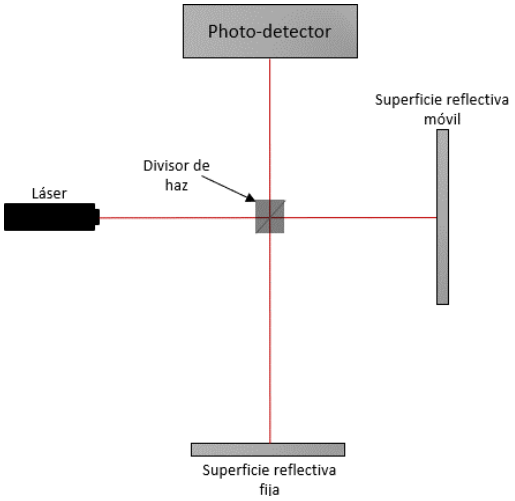


Figura 1 Arreglo esquemático de un interferómetro tipo Michelson.

Los cambios en la trayectoria del haz de prueba provocan un cambio en la fase relativa entre los dos haces, debido a la diferencia de camino óptico. Una vez reflejados ambos haces, estos vuelven a atravesar el divisor de haz y se combinan dando lugar al fenómeno de interferencia. Posteriormente son detectados por un fotodetector y convertidos en una imagen digital o en el caso de un interferómetro de fibra óptica, en una señal eléctrica dependiente del tiempo (t) [Connelly, 2002].

La señal eléctrica obtenida de un interferómetro de fibra óptica tipo Michelson, está descrita por la ecuación 1.

$$I(t) = A + V\cos\left(\frac{4\pi}{\lambda} OPD(t)\right) \quad (1)$$

Donde A y V son parámetros que dependen de valores del interferómetro como la intensidad de la fuente y, los coeficientes de acoplamiento y reflexión de la fibra óptica, entro otros. λ es la longitud de onda de la fuente de luz y la función $OPD(t)$, llamada fase óptica (del inglés Optical Path Difference) da cuenta de la diferencia de camino óptico inducida por la variación del mesurando. En ésta se encuentra la información de interés y el procedimiento para extraerla, recibe el nombre de demodulación. Se reconoce que la información de interés se encuentra dentro del argumento de un coseno y; por lo tanto, al cambiar su valor, cambia la frecuencia del coseno; como en los sistemas de comunicación de frecuencia modulada.

Demodulación

Aunque existen varios esquemas de demodulación o extracción de $OPD(t)$, en el caso de los sensores interferométricos, uno de los algoritmos más utilizados es el que se basa en la detección heterodina. Éste consiste en cambiar la frecuencia de uno de los haces de luz, introduciendo una modulación de frecuencia conocida, proporcional a $C\cos\omega_c t$. Este término equivale a la portadora de un sistema de comunicaciones de FM. Por lo tanto, la información es obtenida mediante una demodulación de FM [Rodríguez, 2015]. En este caso, la foto-intensidad detectada de la ecuación 1, puede ser expresada por la ecuación 2.

$$I(t) = A + V\cos\big(C\cos(\omega_0 t) + OPD(t)\big) \tag{2}$$

Donde C representa la amplitud máxima de la fase arbitraria y ω_0 es la frecuencia de esta modulación.

Existen varias formas de introducir experimentalmente la fase arbitraria $C\cos(\omega_0 t)$. Una de ellas es modulando la frecuencia del láser, lo cual sería una técnica sintético-heterodina [Connelly, 2002] o mediante un estirador de fibra óptica. Cualquiera que sea el caso, la ecuación 2 puede ser reescrita como en la ecuación 3.

$$I(t) = A + V\cos\big(C\cos(\omega_0 t) + \varphi(t)\big) \tag{3}$$

Donde $\varphi(t) = OPD(t) + \text{variaciones indeseables en la fase de interés}$, representa no solo la fase óptica debida al mesurando, sino también a contribuciones indeseables por perturbaciones ambientales como temperatura, presión, vibraciones, etc.

El algoritmo para recuperar la señal de interés $\varphi(t)$ se representa en la figura 2. Básicamente consiste en mezclar el interferograma obtenido experimentalmente con dos osciladores locales a frecuencias angulares ω_0 y $2\omega_0$. Posteriormente, se eliminan mediante filtraje las armónicas por encima de la máxima frecuencia de interés, obteniéndose las ecuaciones 4 y 5.

$$S_1(t) = AVJ_1(C)\sin\big(\varphi(t)\big) \tag{4}$$

$$S_2(t) = AVJ_2(C)\cos\big(\varphi(t)\big) \tag{5}$$

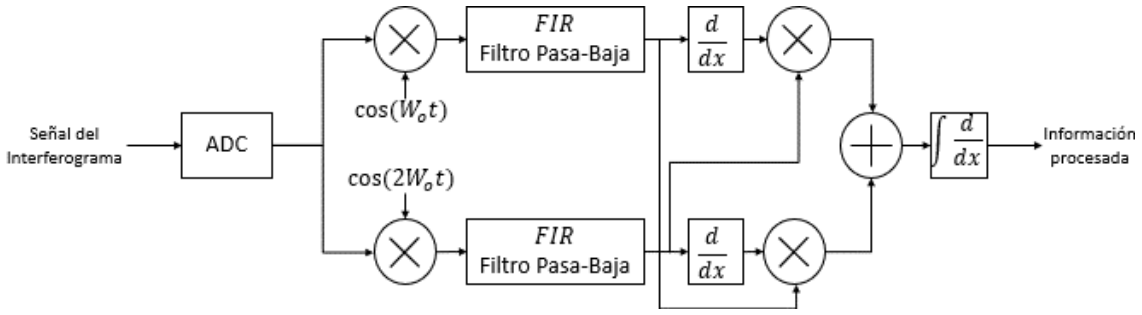


Figura 2 Diagrama de la implementación digital de demodulación heterodina.

Para las trayectorias superior e inferior, respectivamente, ver figura 2. $J_1(C)$ y $J_2(C)$ representan los valores de las funciones de Bessel evaluadas en C de primera clase de orden 1 y 2, respectivamente. El valor de C se escoge de tal manera que $J_1(C) = J_2(C)$; por lo tanto, el primer valor de C que satisface esta condición es $C \cong 2.37\ rad$. Posteriormente ambas señales $S_1(t)$ y $S_2(t)$, se derivan respecto a t, obteniendo las ecuaciones 6 y 7.

$$S_3(t) = AVJ_1(C)\dot{\varphi}(t)\cos\big(\varphi(t)\big) \tag{6}$$

$$S_4(t) = -AVJ_2(C)\dot{\varphi}(t)\sin\big(\varphi(t)\big) \tag{7}$$

Donde $\dot{\varphi}(t) = \frac{d\varphi}{dt}$. Al multiplicar $S_1(t)$ por $S_4(t)$ y $S_2(t)$ por $S_3(t)$ se obtienen las ecuaciones 8 y 9.

$$S_5(t) = -(AV)^2 J_1(C)J_2(C)\dot{\varphi}(t)\sin^2\big(\varphi(t)\big) \tag{8}$$

$$S_6(t) = (AV)^2 J_1(C)J_2(C)\dot{\varphi}(t)\cos^2\big(\varphi(t)\big) \tag{9}$$

Por último, sustrayendo $S_5(t)$ de $S_6(t)$, tenemos la ecuación 10.

$$S_0(t) = AV^2 J_1(C) J_2(C) \dot{\varphi}(t) \tag{10}$$

La ecuación 10 contiene la derivada de la señal de interés; por lo tanto, al integrar respecto a t se recupera la información de interés.

2. Métodos

Modelo de hardware del sistema

El sistema más sencillo para organizar los periféricos es la memoria mapeada de entradas y salidas (MMIO, Memory Mapped Input/Output), el cual se muestra en la figura 3. En este sistema se reserva un espacio en la memoria de datos para que al momento de acceder a ese espacio de memoria; en realidad se estará accediendo a un periférico. Con este subsistema MMIO, un microcontrolador puede activar un periférico específico a través de una dirección [Li, 2003], [Protopapas, 1988], [Vahid, 2010].

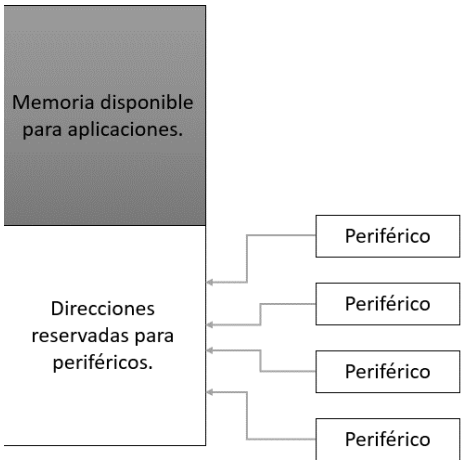


Figura 3 Memoria mapeada de entradas y salidas.

El subsistema MMIO de periféricos está basado en la arquitectura propuesta en [Chu, 2017], el cual es un sistema que decodifica los datos de un registro de función específica y selecciona el módulo de hardware que se utilizará. El sistema decodifica una señal de entrada (proveniente del microcontrolador) para seleccionar y habilitar uno de los módulos de hardware y, además dirige las entradas y salidas

a ese módulo para que realice una función específica. El subsistema MMIO utiliza las siguientes señales para comunicarse:

- Una señal de 32 bits para leer los datos que entran a un módulo de hardware.
- Una señal de 32 bits para escribir los datos que salen de un módulo de hardware.
- Una señal de 11 bits con la cual se activa el módulo de hardware.
- Un bit que habilita la escritura.
- Un bit que habilita la lectura.
- Un bit que habilita todo el subsistema MMIO.

Modelo de software del sistema

Desde la perspectiva del software, el microcontrolador simplemente escribirá o leerá información en un registro sin conocer que en realidad está interactuando con un periférico. Es decir, la interacción del microcontrolador con los módulos del subsistema MMIO será transparente desde la perspectiva del software. El software puede interactuar directamente con los periféricos y hacer que estos funcionen correctamente. En la figura 4 se muestran las capas del sistema embebido en un FPGA. El modelo de software contiene tres capas [Chu, 2017]:

- Especificación de plataforma de hardware: Es la capa más cercana al hardware. En esta capa se describe la implementación de hardware mediante lenguaje VHDL. En esta capa se encuentran los módulos de hardware (microcontrolador Microblaze y subsistema MMIO) [Chu, 2017].
- Paquete de soporte del dispositivo: Es la capa intermedia en la cual se encuentran los drivers y las secuencias de inicio del hardware, basado en la información proveída por el archivo de *especificación de plataforma de hardware*. Este archivo contiene la descripción de hardware del sistema. Del mismo modo, es en esta capa en donde el módulo de hardware y su driver interactúan para transferir datos de una capa a otra.
- Aplicación: Es la capa superior en la cual se encuentra la aplicación y la cual utiliza los recursos proveídos por la capa de *paquete de soporte del dispositivo* [Chu, 2017]. Normalmente, la capa de aplicación se utiliza para el manejo de

datos y su procesamiento. Como en esta arquitectura el procesamiento será realizado mediante módulos de hardware, la capa de aplicación se encargará únicamente del manejo y flujo de datos.

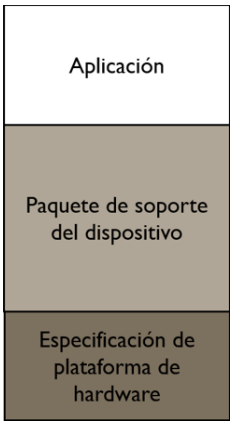


Figura 4 Modelo de software de un sistema embebido implementado en un FPGA.

Implementación del algoritmo de demodulación

La implementación del algoritmo de demodulación heterodino requiere de varias operaciones de un alto consumo de recursos de hardware. En este proyecto las operaciones son encapsuladas para su implementación en el subsistema MMIO (por parte del hardware) y en un objeto de C++ (por parte del software). La arquitectura propuesta para el sistema embebido se enfoca en que las funciones o implementaciones deben ser independientes y encapsulables. Para esto, la ejecución del módulo requiere de dos implementaciones (hardware y software).

Por la parte de hardware, la encapsulación y manejo de datos es proveída por el subsistema MMIO el cual se encarga de controlar qué datos llegan a los módulos y cuándo deberán ser activados. Por parte del software se utiliza el lenguaje de programación orientado a objetos, el cual define que cada objeto es una identidad distinta e independiente. En la figura 5 se observa cómo se relacionan entre si todos los elementos del sistema embebido.

En la figura 6 se observa que cada módulo encapsulado en el sistema consta de dos partes. La primera (hardware) es una descripción en VHDL del procesamiento que realizará el módulo.

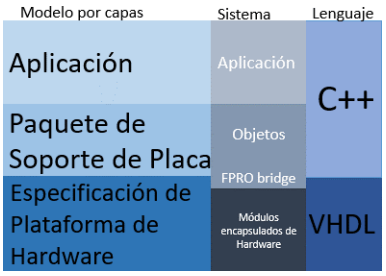


Figura 5 Implementación del sistema representado por capas.

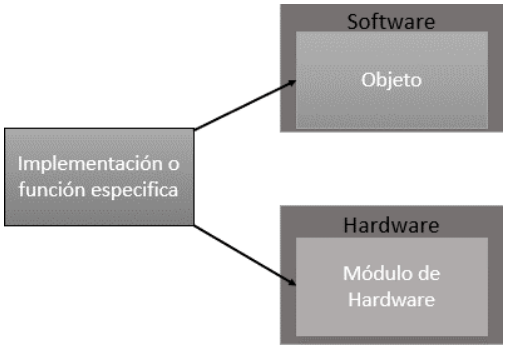


Figura 6 Implementación de una funcionalidad del sistema embebido.

Esta descripción contiene la lógica interna de procesamiento, un módulo previamente diseñado y encapsulado en una interfaz de comunicación con el exterior. La segunda (software), por parte del microcontrolador, el objeto del módulo contiene la descripción del objeto que controlará, los métodos utilizados por el objeto, la dirección referente al módulo y tipos de datos específicos.

Máquina de estados

Los módulos de filtro, derivada e integral requieren conservar el valor previo de entrada. En la figura 7 se muestra la máquina de estados utilizada. Ésta es requerida para evitar la pérdida del dato anterior por un valor de entrada no actualizado.

Los estados son los siguientes:

- Esperando nuevos datos. - En este estado se espera la operación de escritura por parte del microcontrolador y se guarda el dato previo introducido al módulo.
- Procesamiento. - En este estado se realiza el procesamiento específico del módulo.

- Esperando lectura de datos. - En este estado se espera la operación de lectura por parte del microcontrolador para obtener el dato procesado.

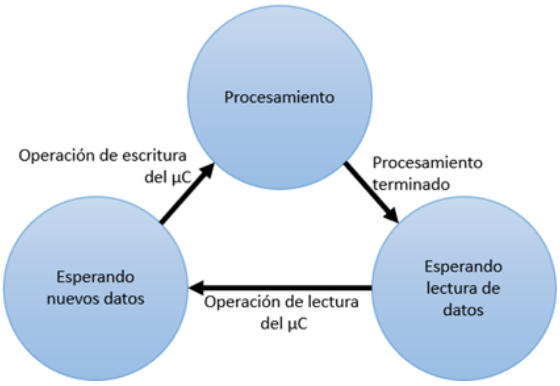


Figura 7 Máquina de estados para evitar la pérdida de datos.

Es importante mencionar que la máquina de estados no está diseñada para ser utilizada como un módulo de hardware encapsulado en el subsistema MMIO. Solo se utiliza para controlar el flujo de datos dentro de los módulos de hardware.

Look-up table

Un módulo LUT (del inglés Look-Up Table) es un arreglo de memoria que contiene los datos de salida de una función. En el sistema embebido propuesto las LUTs se utilizan para guardar los valores de una señal senoidal como se muestra en la figura 8, evitando todo el proceso que conlleva el cálculo de la función trigonométrica.

La implementación de una LUT toma un valor de entrada que indica un índice de la tabla de valores. La implementación en software de la clase en C++ contiene los siguientes atributos:

- Un constructor que requiere la dirección del módulo de VHDL.
- Un método de escritura, el cual requiere un índice de 32 bits como parámetro de entrada.
- Un método de lectura, el cual recibe el dato de 32 bits de la LUT como resultado.

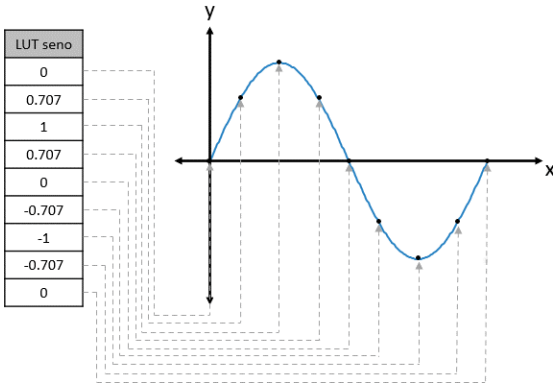


Figura 8 Gráfica de los valores guardados en una LUT.

Filtro FIR pasa-bajas

La implementación del filtro FIR se basa en utilizar entradas actuales y previas para obtener una salida. Su expresión en el dominio del tiempo está definida por la ecuación 11.

$$salida[n] = \sum_{i=0}^{N-1} b_i x[n-i] = b_0 x[n] + b_1 x[n-1] + \cdots + b_i x[n-i] \qquad (11)$$

Donde b_i representa los pesos o coeficientes del filtro, $x[n-i]$ denota los valores de entrada y N representa el orden del filtro. La ecuación 11 puede ser reemplazada por la implementación del diagrama de la figura 9 en donde los coeficientes definen el funcionamiento del filtro.

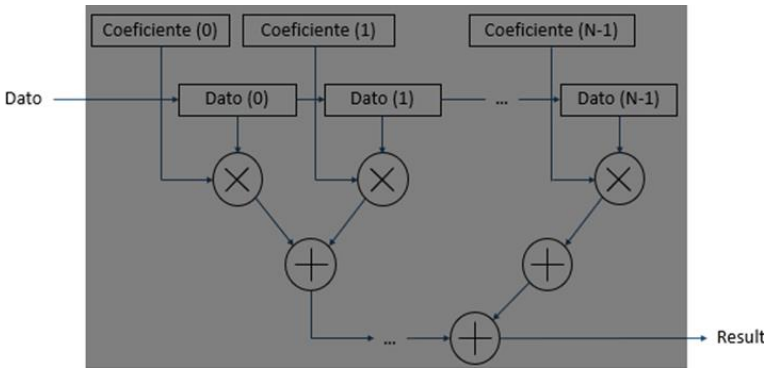


Figura 9 Implementación del filtro FIR.

Es importante señalar que en esta primera etapa del desarrollo del algoritmo heterodino en hardware, la implementación del filtro se realizó únicamente en

software. Es conocido que los tiempos de procesamiento no serán óptimos, pero nos permitirá validar el comportamiento del algoritmo heterodino en su totalidad. Actualmente, se trabaja en la implementación del filtro FIR en hardware.

Derivador

El módulo derivador se basa en el método de la secante, el cual obtiene la recta tangente a un punto mediante la pendiente entre dos puntos cercanos. De modo que, mientras más cercanos sean los puntos entre sí, mayor será la precisión como se observa en la figura 10.

Podemos definir la pendiente de la señal de la figura 10 como en la ecuación 12.

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{12}$$

Donde:

- y_2 representa el valor de entrada.
- x_2 representa tiempo en el que se realizó el muestreo.
- y_1 representa el valor previo de entrada.
- x_1 representa tiempo en el que se realizó el muestreo anterior.

La implementación del módulo en VHDL de la derivada toma el valor de entrada, le resta el valor previo y el resultado se divide entre el tiempo de muestreo. Esto implica que la precisión del módulo dependerá de la tasa de muestreo. Además, se utilizó la máquina de estados de la figura 7 para evitar la acumulación de datos en momentos de operación incorrectos. Respecto al software la implementación de la clase en C++ contiene los mismos atributos que se utilizaron en el módulo de la LUT.

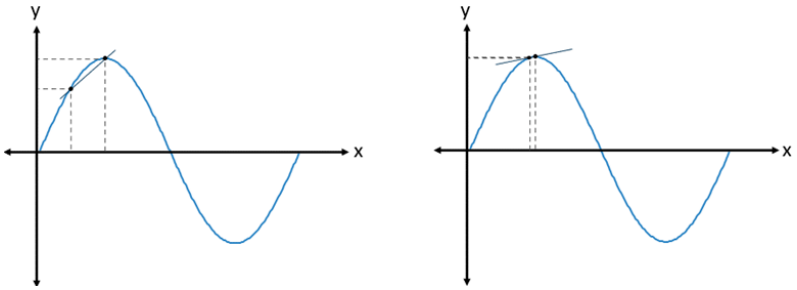


Figura 10 Comparación de la pendiente entre dos puntos a diferente distancia.

Integrador

El módulo integrador se implementó con la regla del trapecio, la cual aproxima, localmente, la curva por una línea recta. Tomando dos puntos se puede crear un trapecio para obtener una aproximación del área bajo la curva, como se observa en la figura 11. El área bajo la curva (a) se puede definir por la ecuación 13.

$$a = \frac{x_2 - x_1}{2}(y_1 + y_2) \tag{13}$$

Donde:

- y_2 representa el valor de entrada.
- x_2 representa tiempo en el que se realizó el muestreo.
- y_1 representa el valor previo de entrada.
- x_1 representa tiempo en el que se realizó el muestreo anterior.

De la misma manera que en el módulo de la derivada, la diferencia entre x_2 y x_1 corresponde al tiempo de muestreo y el numerador de la ecuación 13, puede ser reemplazado por el tiempo de muestreo, ecuación 14.

$$a = \frac{t_{muestreo}}{2}(y_1 + y_2) \tag{14}$$

Como se puede observar en la figura 11, la resolución del módulo de la integral también depende de la tasa de muestreo.

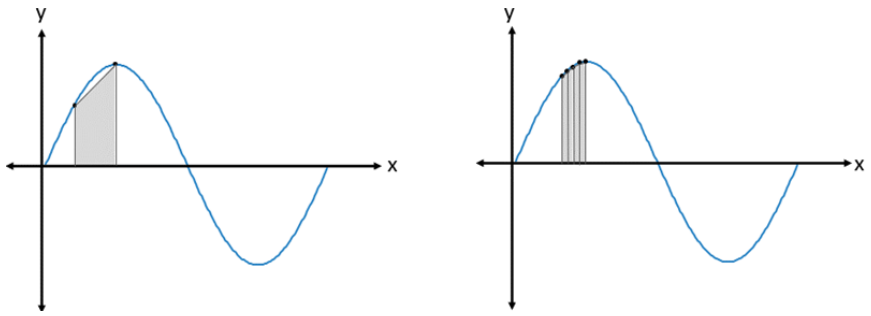


Figura 11 Comparación del área bajo la curva con trapecios de distinto tamaño.

La implementación del módulo en VHDL de la integral toma el valor de entrada, le suma el valor previo y el resultado es divido entre 2. Esta división es implementada mediante un desplazamiento a la izquierda. También se utilizó la máquina de

estados de la figura 7 para controlar la perdida de datos previos. Respecto al software, la implementación de la clase en C++ contiene los mismos atributos que se utilizaron en los módulos LUT y derivador.

Como se observa en la figura 2, es necesario implementar dos operaciones de derivada y dos de integración. Entonces, por cada operación (derivada o integral), se optó por utilizar dos módulos de VHDL y dos objetos con la finalidad de optimizar tiempos de ejecución. Es importante mencionar que ambos objetos son idénticos a excepción de la dirección utilizada.

3. Resultados

En esta sección se muestran los resultados obtenidos de la implementación del sistema embebido al introducir una señal de entrada. La señal de entrada al demodulador heterodino definida por la ecuación 2 con $A = V = 1$ se define en ecuaciones 15, 16 y 17.

$$I(t) = 1 + \cos(C\cos(2\pi f_0 t) + \varphi(t)) \tag{15}$$

$$C\cos(2\pi f_0 t) = 2.37\cos(2\pi 1000t) \tag{16}$$

$$\varphi(t) = 0.75\cos(2\pi 100t) \tag{17}$$

Se puede observar que la señal de prueba tiene una portadora de frecuencia f_0 =1 kHz y amplitud $C = 2.37\,rad$. De la misma manera, se ha supuesto una señal moduladora o señal de interés senoidal pura con amplitud de $0.75\,rad$ y frecuencia de 100 Hz. Por lo tanto, el ancho de banda del filtro FIR utilizado en el demodulador fue para una frecuencia de corte de 150 Hz.

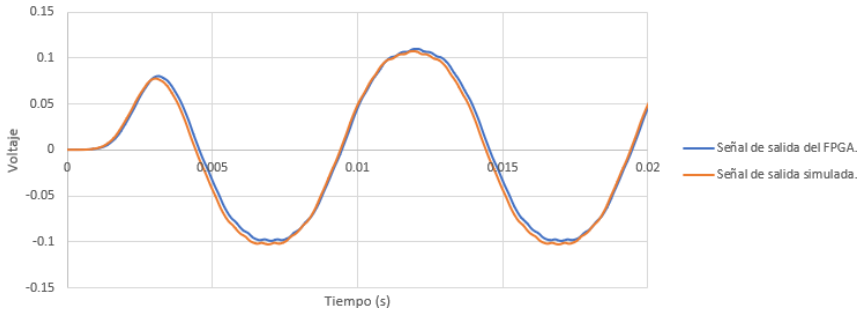
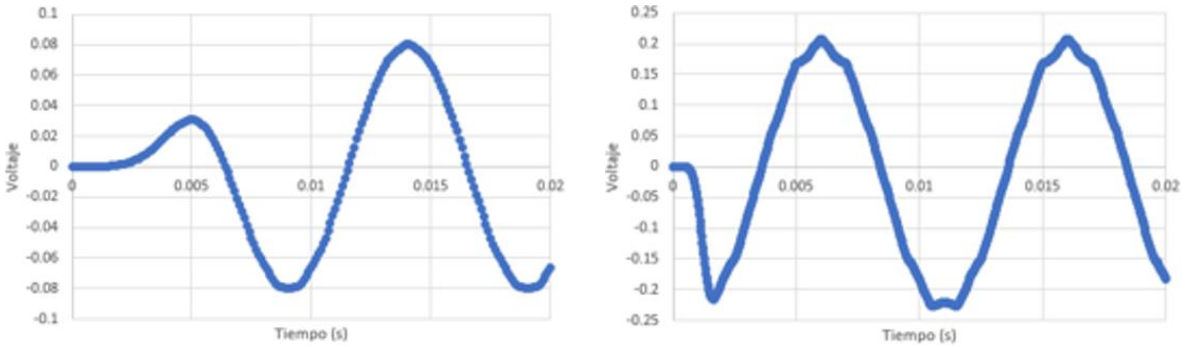


Figura 12 Señal de salida del sistema embebido implementado en el FPGA.

Una vez que el sistema propuesto realizó el procesamiento del interferograma de entrada, se obtuvo una señal como la que se muestra en la figura 12. Se puede observar que la señal de salida generada por el sistema embebido propuesto y la señal de salida de la simulación en Matlab son muy parecidas.

Muestreo a una frecuencia diferente de 20 kHz

Comúnmente se desea que la frecuencia de muestreo de una señal sea lo más grande posible. En el caso de este sistema embebido, es importante mantener la misma frecuencia de muestreo debido a que los Filtros FIR fueron diseñados para trabajar a una frecuencia de muestreo de 20 kHz. El utilizar una frecuencia de muestreo distinta a 20 kHz puede generar resultados erróneos. En la figura 13a se muestra una señal de salida, la cual fue creada a partir de una señal muestreada a 10 kHz. Por otra parte, en la figura 13b se muestra una señal de salida, la cual fue creada a partir de una señal muestreada a 40 kHz.

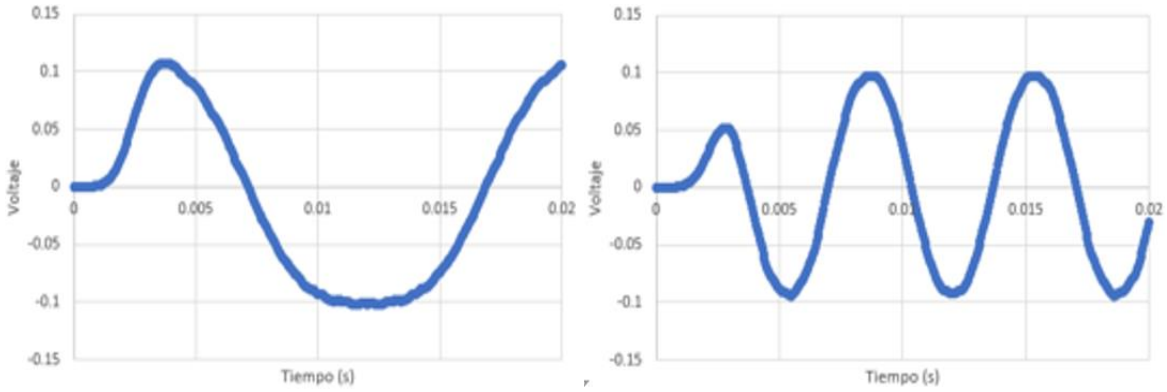


a) Frecuencia de muestreo de 10 kHz *b) Frecuencia de muestreo de 40 kHz*

Figura 13 Señales de salida del sistema utilizando diferentes frecuencias de muestreo.

Señal de entrada con una señal moduladora distinta a 100 Hz

La señal moduladora no siempre será de la misma frecuencia, por lo que es importante que el sistema sea capaz de procesar la señal de entrada con diferentes frecuencias en la señal moduladora. En la figura 14 se muestran dos señales de salida del sistema, las cuales tienen como entrada una señal moduladora de 50 Hz ver figura 14a y 150 Hz ver figura 14b. Como se puede observar, los cambios de frecuencia en la señal moduladora no tienen un efecto notable en la señal de salida.



a) Señal moduladora de 50 Hz *b) Señal moduladora de 150 Hz*

Figura 14 Señales de salida del sistema utilizando una señal moduladora anormal.

4. Discusión

Las principales limitantes del sistema propuesto son las siguientes: la frecuencia de muestreo del sistema se vio limitada por el requerimiento del Filtro FIR con una frecuencia de corte de 600 Hz debido a que al utilizar frecuencias de muestreo mayores a 20 kHz aumentan los recursos requeridos para su implementación. Por otro lado, la tablilla de desarrollo Nexys 4 DDR no cuenta con un DAC para convertir la señal procesada. También se tiene que considerar que el ADC del FPGA solo trabaja en un rango de 0 a 1 V para la configuración unipolar y de -0.5 V a 0.5 V para la configuración bipolar. Esto implicaría un pre-acondicionamiento en el caso de no tener la señal en el rango requerido, y a su vez un post-acondicionamiento para ser procesada por el sistema. Finalmente, se debe considerar la migración del sistema embebido a un FPGA con un ADC de mayor resolución con la finalidad de tener un mayor rango de valores significativos y por ende tener una mayor precisión. Es recomendable considerar el reemplazo del uso de la multiplicación (proveída por los bloques DSP del FPGA) por un módulo de multiplicación especializado para la reducción de tiempo en procesamiento y consumo de área. Por último, el sistema embebido está configurado de modo que solo puede procesar señales con una portadora de 1 kHz. Por lo que sería recomendable la implementación de un módulo que detecte la frecuencia de la señal de entrada y determine la frecuencia requerida de los osciladores locales. Esto implicaría el diseño de osciladores dinámicos y no estáticos como fueron implementados en este sistema.

5. Conclusiones

Se diseño satisfactoriamente una primera versión del sistema embebido portable que integra distintos periféricos para el procesamiento de interferogramas. En particular, se implementó digitalmente la demodulación heterodina en un FPGA. Esto conlleva también la codificación de los periféricos y su driver correspondiente, requerido en la etapa de procesamiento. Cada módulo fue verificado por separado, obteniendo respuestas muy similares a las obtenidas mediante su simulación numérica. La selección de los esquemas numéricos para realizar los módulos derivador e integrador se base en un esquema de primer orden. Esto con la finalidad de tener un tiempo de respuesta mínimo. Los esquemas utilizados fueron satisfactorios y la resolución obtenida no se vio comprometida. La señal analizada del interferograma fue introducida al sistema mediante una LUT. Los resultados obtenidos, al utilizar una señal con una resolución de 12 bits y muestreada a 20 kHz, son satisfactorios. Como trabajo futuro se contempla la implementación en harwdware del filtro y la adquisición de la señal con el módulo analógico-digital.

6. Bibliografía y Referencias

[1] Bulut, K., & Inci, M. N., (2005). Three-dimensional optical profilometry using a four-core optical fibre. Optics & Laser Technology, 37(6), 463-469.

[2] Chu, P. P., (2017). FPGA Prototyping by VHDL Examples: Xilinx MicroBlaze MCS SoC. John Wiley & Sons.

[3] Connelly, M. J., (2002). Digital synthetic-heterodyne interferometric demodulation. Journal of Optics A: Pure and Applied Optics, 4(6), S400.

[4] Cui, K., Li, S., Ren, Z., & Zhu, R., (2017). A highly compact and efficient interrogation controller based on FPGA for fiber-optic sensor array using interferometric TDM. IEEE Sensors Journal, 17(11), 3490-3496.

[5] Klempner, A. R., (2006). Development of a modular interferometric microscopy system for characterization of MEMS, Doctoral dissertation, MS Thesis, Worcester Polytechnic Institute, Worcester, MA.

[6] Li, Q., & Yao, C., (2003). Real-time concepts for embedded systems. CRC press.

[7] Ehtesham, A., Zabit, U., Bernal, O. D., Raja, G., & Bosch, T., (2017). Analysis and implementation of a direct phase unwrapping method for displacement measurement using self-mixing interferometry. IEEE Sensors Journal, 17(22), 7425-7432.

[8] Malacara, D., (2007). Optical shop testing, Vol. 59. John Wiley & Sons.

[9] Malacara, D., Servin, M., & Malacara, Z., (2005). Interferogram analysis for optical testing. CRC press.

[10] Monroy Ramirez, F., & Garcia Sucerquia, J., (2014). Monitoring micro-mechanical changes in electronic circuit boards with digital holographic interferometry. Optik, 125(9), 2113-2116.

[11] Protopapas, D. A., (1988). Microcomputer hardware design. Prentice-Hall, Inc.

[12] Rodríguez Antonio, A., (2015). Diseño y construcción de un circuito demodulador de fase óptica para su utilización en la extracción de señales de biosensores interferométricos. Licenciatura en Ingeniería Biomédica.

[13] Sundararajan, P., (2010). High performance computing using FPGAs, pp. 1-15. Technical Report: www.xilinx.com/support/documentation/whitepapers /wp 375 HPC Using FPGAs.pdf.

[14] Vahid, F., (2010). Digital design with RTL design, VHDL, and Verilog. John Wiley & Sons.

[15] Vera Salas, L. A., Moreno Tapia, S. V., Garcia Perez, A., Romero Troncoso, R. D. J., Osornio Rios, R. A., Serroukh, I., & Cabal Yepez, E., (2011). FPGA-based smart sensor for online displacement measurements using a heterodyne interferometer. Sensors, 11(8), 7710-7723.