IMPLEMENTACIÓN DE UN SISTEMA DE ANÁLISIS DE SEÑALES TIPO POLISCOPIO UTILIZANDO FPGA

Jhonatan Motta Gómez
Diego Fernando Reyes

Trabajo de Grado en Modalidad de Desarrollo

Monografía presentada para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Director
Ing. DR. Pablo Emilio Jojoa

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Línea de Desarrollo en Procesamiento Digital de Señales
Popayán, Diciembre de 2010

Tabla de contenido

Índice de Ilustraciones

Índice de Tablas

Lista de Acrónimos

ADC: Analogic-Digital Converter

ASCCI: Application Specific Integrated Circuit

ASSP: Application Specific Standard Products

CDMA2000: Code division multiple access

CLB: Configurable Logic Block

CORDIC: COordinate Rotation DIgital Computer

CPLD: Complex Programmable Logic Device

CRT: Cathode Ray Tube

DAC: Digital Analog Converter

DCM: Digital Clock Manager

DDR2: Double Data Rate Version 2

DFT: *Discrete Fourier Transform*

DSP: Digital Signal Processing

EDGE: Enhanced Data rates for GSM of Evolution

FFT: Fast Fourier Transform

FIET: Facultad de Ingeniería Electrónica y telecomunicaciones

FIFO: First In First Out

FIR Finite Impulse Response

FLP: Field Logic programable

FPGA: Field Programmable Gate Array, Arreglo de Compuertas Programables

GSM: Global System for Mobile Communications

HSDPA: High Speed Downlink Packet Access

IBM: International Business Machines

IEEE: Institute of Electrical and Electronic Engineering

IOB: IN/OUT Block

LCD: Liquid Crystal Display

LUT: Look Up Table, Tabla de verdad

MAC: Multiplicator – Acumulator

MATLAB: MATrix LABoratory

PC: Personal Computer

PDSP: Programmable Digital Signal Processor

PROM: Programmable Read-Only Memory

RAM: Ramdom Access Memory

RF: Radio Frecuencia

RISC: Reduced Instruction Set Computer

SDRAM: Synchronous Dynamic Random Access Memory

SMA: Subminiature versión A

SPI: Serial Peripheral Interface Bus

VGA: Video Graphics Array

WCDMA: Wideband Code Division Multiple Access

WLAN: Wireless Local Area Network

Introducción

Realizar medidas es una parte fundamental en la mayoría de actividades técnicas o científicas. Es evidente que representar cuantitativamente los fenómenos permite conocer mejor el objeto de estudio y esto depende en gran parte del proceso de medición que se lleve a cabo y del equipo que se utilice. En comunicaciones electrónicas, las señales de voltaje o corriente que se necesitan medir son magnitudes variantes en el tiempo que comúnmente se representan en el dominio de la frecuencia, mediante el uso de técnicas matemáticas como la transformada de Fourier, por lo cual las mediciones de este tipo de señales son un proceso complejo que requiere el estudio simultáneo tanto del voltaje como de la frecuencia. Para determinar la respuesta en frecuencia de los circuitos, se caracteriza la magnitud de voltaje contra el desplazamiento en frecuencia, ya que en esta curva se determinan las principales características de la red, como su intensidad relativa y ancho de banda. Realizar esta labor requiere el uso de varios equipos entre los que se encuentran, generadores, multímetros, osciloscopios, frecuencímetros.

El poliscopio es un instrumento de prueba que integra los dispositivos necesarios para realizar medidas simultáneas de voltaje contra frecuencia en un solo equipo y despliega una curva de estas dos variables en un rango determinado. Este equipo es de gran importancia para el estudio de circuitos y líneas de transmisión y también para las prácticas de laboratorio que se llevan a cabo a lo largo de la carrera Ingeniería Electrónica y Telecomunicaciones y carreras afines, puesto que sus múltiples aplicaciones permiten una mejor apropiación y verificación de conceptos.

La principal diferencia entre un poliscopio y un analizador de espectro es que el poliscopio genera la señal con la que se estimula el circuito de prueba y realiza una comparación de la señal de salida y la señal de entrada, mientras que un analizador de espectro solo muestra la composición espectral de una señal que se conecte a su entrada. Ambos equipos pueden utilizar un receptor superheterodino donde el oscilador local barre una gama de frecuencias y se afina automáticamente dentro de una gama de fija.

Considerando que el estado actual del poliscopio no es funcional, que es un equipo de mucha antigüedad, lo que hace difícil conseguir sus refacciones, y teniendo en cuenta que El Departamento de Telecomunicaciones de la FIET cuenta con la FPGA Spartan 3A Starter kit de la compañía Xilinx® cuya exploración ha sido mínima, se plantea como objetivo principal de este proyecto Implementar un sistema de análisis de señales tipo poliscopio utilizando FPGA. Los objetivos específicos definidos son los siguientes:

1. Caracterizar el poliscopio y los sistemas actuales que permiten visualizar curvas de voltaje-frecuencia.
2. Definir los requerimientos necesarios para la implementación del Sistema de Análisis de Señales de tipo poliscopio.
3. Definir la arquitectura modular básica para la implementación del Sistema de Análisis de Señales de poliscopio.

Dada la complejidad del poliscopio, su total implementación ha sido dividida en dos partes, la primera parte se encargara de la etapa de generación, adecuación y captura de las señales del poliscopio. La segunda parte consiste en la implementación del sistema que realiza el análisis de las señales generadas por el poliscopio. Cada una de estas etapas ha sido propuesta como trabajos de grado por separado. El presente trabajo de grado se centra en el desarrollo de la segunda etapa.

En este documento se presenta la descripción del proceso realizado para alcanzar los objetivos planteados. Además se examinan conceptos generales que se tuvieron en cuenta para el desarrollo del proyecto. El presente documento está organizado de la siguiente forma:

En el capítulo 1 se presenta la descripción del poliscopio, sus componentes y principales aplicaciones, así como otros sistemas para visualizar curvas de voltaje frecuencia, con lo que se pretende dar cumplimiento con el objetivo específico número uno. También se describe la estructura básica de una FPGA, su papel en procesamiento de señales y las características principales de la Spartan 3A. Además en este capítulo se mencionan conceptos generales que son importantes para el desarrollo del proyecto.

En el capítulo 2 se presenta la arquitectura modular propuesta para la implementación del sistema de análisis de señales de tipo poliscopio utilizando FPGA, se realiza la descripción de cada uno de los módulos funcionales y se muestran detalles de la implementación realizada. Con esto se busca dar cumplimiento a los dos objetivos específicos restantes.

En el capítulo 3 se realizan las pruebas que validan el prototipo del sistema de análisis de señales de tipo poliscopio implementado utilizando FPGA. Con esto se cumple el objetivo general propuesto para este proyecto.

Adicionalmente, en los anexos se incluye un manual para el uso del entorno ISE de Xilinx con el cual se programa la FPGA. También se incluye una descripción de la herramienta software desarrollada en Matlab para realizar aplicaciones de poliscopio.

Capítulo I Conceptos Generales.

En este capítulo se presenta el poliscopio como instrumento para análisis de señales, se muestra su importancia en el campo de las telecomunicaciones y se habla de sus componentes. Debido a que el proyecto es el sistema de análisis de señales tipo poliscopio utilizando FPGA[1], también se hará una breve descripción acerca de lo que es una FPGA, su diferencia con los CPLD y su importancia en el diseño digital. Por último se aborda el procesamiento digital de señales con FPGA, presentando las principales características de la FPGA Spartan 3A de Xilinx, se menciona la transformada rápida de Fourier y se explican los aspectos generales de la representación en punto flotante.

1.1 El poliscopio.

El poliscopio es un instrumento de prueba que combina los componentes necesarios para determinar medidas exactas de voltaje y frecuencia simultáneamente, permitiendo desplegar una gráfica de estas medidas en el rango de frecuencia de interés.

La importancia del poliscopio como equipo de prueba en el campo de las telecomunicaciones radica en que gracias a su funcionalidad es posible observar cómo se comporta una red cuando es afectada por una señal específica, es decir, en el poliscopio es posible ver la distribución de voltaje a lo largo de la frecuencia que se produce cuando se conecta un circuito de prueba en medio de sus terminales [1]. Con el poliscopio es viable realizar medidas en una fracción del tiempo que se necesitaría para hacer medidas del tipo punto a punto y los resultados tienen una precisión muy alta que requeriría gran esfuerzo utilizando otro medios.

Las ventajas del despliegue simultáneo de voltaje contra frecuencia son el ahorro de tiempo y claridad de la exposición de la información. Además, frecuentemente hay perdida de información cuando se realizan medidas del tipo punto a punto, generalmente por la presencia de discontinuidades indeseables en las curvas de respuesta de circuitos o sistemas o variaciones espontáneas causadas por fallos en contactos y perturbaciones introducidas por señales no llevadas a tierra correctamente [2]. El equipo necesario para tomar las lecturas simultaneas de voltaje frecuencia que realiza el poliscopio incluye varios generadores de señal para cubrir el rango de frecuencia deseado, un atenuador calibrado, un voltímetro con circuito para toma de medida antes del circuito de prueba, un voltímetro con circuito para entrega de medida después del circuito de prueba y una punta para prueba de RF[2] [2]. En la figura 1.1 se muestra un diagrama de interconexión del equipo necesario para realizar el tipo de medidas que realiza el poliscopio. El bloque de presentación de la medida que se muestra en la Figura 1.1 se refiere a un osciloscopio convencional. El bloque denominado toma de medida consiste en un circuito de

[1] Field Programable Gate Array, Arreglo de Compuertas Lógicas Programable.
[2] Diodo de Radio Frecuencia.

adaptación de impedancias para que las terminales del poliscopio sean transparentes al circuito de prueba.

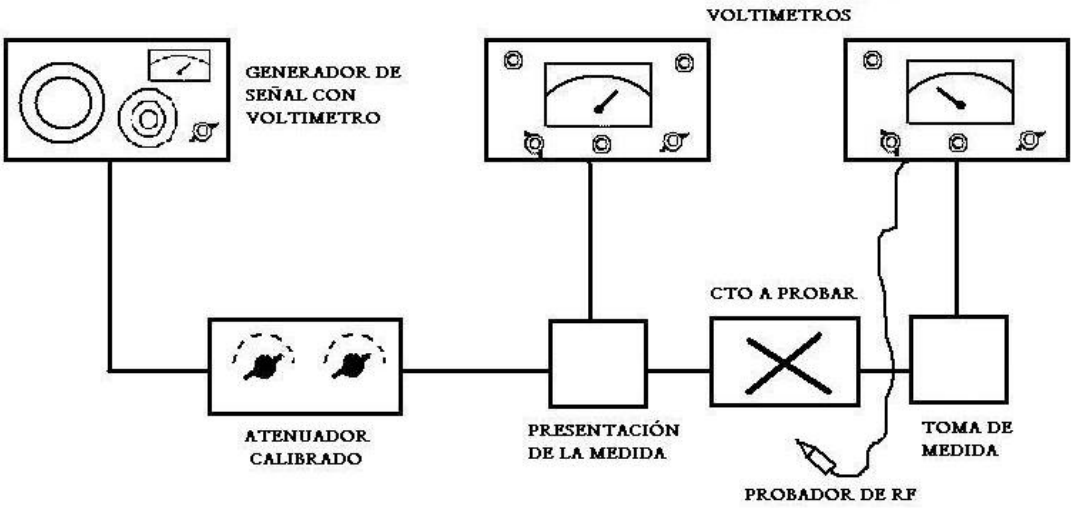

Figura 1.1. Diagrama de interconexión para medidas tipo poliscopio.

El Laboratorio de Telecomunicaciones de la FIET[3] cuenta con un poliscopio de barrido o tipo SWOB[4], referencia BN4244 de fabricación alemana y producido por la empresa Rohde & Schwarz hace unos 50 años [1]. Trabaja en un rango de frecuencia que va desde 300KHz hasta 1GHz y es un equipo con un buen desempeño en medidas de banda ancha [3]. En la Figura 1.2 se muestra un dibujo de un poliscopio de tipo SWOB, allí podemos ver lo sencillo de la conexión del circuito a probar y los controles principales del poliscopio.

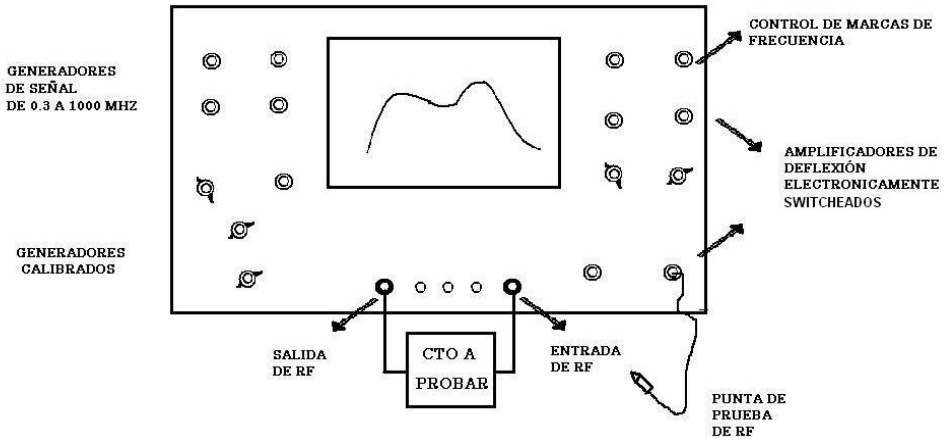

Figura 1.2. Poliscopio tipo Swob

[3] Facultad De Electrónica Y Telecomunicaciones.
[4] SWOB, del alemán se traduce como barrido

1.2 Componentes de un poliscopio BN4244.

El poliscopio BN4244 Incorpora varios generadores de señal con un circuito de control automático para mantener constante la fuerza electromotriz y así proporcionar una diferencia de potencial adecuada, atenuadores calibrados que sirven para controlar el voltaje de salida y atenuarlo por debajo de 70dB, un voltímetro a tubos con circuito para toma de medida y un voltímetro a tubos con circuito para entrega de medida. Para el despliegue de la curva voltaje contra frecuencia el poliscopio incluye una pantalla tipo CRT[5] [1].

La estructura del poliscopio funcionalmente se puede dividir en dos partes, primero, una unidad generadora de señales de barrido de frecuencia WG-4[1] que trabaja en un rango de frecuencia de 300 KHz a 1 GHz y puede desplegar frecuencias en dos o cuatro terminales según el tipo de red que se quiera conectar. Segundo, una unidad amplificadora de despliegue que adecua la señal antes de ser entregada al sistema de visualización. Además la unidad de despliegue SG-1, que consiste en una pantalla de tubo de rayos catódicos CRT de 28 cm. con mayor sensibilidad que la de un osciloscopio convencional [2]. La unidad de despliegue posee dos entradas separadas de RF haciendo posible la conexión de dos señales detectadas para su medición, las cuales pueden ser desplegadas alternativamente en la pantalla de CRT, por medio de switch electrónico.

1.3 Aplicaciones del poliscopio.

A continuación se presentara una breve descripción de las principales aplicaciones del poliscopio, la forma detallada de cómo realizar estas aplicaciones puede encontrarse en la documentación del poliscopio [1]

1.3.1 Medida de circuitos resonantes simples.

En un circuito resonante puede determinarse frecuencia de resonancia y ancho de banda. El circuito resonante simple puede ser el acople entre dos etapas amplificadoras y pueden realizarse las medidas directamente sobre la totalidad del circuito y particularmente en los puntos de interés para el caso. Como ya se ha dicho es posible medir la frecuencia de resonancia y el ancho de banda, adicionalmente se puede determinar el factor Q del circuito sintonizado a partir del ancho de banda.

1.3.2 Medidas en filtros.

Se puede determinar coeficiente de acoplamiento y ancho de banda para medidas y alineamiento de filtros pasa banda, pasa bajo y pasa alto, existe la posibilidad de presentar simultáneamente dos resultados que trascienden en el estudio de filtros pasa banda: el ancho de banda y el factor de acople de un filtro pasa banda.
En un sistema de amplificación multietapa es posible realizar el alineamiento de entrada hacia la salida tomando cada etapa en forma individual.

[5] Tubo de Rayos Catódicos.

1.3.3 Limitadores y discriminadores.

Limitadores: Es posible conocer el rango de acción de un limitador por medio de la visualización de la curva de respuesta del voltaje de salida el circuito al cual se le inyecta la señal desde el poliscopio la cual se puede atenuar mediante un atenuador calibrado.

Discriminadores: En un discriminador se puede observar su curva de respuesta y ajustarla correctamente a la frecuencia central.

1.3.4 Alineamiento de filtros.

Elementos como filtros pasabajo, pasaalto o pasabanda pueden medirse y alinearse con el poliscopio, adicionalmente puede observarse la respuesta de salida, el grado de desadaptación de un filtro a un cable de impedancia característica conocida o desconocida, observando el valor máximo de voltaje de salida en las formas de onda de la curva de respuesta[6].

1.3.5 Medida de la ganancia de una etapa amplificadora.

Se puede obtener el valor de ganancia de una etapa amplificadora, haciendo uso del poliscopio y en especial de los atenuadores de señal que este provee, obteniendo una medida con alto grado de precisión.

1.4 Otros Instrumentos para medidas de voltaje contra frecuencia.

Los actuales sistemas de medida y de análisis de señal para curvas de voltaje frecuencia no se basan en el funcionamiento del poliscopio tipo SWOB, no presentan como tal una curva voltaje contra frecuencia como lo hace el poliscopio sino que son aplicaciones de procesamiento digital que grafican muchos parámetros incluida la magnitud de la transformada de Fourier que ejecutan sobre una señal de entrada. Estos equipos no generan su propia señal de prueba sino que simplemente realizan el análisis de la señal que se aplica a sus terminales de entrada. El costo de un equipo de este tipo depende del rango de frecuencia operación. Los equipos de mejores características y más conocidos por su calidad y trayectoria son de la marca Rohde & Schwarz, que es la misma empresa que fabrico el poliscopio tipo SWOB. Estos equipos son analizadores de señales que cubren un ancho de banda desde unos pocos Hertz hasta los 67 GHZ para equipos de gama alta de la serie R&S®FSU. Estos equipos incorporan aplicaciones de filtrado y aplicaciónes de firmware para GSM / EDGE, Bluetooth, WCDMA / HSDPA / TD-SCDMA, CDMA2000/, WLAN. El costo de equipos de este tipo va desde 13600 Euros para los de gama baja, hasta 53000 Euros para los de gama alta.

[6] El poliscopio tipo SWOB es capaz de realizar medidas únicamente sobre filtro pasivos.

1.5 Conceptos básicos sobre las FPGA.

Una FPGA (Field Programmable Gate Array) es un conjunto de circuitos integrados digitales que contienen bloques lógicos configurables e interconexiones también configurables entre dichos bloques. Este conjunto de dispositivos que conforman una FPGA se programa, interconectando los bloques lógicos de manera conveniente para realizar una gran variedad de tareas [4].

Las FPGAs fueron inventadas en 1984 por Ross Freeman y Bernard Vonderschmitt, co-fundadores de Xilinx [5] y surgen como evolución de los CPLD (Complex Programmable Logic Device). La diferencia más importante entre una FPGA y un CPLD radica en la densidad de elementos lógicos programables que tiene cada dispositivo, en un CPLD pueden ser del orden de decenas de miles mientras que en una FPGA se puede tener hasta cientos de miles de bloques configurables [4].

Los principales componentes de una FPGA son los **CLB** (Configurable logic block, Bloque Lógico Configurable), los **IOB** (In/out blocks, Bloques de Entrada/Salida), multiplexores y switches programables y las interconexiones entre ellos [4]. En la figura 1.3 se presenta la estructura básica de una FPGA y se muestra la disposición de los CLB y los IOB.

Un CLB es la unidad básica de una FPGA, es un conjunto de matrices de conmutación configurable, multiplexores y flip-flops. La matriz de conmutación es adaptable a múltiples configuraciones y realiza diversas funciones incluido el almacenamiento de datos [5].

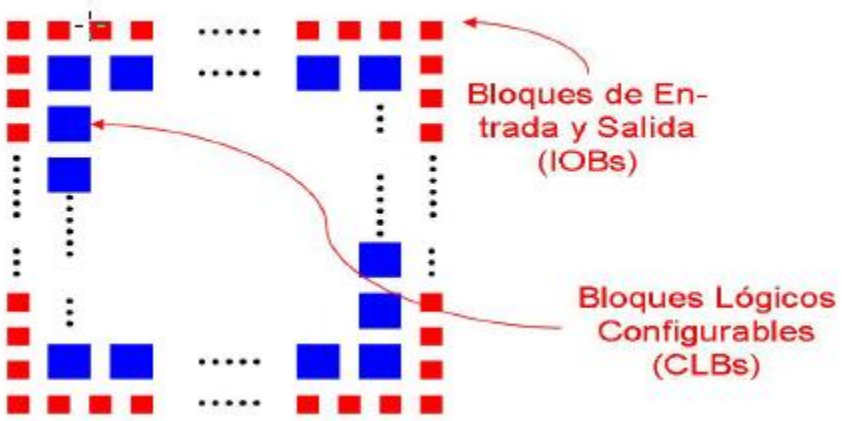

Figura.1.3. Estructura básica de una FPGA.

Los IOB son conjuntos de bancos de entradas y/o salidas; cada banco es capaz de soportar diferentes configuraciones y controlar el flujo de datos entre los pines de entrada y salida y la lógica interna del dispositivo, también acepta configuración del tipo bidireccional [5].

Las interconexiones entre los bloques lógicos pueden ser interconexiones cercanas “entre vecinos”[7] o interconexiones segmentadas. Las interconexiones cercanas permiten que bloques lógicos se comuniquen con otros bloques lógicos adyacentes, mientras que las interconexiones segmentadas añaden mayor complejidad al comunicar a los bloques lógicos más lejanos creando rutas y pistas por medio de switches y multiplexores programables a través de interconexiones cercanas [5].

Además de estos elementos fundamentales, las FPGA pueden integrar otros elementos que dependen del fabricante y del modelo del integrado. Estos recursos van desde memorias, multiplicadores, hasta arreglos lógicos más complejos como lo son los administradores de reloj DCM (Digital Clock Manager). Todos estos recursos añadidos permiten realizar diversas tareas en los distintos diseños basados en FPGA, y al estar dedicados para funciones específicas permiten realizar aplicaciones con alta eficiencia. Esto se traduce en un mejor aprovechamiento de los recursos de uso global, especialmente en la minimización de la cantidad de CLB utilizados.

La importancia de las FPGA en el diseño digital radica en la versatilidad de su uso, en lo rápido que puede resultar un desarrollo haciendo uso de ellas, en el bajo costo del desarrollo, en lo relativamente sencillo que resulta hacer cambios en el prototipo, así que es muy común que primero se evalué un diseño digital sobre una FPGA, se observe su comportamiento, se depure y se realicen las consideraciones hardwae y software respectivas antes de crear ese diseño sobre un circuito integrado de tipo ASCCI (Circuito Integrado para Aplicaciones Específicas) o ASSP (Producto de Estándar Específico de Aplicación) [6].

1.5.1 Las FPGA y el procesamiento digital de señales.

El procesamiento digital de señales es un campo de la ingeniería que se encarga de estudiar y analizar señales del tipo discreto, para ello hace uso de diversas herramientas matemáticas, entre las cuales, la más importante es la transformada discreta de Fourier (DFT), que convierte una señal del dominio del tiempo al dominio de la frecuencia, análisis que resulta muy útil en determinadas aplicaciones como eliminación de ruido y aplicaciones de filtrado [9].

El procesamiento digital de señales ha madurado como tecnología y como campo de estudio en los últimos tiempos, y en la actualidad, ha remplazado al tradicional procesamiento analógico de señales, a tal punto que es muy común encontrar aplicaciones de DSP (Digital Signal Processing) en diversos campos, que van desde las comunicaciones, audio, voz, hasta sistemas de tecnología biomédica [7].

Existen dos hitos muy importantes en el procesamiento digital de señales con FPGA. El primero, es la divulgación por parte de Cooley and Tuckey (1965) de un algoritmo eficiente para calcular una Transformada Discreta de Fourier (DFT), el cual ha facilitado

[7] Traducción de Neighbors en la referencia citada.

en gran medida el desarrollo de aplicaciones de lógica digital. El segundo hito importante es el desarrollo de Procesadores Digitales de señales, que en la actualidad se incorporan en las tarjetas de desarrollo que acompañan las FPGA modernas. [8]

En la práctica el procesamiento digital de señal se lleva a cabo mediante circuitos integrados, microprocesadores y computadores. La lógica digital hace posible efectuar operaciones con las muestras que representan una señal y así realizar el tratamiento necesario para los diferentes análisis sobre la esta. Dichas operaciones involucran algoritmos, sistemas de ordenamiento, sumas, multiplicaciones, divisiones y demás sobre vectores binarios en formato de punto fijo y en formato IEEE 754[8]. En la mayoría de los casos este procesamiento se realiza utilizando DSP (Digital Signal Processor) aunque en los últimos tiempos se viene utilizando arquitecturas sobre FPGA para desarrollar estos procesos.

Los procesadores programables digitales de señales de propósito general PDSP (Programmable Digital Signal Processor) han tenido un alto desarrollo durante los últimos 20 años, se basan en un conjunto reducido de instrucciones llamado RISC (Reduced Instruction Set Computer) y cuentan con un modelo que hace uso de al menos un vector rápido de multiplicadores, ya sea de 16 x 16 bits o de 24 x 24 bits en punto fijo o 32 bits en punto flotante, el cual tiene un acumulador con un ancho de palabra extendido. Los PDSP tienen la ventaja de que la mayoría de algoritmos de procesamiento digital están diseñados para actuar sobre multiplicadores y acumuladores (MAC, Multiplicator - Acumulator), sin embargo, al usar una arquitectura de conexión multinivel, la velocidad de los MAC está limitada por la velocidad del arreglo multiplicador. Aunque superado este inconveniente los PDSP tienen la ventaja sobre las FPGA de tener un costo mucho menor [8].

Por otro lado, las FPGA como dispositivos programables permiten adecuar sus bloques lógicos para ejecutar las instrucciones necesarias para el tratamiento digital de una señal, se pueden implementar sobre ellas celdas de tipo MAC que poseen la complejidad computacional, la capacidad de memoria requerida y en ellas es posible manejar la longitud de palabra adecuada para la mayoría de operaciones que se requieren en aplicaciones de procesamiento digital. Sin embargo, en ocasiones la complejidad de estas operaciones es tal, que ocupan gran parte de los recursos de la FPGA, sobre todo cuando se trata de aplicaciones que demandan inmediatez de respuesta, aunque en aplicaciones de banda ancha las FPGA tienen la ventaja de proveer mayor ancho de banda que los DSP gracias a que integran múltiples MAC en un solo chip [10], por lo cual vemos que la cuota de aplicaciones tales como wireless, multimedia trasmisión satelital es muchísimo mayor en FPGA que en PDSP. [8]. Además existen muchos algoritmos implementados que funcionan funcionar sobre FPGA tales como CORDIC[9] (COordinate Rotation DIgital

[8] Tipo de dato de coma flotante consiste en precisión simple de 32 bits o precisión doble de 64 bits. El formato viene dado por tres campos, el primer campo se refiere al signo, el segundo campo el exponente y el tercer campo es la matiza[ver sección 1.8 de este documento]

[9] CORDIC es un algoritmo simple y eficiente para calcular funciones hiperbólicas y trigonométricas. Típicamente es usado cuando no hay disponible un hardware para multiplicaciones (por ejemplo,

Computer o el método de dígito por dígito), corrección de errores, donde la tecnología de arreglos de lógica programable o FLP (field Logic programmable) ha demostrado ser más eficiente que en los PDSP [11].

En los últimos tiempos, gracias al desarrollo tecnológico en el campo de los semiconductores es posible encontrar FPGA que incorporan en sus tarjetas de prueba procesadores digitales de señales DSP que sobrellevan gran parte del trabajo que de otra manera tendría que ser soportado sobre los multiplicadores y bloques lógicos de la FPGA, de esta manera, es posible aprovechar estos recursos en tareas relacionadas con la captura y visualización de la información [10].

En el futuro inmediato del procesamiento digital de señales, los PDSP dominaran aplicaciones que requieran complicados algoritmos como bloques complejos de estructuras if-then-else, mientras que las FPGA estarán presentes en el diseño de aplicaciones de tipo Front-End[10] como filtros FIR, algoritmos CORDIC y FFT [11].

1.6 Arquitectura FPGA Spartan 3A y características de la placa de desarrollo Spartan 3A starter kit de Digilent.

En el proyecto de desarrollo del prototipo de un sistema de análisis de señales de tipo poliscopio sobre FPGA, se utilizará una placa de desarrollo SPARTAN 3A Starter Kit de Digilent, que integra la FPGA Spartan 3A fabricada por la empresa Xilinx Inc. de referencia XC3S700A-FG484. Esta FPGA es un dispositivo de lógica programable de bajo costo y de gama media, que incorpora 700 mil compuertas equivalentes a 13248 celdas lógicas programables, cuenta con un arreglo de 896 CLB para un total de 3584 Slices (cada CLB incorpora 4 Slices) distribuidos en una matriz de 40 filas por 24 columnas. Ademas, la FPGA Spartan 3A XC3S700A incorpora 8DCM y 20 multiplicadores dedicados. En la tabla 1.1 se muestran las principales características de la FPGA Spartan 3A XC3S700A [12].

La distribución de los elementos básicos de una FPGA descritos en la sección 1.5 para la FPGA Spartan 3A XC3S700A se muestra en la figura 1.4. Aquí se observa que los recursos de uso global ocupan el área más extensa del circuito integrado. Aunque también se presentan algunos circuitos para operaciones específicas. Los recursos adicionales son los DCM (Digital Clock Manager, Administradores Digitales de Reloj), las memorias RAM y los multiplicadores dedicados de 18x18 bits con signo.

En microcontroladores y FPGAs simples) pues las únicas operaciones que requiere son suma, resta, desplazamiento de bits y búsqueda en tablas.

[10] El término se refiere a aplicaciones de diseño de lógica digital que interactúan de una manera u otra con el usuario, entregando información o prestando algún tipo de servicio. Este usuario puede ser un humano que haga uso del sistema o puede ser otro sistema. La idea general es que el front-end es responsable de recoger entradas de los usuarios, y ser procesadas de tal manera que cumplan las especificaciones para que el back-end pueda usarlas. La conexión entre front-end y el back-end es un tipo de interfaz.

Dispositivo		XC3S700A
Compuertas de sistema		700K
	Equivalente en celdas lógicas	13.248
Arreglo de CLB	Filas	48
	Columnas	32
	CLB	1.472
	Sílices	5.888
RAM distribuida(bits)		92K
Bloques de RAM(bits)		360K
Multiplicadores dedicados		20
DCM		8
Máximo de E/S		372
Máximo de pares de E/S diferenciales		165

Tabla 1.1. Principales características de la FPGA Spartan 3A XC3S700A [12].

Los CLB de la Spartan 3A están formados por 4 paquetes o divisiones llamados Slices, a su vez estos Slices están constituidos por 2 LUT (Look Up Table, Tabla de verdad) cada uno. Los Slices, al contar con 2 LUT, permiten la implementación de cualquier función lógica de cuatro entradas, sin limitación en cuanto a la complejidad y su retardo será constante sin importar la función lógica que realice, pero habrá limitación si se excede las 4 entradas. En la Figura 1.5 se muestra un diagrama de la distribución interna de un Slice [12].

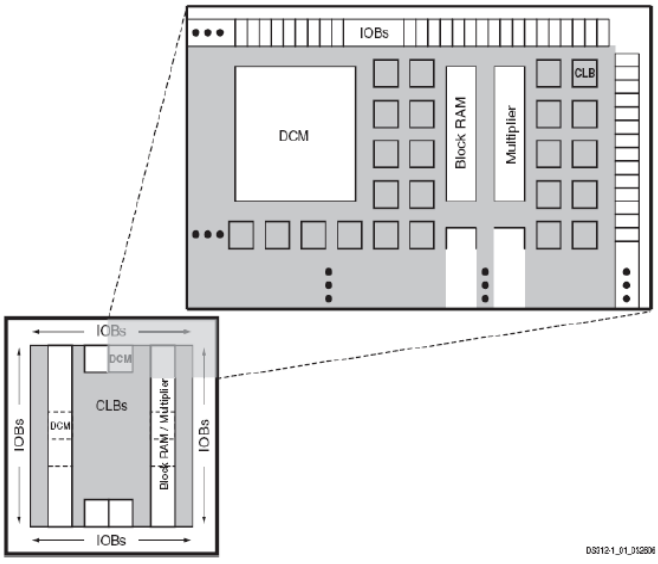

Figura 1.4. Arquitectura de la FPGA Spartan 3A [12].

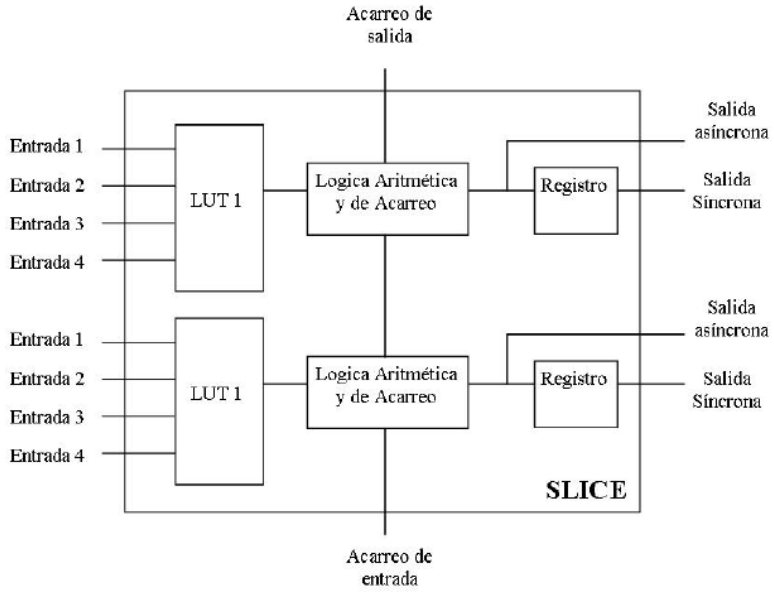

Figura 1.5. Distribución Interna de un Slice [12].

La tarjeta de desarrollo Spartan 3A Starter kit Integrada por Digilent Inc. sobre la que está la FPGA Spartan 3A XC3S700A, es una placa de evaluación que integra diversos componentes de diferentes fabricantes, que se pueden utilizar en una implementación sobre ella. Así por ejemplo, los integrados que conforman el ADC (Analogic-Digital Converter) son de Linear Technology Inc., el LCD es Hitachi, las memorias son fabricadas por Intel, etc. Una de las características más importantes de la tarjeta tiene que ver con el reloj que integra, éste es de 50 MHZ, aunque también se puede integrar un reloj auxiliar mediante conexión SMA[11]. La tarjeta también cuenta con un conector VGA[12] de cuatro bits por cada color, alcanzando hasta 4096 colores a una resolución de 640x480 pixeles. Además, se integra en la placa dos puertos PS/2[13] para ratón y teclado y dos puertos de tipo serial con interfaz RS232, también cuenta con un puerto RJ45 Ethernet 10/100A y con un conector estero para salida de audio. El circuito de captura de señal analógica presente en la tarjeta es un ADC dual LTC1407A de 14 bits de resolución con preamplificador LTC6912 de ganancia programable, soporta una entrada entre 0.4V hasta 2.9V y entrega la salida de manera serial mediante la interfaz SPI. [13]

A continuación, en la tabla 1.2 se presentan los diferentes componentes que integra la placa de evaluación Spartan 3A Starter kit.

[11] (Subminiature versión A) Tipo de Conctor en rosca para cable coaxial útil hasta una frecuencia de 33Ghz
[12] Video Graphics Array , Sistema gráfico de pantallas para PC que utiliza conector VGA de 15 clavijas D subminiatura que se comercializó por primera vez en 1988 por IBM.
[13] PS/2 es un conector de 6 pines mini-Din que se utiliza para conectar teclados y ratones a un sistema de cómputo compatible con PC.

Interruptores y pulsadores:	• Cuatro interruptores deslizables. • Interruptor de suspensión • Interruptor de encendido. • Cuatro pulsadores. • Botón pulsador rotatorio. • Pulsador de reprogramación
Relojes:	• Reloj integrado de 50MHz • Se puede integrar un reloj auxiliar de 133MHz • Conector SMA para proveer una fuente de reloj externa. •
Salidas /entradas:	• pantalla LCD de 2 líneas de 16 caracteres. • conector hembra VGA estándar. • Dos puertos serial RS-232 (uno hembra y uno macho). • 2 puertos PS/2 DIN[14] de seis pines. • RJ45 Ethernet 10/100. • ADC de dos canales con preamplificador • DAC de 4 canales de 12 bits. • Dos interface Jtag, una USB y otra paralela. • conector estéreo de audio digital.
Memorias:	• Memoria flash PROM (NOR) de 32Mbit • Dos interfaces flash SPI[15], • memoria flash PROM de 16Mbit • memoria data-flash de 16Mbit. • Memoria DDR2 SDRAM de 512Mbit
Expansión	• conector FX-2 de 100 pines • Dos conectores diferenciales de 43 pines con capacidad de transmisión-recepción 600Mbps por cada par. • Dos conectores de uso general de seis pines.

Tabla 1.2 Componentes de la Placa de Desarrollo Spartan 3A Starter Kit [13].

[14] conectores eléctricos de alta frecuencia, multi-pin, estandarizados por el Deutsches Institut für Normung.
[15] Serial Peripheral Interface – Es un bus estándar de comunicaciones.

1.7 Transformada rápida de Fourier.

Para realizar el análisis en frecuencia requerido para obtener una gráfica de voltaje contra frecuencia que se quiere lograr con el sistema de análisis de señales de tipo poliscopio, es necesario utilizar la transformada rápida de Fourier, así que en esta sección se abordan los aspectos generales concernientes a esta herramienta del procesamiento digital de señales.
La Transformada de Fourier es una herramienta matemática que tiene un amplio uso en lo referente al tratamiento digital de señales, se encuentra implementada bajo la forma de dispositivos electrónicos de reconocimiento de voz e imagen; puede ser aplicada a varios campos como análisis espectral, ecuaciones diferenciales, resolución de problemas elásticos estacionarios y dinámicos, etc [14].

Aunque el análisis de Fourier tiene una larga historia que involucra un gran número de personas, el desarrollo matemático de la transformada de Fourier fue explicado por Jean Baptiste Joseph Fourier, en su libro la Teoría Analítica del Calor, publicado en 1822 [9]; posteriormente, en 1965 Cooley y Tukey publicaron su artículo “Un algoritmo para calcular las Series de Fourier Complejas”, el cual es conocido como algoritmo FFT (Fast Fourier Transform) y que con el desarrollo acelerado de las computadoras digitales ha permitido la aplicación de la FFT (Fast Fourier Transform, Transformada Rápida de Fourier) a diferentes campos [15].

La FFT es un algoritmo de alta eficiencia para calcular la Transformada Discreta de Fourier DFT (Discrete Fourier Transform) y su inversa. Permite transformar una señal digital del dominio del tiempo al dominio de la frecuencia y de este modo conocer su espectro.

Cuando se lleva a cabo una Transformada discreta de Fourier de una señal, se obtiene una secuencia de números complejos de longitud *N/2 +1* que consta de dos señales a las salida, las cuales contienen información de las magnitudes de los componentes en forma de seno y coseno, es decir, parte real y parte imaginaria.[16].

La DFT se expresa como:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-j2k\pi n}{N}} \qquad \text{Ec.1.1}$$

Dónde:

- x(n) es el conjunto de datos original.
- X(k) es la transformada de x(n).
- N es el número de elementos.
- k representa la variable en el dominio de la frecuencia.

Se percibe que la ecuación 1.1 describe el cómputo de N ecuaciones. Por ejemplo, si N = 4 y si se hace:

$$W=e^{\frac{-j2\pi}{N}} \qquad \text{Ec.1.2}$$

La ecuación 1.1 puede ser escrita como

$$\begin{aligned} X\ 0 &= x_0\ 0 \cdot W^0 + x_0\ 1 \cdot W^0 + x_0\ 2 \cdot W^0 + x_0\ 3 \cdot W^0 \\ X\ 1 &= x_0\ 0 \cdot W^0 + x_0\ 1 \cdot W^1 + x_0\ 2 \cdot W^2 + x_0\ 3 \cdot W^3 \\ X\ 2 &= x_0\ 0 \cdot W^0 + x_0\ 1 \cdot W^2 + x_0\ 2 \cdot W^4 + x_0\ 3 \cdot W^6 \\ X\ 3 &= x_0\ 0 \cdot W^0 + x_0\ 1 \cdot W^3 + x_0\ 2 \cdot W^6 + x_0\ 3 \cdot W^9 \end{aligned}$$

Ec.1.3

El conjunto de ecuaciones 1.3 puede ser representado más fácilmente en forma matricial

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \cdot \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix}$$

Ec.1.4

O en forma compacta:

$$X\ n = W^{n \cdot k} \cdot x_0(k) \qquad \text{Ec.1.5}$$

El examen de Ec.1.5 revela que, ya que W y posiblemente $x_0(k)$ sean complejas, entonces son necesarias N^2 multiplicaciones complejas y N(N-1) adiciones para realizar el cómputo matricial requerido.

La FFT debe su éxito al hecho que el algoritmo reduce el número de multiplicaciones y adiciones requeridas en el cálculo de Ec.1.4 de N^2 a $N\log_2 N$ [16].

La idea que permite esta optimización es la descomposición de la transformada a tratar en otras más simples y éstas a su vez hasta llegar a transformadas de 2 elementos donde k puede tomar los valores 0 y 1. Una vez resueltas las transformadas más simples hay que agruparlas en otras de nivel superior que deben resolverse de nuevo y así sucesivamente hasta llegar al nivel más alto. Al final de este proceso, los resultados obtenidos deben reordenarse. El algoritmo pone algunas restricciones en la señal y en el espectro

resultante. Por ejemplo: la señal de la que se tomaron muestras y que se va a transformar debe consistir de un número de muestras igual a una potencia de dos. [17]

Para llevar a cabo la transformada rápida de Fourier se requieren tres etapas:
Primero, se divide una señal de N muestras en el dominio del tiempo en N señales cada una conformada por un solo punto [18].

En la figura 1.6 se muestra la descomposición en el tiempo de una señal de 16 muestras. Aquí se observa el proceso descrito en el primer paso del algoritmo para el cálculo de la FFT.

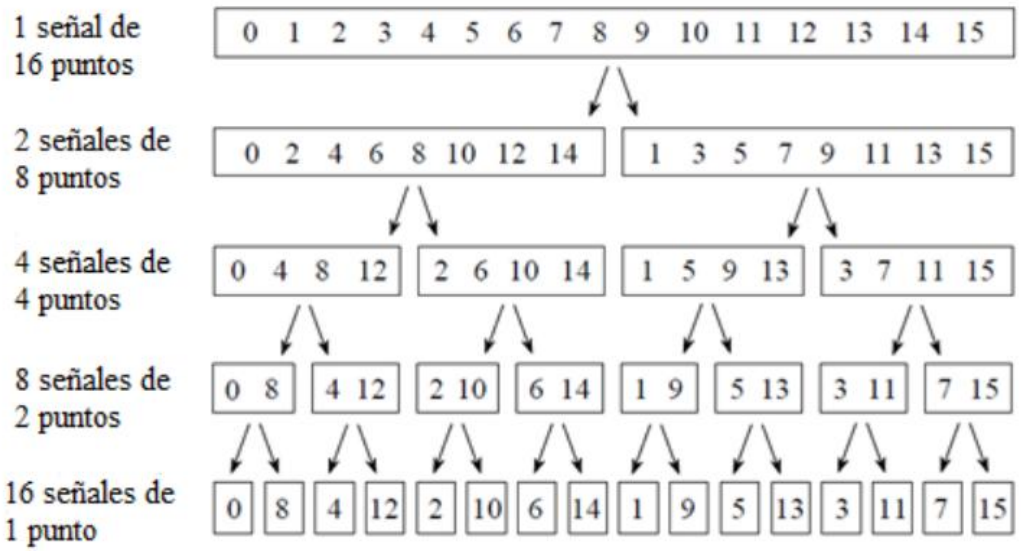

Figura 1.6. Descomposición en el tiempo de una señal de 16 muestras. Algoritmo de Inversión de bits

El algoritmo mostrado en la figura 1.6 se conoce como algoritmo de Inversión de bits, ya que el orden de las N muestras es reordenado contando en binario con los bits rotados de izquierda a derecha. [18]

El Segundo pasó, es obtener el espectro de frecuencia correspondiente a cada una de las muestras, el cual es equivalente a su valor actual, ya que debido al paso anterior, se encuentran en el dominio de la frecuencia. [18]

Por último los espectros de frecuencia en la FFT se combinan en pares, duplicándolos y sumándolos de tal manera que los espectros coincidan. Esto se logra agregando ceros en el dominio del tiempo a ambas señales. En una señal los puntos impares son ceros, mientras que en la otra señal los puntos pares serán ceros, es decir, una de las señales será desplazada a la derecha por una muestra, lo cual será equivalente a multiplicar su espectro por una señal sinusoidal [18]. De esta Manera, al sumar ambos espectros en el orden mostrado en la figura 1.7, se obtendrá el espectro final a la salida de la FFT. [19]

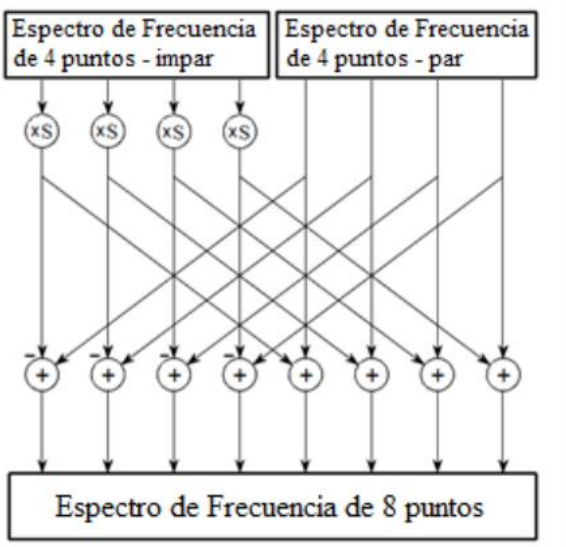

Figura 1.7. Método para obtener el espectro final.

A la estructura básica de este algoritmo en la cual se toman dos números complejos a la entrada y se obtienen dos números complejos diferentes a la salida, se le conoce como mariposa, debido a su apariencia alada. En la figura 1.8 se muestra un diagrama para una entrada de dos puntos.

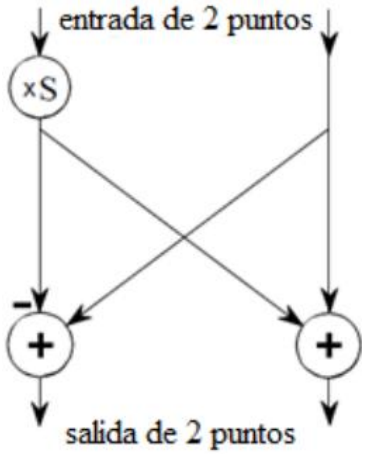

Figura 1.8. Mariposa de la FFT [19].

Por último, para mostrar la eficiencia que se obtiene con una implementación de la FFT contra la DFT, en la Figura 1.9 se presenta la relación entre el número de multiplicaciones requeridas usando el algoritmo FFT comparada con el número de multiplicaciones del método directo.

En las Ecuaciones 1.6 se muestra el número de operaciones requerida para la forma directa de la DFT y para la FFT.

$$f_{dir}\ N\ = N^2 \qquad f_{fft}\ N\ = \frac{N}{2} \cdot \frac{\ln(N)}{\ln(2)} \qquad N = 2 \ldots .1024$$

$$Ec.\,1.6$$

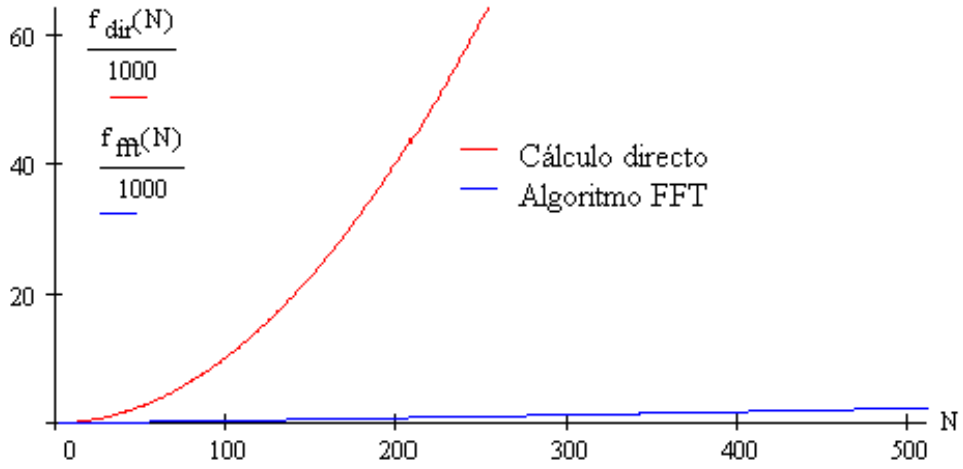

Figura 1.9. Variación del Número de multiplicaciones para el método directo y para el algoritmo de la FFT.

1.8 Representación en punto flotante o IEEE 754.

Muchas de las operaciones que se realizan dentro de los algoritmos implementados en el procesamiento de señales sobre FPGA, se llevan a cabo utilizando la representación en punto flotante de una señal, debido a que la manipulación aritmética y compleja es mucho más fácil y sobretodo porque esta representación ofrece mayor flexibilidad al momento de operar y facilita la presentación de la información. Teniendo en cuenta este hecho, en esta sección se presentan los aspectos básicos de la representación en punto flotante o IEEE 754.

El IEEE (Institute of Electrical and Electronic Engineering) aprobó en 1985 mediante el ANSI/IEEE Std 754-1985, el formato normalizado IEEE 754 para aritmética en coma flotante, el cual especifica cuatro formatos para la representación de valores en coma flotante: precisión simple (32 bits), precisión doble (64 bits), precisión simple extendida (≥ 43 bits) y precisión doble extendida (≥ 79 bits). Este estándar surge para evitar las limitantes de representación que se tenían al expresar un número mediante punto fijo, debido a que mediante este sistema el rango de valores que es posible representar con un número determinado de bits es mucho menor al estar la coma fija que al tener el sistema de coma flotante, aunque los algoritmos de las operaciones son mucho más complejos.

La representación en punto flotante de un número en un sistema de numeración base B se realiza mediante una mantisa m y un exponente e de la siguiente forma:

$$n=mB^{e} \qquad \text{Ec.1.7}$$

En la que *m* es el numero en punto fijo, *B* es la base del sistema de numeración y *e* es el numero entero que se denomina exponente.

Un ejemplo de un numero representado en el sistema de numeración decimal, B=10, en el formato de punto flotante es el $22.5x10^{3}$. Pero este número se puede representar de muchas formas, por ejemplo:

$$22.5x10^{3}=0.0225x10^{6}=225000x10^{-1}$$

Que se diferencian por la situación del punto en la mantisa y por el valor del exponente así que como se puede observar por este motivo esta representación recibe el nombre de punto flotante. Aunque las anteriores representaciones son válidas como punto flotante, se suele optar por una representación donde el punto este antes de la cifra más significativa distinta de cero. En el ejemplo seria $0.225x10^{5}$.

En el estándar se definen formatos para la representación de números en coma flotante, que incluye el cero y valores desnormalizados, así como valores especiales como infinito y NaN[16] (Not a Number, No es un Número), con un conjunto de operaciones en coma flotante que trabaja sobre estos valores. También se definen los métodos de redondeo que pueden ser hacia arriba o hacia abajo y el truncamiento.

En la figura se muestra un diagrama de la especificación del estándar para una representación binaria en punto flotante de precisión simple o de 32 bits, ya que en este caso el número se almacena en una palabra de 32 bits. S es el bit de signo, 0 si es un número Positivo y 1 si es un número negativo. Exp es el campo para el exponente de 8 bits que va desde -126 hasta 127. El campo de la mantisa de 23 bits expresa la base del exponente.

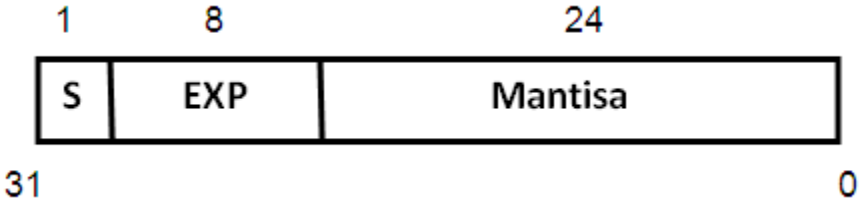

Figura 1.8. Representación de un número en Punto Flotante.

[16] se usa generalmente para expresar un resultado imposible de calcular, como el caso de las raíces negativas, indeterminaciones, etc.

Capítulo II Diseño e implementación.

En este segundo capítulo se presenta el diagrama modular propuesto para el sistema de análisis de señales tipo poliscopio utilizando FPGA, primero se describen cada uno de los bloques del prototipo, su función, entradas y salidas, luego se detallan aspectos de la implementación y funcionamiento del bloque, de los módulos y del prototipo. Además, se exponen las consideraciones de diseño, las limitaciones y alcances de sistema.

2.1 Diagrama Modular.

Una arquitectura de diseño sobre FPGA se basa en definir los diferentes componentes que implementa el sistema. Para el diseño se debe definir el funcionamiento de cada uno de estos componentes. Un sistema digital puede definirse a distintos niveles de abstracción y en tres dominios diferentes: comportamiento, estructura y físico. El dominio de comportamiento describe lo que hace el sistema, el dominio de estructura describe al sistema como una interconexión de componentes y el dominio físico describe la implementación física del sistema. [21]

Teniendo en cuenta esto, el diseño del sistema de Análisis de Señales tipo poliscopio Utilizando FPGA se basa en la arquitectura modular que se presenta en la figura 2.1. Aquí se muestran los componentes que conforman el prototipo y la interconexión entre ellos. Estos bloques son los componentes requeridos para la implementación del sistema de análisis de señales tipo poliscopio utilizando FPGA.

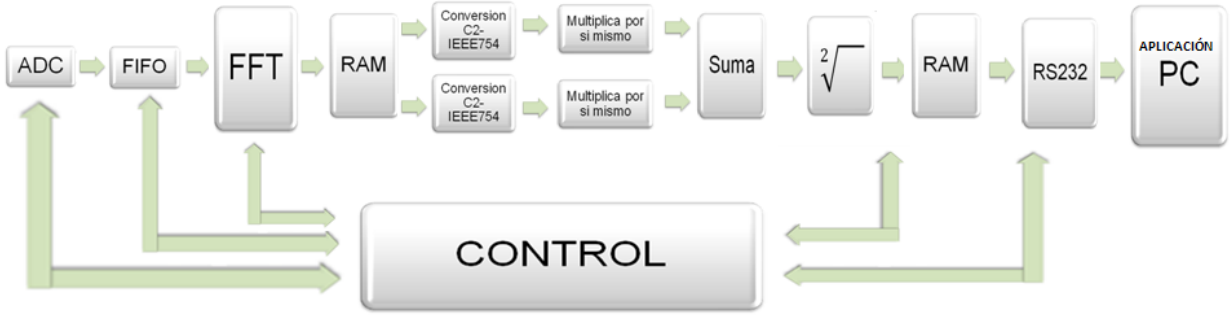

Figura 2.1 Diagrama modular del sistema de análisis de señales tipo poliscopio

EL objetivo del sistema de señales tipo poliscopio es analizar una señal de entrada para entregar una gráfica de voltaje contra frecuencia de esa señal en la salida, así que lo primero que se necesita es llevar esa señal al mundo discreto para poder manipularla y operarla utilizando la lógica digital de las FPGA; para esto se utiliza un conversor analógico a digital o ADC. Luego de esto, es necesario almacenar las muestras de la señal digitalizada utilizando una memoria, la particularidad de esta memoria es que el orden de las muestras de la señal que se tomaron de primero se almacenen primero para

que la señal original conserve su forma, así que se trata de una memoria FIFO (First In First Out, lo primero que entra es lo primero que sale).

Después de almacenar las muestras de la señal digitalizada se requiere realizar un análisis en frecuencia con el fin de determinar qué cantidad de energía está presente en las diferentes componentes de frecuencia de la señal, para esto se hace uso de la transformada rápida de Fourier, que es un algoritmo muy eficiente de la transformada discreta de Fourier, el cual se utiliza para llevar una señal de tiempo discreto al dominio de la frecuencia.

Una vez obtenidos los coeficientes discretos de Fourier, se analiza la magnitud de esta señal en el dominio de la frecuencia así que se obtiene el módulo de los coeficientes y para esta tarea es necesario que la señal se opere de manera adecuada, por lo tanto el prototipo del sistema de análisis de señales tipo poliscopio incorpora un conjunto de bloques de operaciones que, primero toman las salidas de la FFT (parte real y parte imaginaria) y las convierten de complemento a dos a una representación en punto flotante, luego elevan al cuadrado cada una de las salidas de la FFT, multiplicándolas por sí mismas, después suman ambos valores con el bloque de suma y por ultimo obtienen la raíz cuadrada de esta suma para así concluir con un valor de magnitud de la FFT de la señal de entrada en una representación de punto flotante o IEEE754.

Por último, se almacenan los valores de magnitud de la señal en una memoria RAM y se trasmiten al computador, al software de procesamiento matemático MATLAB, utilizando la interfaz RS232 con la que cuenta la placa de desarrollo de la FPGA. Es necesario utilizar una memoria RAM debido a que la velocidad con la que se procesan las muestras en el bloque de trasformada de Fourier y en los bloques de operaciones no es la misma velocidad con la que se trasmiten mediante el puerto RS232.

Se utiliza MATLAB para realizar la gráfica que se desea obtener y para desarrollar la aplicación de software que incorpora características de poliscopio, ya que este entorno de procesamiento matemático ofrece una gran versatilidad y muchas prestaciones, lo que permite que el manejo sea mucho más sencillo y que el sistema pueda ser escalable incorporando más funcionalidades como pueden ser aplicaciones de filtros, características de la señal, obtención de parámetros de líneas de transmisión cuyas señales se procesan en la FPGA, realizar capturas de pantalla o compartir resultados del sistema en una red y también comparar los resultados obtenidos sobre la FPGA con otros sistemas que realicen el mismo proceso utilizando procesamiento de software y no descripción de hardware.

2.2 Descripción de los bloques que integran el prototipo del sistema de análisis de señales tipo poliscopio utilizando FPGA.

A continuación se realiza una breve descripción de cada uno de los bloques presentes en el diseño del prototipo del sistema de análisis de señales tipo poliscopio. Esta descripción no profundiza en detalles de la implementación, solo se refiere a la función de cada bloque y se menciona sus entradas y salidas.

2.2.1 Bloque ADC.

El primer bloque del sistema es un conversor analógico a digital o ADC, pues se requiere que la señal de entrada se lleve a una representación discreta para poder realizar su tratamiento en la FPGA. Este ADC está incorporado en la placa de la FPGA y es un integrado de Linear Technology cuya referencia es LTC1407A, es un ADC de dos canales con preamplificador programable de ganancia variable de referencia LTC6912, cuenta con una resolución de 14 bits, con voltaje de referencia de 1.65 voltios por canal y una frecuencia de muestreo según el datasheet de hasta 1.5MHZ si se le incorpora un reloj de 100MHZ, por lo que, de acuerdo con el teorema de muestreo de Shanon-Nyquist, puede soportar señales de frecuencia de hasta 700KHZ. Cada canal del ADC soporta entradas entre 0.4V y 2.5 voltios, motivo por el cual es necesario que en su entrada se adecue el voltaje de la señal mediante algún circuito sujetador o retenedor. Las muestras digitales de la señal de entrada, de acuerdo a la conversión que realiza el ADC, corresponden a un valor entre -8192 y 8192 contenido en 14 bits en representación de complemento a dos, que se entregan de manera serial en una misma señal del ADC para ambos canales.

2.2.2 Bloque de memoria FIFO.

Esta memoria consiste en un bloque de memoria de acceso secuencial con registros de 14 bits para almacenar cada una de las muestras del ADC. El número de registros, es decir la profundidad de la memoria, depende del tamaño de FFT que se quiera para el sistema. La característica más importante de esta memoria es que conserva el orden de las muestras que recibe; en el ciclo de escritura la primera muestra se almacena en el primer registro, la segunda en el segundo registro y así sucesivamente. En el ciclo de lectura pasa lo mismo, la primera muestra leída es la del primer registro, la segunda es la del segundo registro, etc. Con esto se consigue que lo primero que fue guardado en la memoria sea lo primero que sea leído, es decir, el comportamiento FIFO (First In, First Out). Este comportamiento por parte de la memoria es necesario para mantener el orden de las muestras de la señal y con esto la forma de la señal muestreada.

2.2.3 Bloque de FFT.

La FFT (Fast Fourier Transform, transformada rápida de Fourier) es un algoritmo de alta eficiencia para el cálculo de transformada de Fourier de tiempo discreto DFT. Es de alta eficiencia porque, como se expuso en el capítulo 1, sección 1.7, la FFT logra disminuir el número de operaciones necesarias con relación a la DFT de N^2 a $N\log_2 N$, siendo N el número de muestras de la Transformada o tamaño de la transformada.

Este bloque de FFT toma N número de muestras, con 14 bits de longitud por muestra, de acuerdo a lo que se obtiene del bloque ADC, y en la salida entrega dos vectores en complemento a dos, uno para la componente real y otro para la componente imaginaria, ambos de tamaño $b_{xk}=b_{xn}+\log_2 N+1$, para una implementación de la FFT sin

escalamiento[17], donde b_{xn} es la longitud de cada muestra y N como ya se ha dicho, es el tamaño de la transformada. En este caso, como el ADC entrega 14 bits por muestra, b_{xn} = 14, así que, por ejemplo, para una FFT de N= 512 tendremos que, el tamaño de las salidas, tanto la parte real como la parte imaginaria, será de 24bits. [8]

El tamaño de la transformada no se define en esta etapa del diseño del prototipo debido a que es necesario realizar pruebas con los diferentes tamaños para determinar cuál es el más apropiado, que tenga el mayor rendimiento y un eficiente uso de recursos, ya que, en una implementación de este algoritmo de FFT se utilizan gran cantidad de recursos del dispositivo programable, en especial se utilizan muchos bloques multiplicadores y acumuladores, así que, si no se consideran los demás componentes de la implementación, como memorias, bloques de operaciones y los recursos utilizados en tiempo de ejecución, el sistema podría tener un mal desempeño, comportamiento errático (bloqueos, salidas inconsistentes, etc.), no podría funcionar al momento de procesar señales o simplemente no sería sintetizado a componente hardware.

2.2.4 Bloque de memoria RAM.

Este bloque almacena la salida de la FFT para que luego sea entregada al bloque de conversión de complemento a dos a punto flotante. El tamaño es igual al tamaño de la transformada rápida de Fourier que se implemente.

2.2.5. Bloque de conversión de complemento a dos a punto flotante (C2-IEEE754).

Desde la señal entregada por parte del ADC, se tiene una representación en complemento a dos para cada muestra, así que para cada entrada de la FFT con N=512, la salida que se obtiene es un par de vectores de 24 bits cada uno en representación de complemento a dos. La conversión de complemento a dos a representación en punto flotante se realiza debido a que, para obtener la magnitud de la FFT es necesario realizar una serie de operaciones sobre este par de muestras de salida y la representación en complemento a dos no es conveniente para este tipo de operaciones por varias razones, por ejemplo, porque al elevar al cuadrado cada componente de la FFT, se está elevando al cuadrado un vector de longitud 24 bits, así que se requiere de un vector de 48 bits para almacenar el resultado de esta operación, lo cual es excesivo y dispendioso para el manejo, en cuanto a recursos físicos y en cuanto a lógica a implementar. Además, porque al continuar realizando operaciones con este número de bits se requiere el uso de muchos acumuladores y multiplicadores y una alta exigencia del algoritmo CORDIC (Coordinate Rotation Digital Computer, método de dígito por dígito), lo que resulta en un amplio uso de recursos de la FPGA que disminuyen y hasta anulan la capacidad de esta para realizar otras tareas mucho más importantes, como la FFT. La representación en punto flotante

[17] en una implementación de un algoritmo de FFT, el escalamiento consiste en fijar las salidas en un tamaño igual al de la entrada y dividir estas salidas por un vector de escalamiento cuyo máximo valor se calcula con $2\log_2 N$ para evitar que las salidas excedan el tamaño fijado. Tiene múltiples inconvenientes, por ejemplo problemas de desbordamiento y menor exactitud que otros tipos de implementación de FFT.

siempre va a tener el mismo número de bits y permite realizar operaciones de una manera más cómoda ya que aprovecha las propiedades de los exponenciales, lo que ahorra recursos y hace un uso más eficiente de los multiplicadores y acumuladores.

El bloque de conversión de complemento a dos a punto flotante o estándar IEEE754 toma un vector de entrada en punto fijo con signo y lo transforma a un vector de punto flotante de precisión simple de 32 bits de longitud como especifica el estándar para este tipo de precisión. En la sección 1.8 del capítulo 1 se presentó una descripción de la representación en punto flotante para el caso de precisión simple.

En el Diseño del Prototipo del Sistema de Análisis de Señales Tipo poliscopio de la figura 2.1 se muestran dos bloques de conversión de complemento a dos a punto flotante a la salida del bloque de la RAM que almacena los resultados del bloque FFT, esto es porque la FFT como ya se mencionó, entrega a la salida dos vectores, uno para la parte real y otro para la parte imaginaria, y es necesario realizar la conversión de ambas componentes.

2.2.6 Bloque de multiplicación por sí mismo.

Elevar al cuadrado un número es multiplicar ese número por sí mismo. Este bloque se ha denominado multiplicación por sí mismo porque es mucho más sencillo y ocupa menos recursos, realizar una implementación que lea dos vectores de entrada, los multiplique y entregue el resultado a la salida, que una implementación que tome únicamente un vector de entrada y lo manipule para elevarlo al cuadrado que es lo mismo que multiplicarlo por el mismo. En este caso los dos vectores de entrada son el mismo vector, de esta forma se ahorra la lógica que adecua el vector para realizar la operación. La entrada de este bloque consiste de dos vectores de 32 bits que contienen el mismo valor en punto flotante y la salida del bloque es un vector de 32 bits en punto flotante con el resultado de la operación.

En el diagrama del prototipo del sistema de análisis de señales tipo poliscopio de la figura 2.1 se muestran dos bloques de multiplicación por sí mismo porque es necesario elevar al cuadrado tanto la parte real de la salida de la FFT como la parte imaginaria.

2.2.7 Bloque de suma.

Este bloque realiza la operación de suma de las dos componentes, la componente real y la imaginaria, una vez han sido elevadas al cuadrado. Las entradas del bloque son los dos vectores de 32 bits de componente real e imaginaria de la FFT elevados al cuadrado, en representación de punto flotante, que provienen de los bloques que los multiplican por sí mismos. La salida de este bloque es un vector de 32 bits en punto flotante que contiene el resultado de la operación.

2.2.8 Bloque de raíz cuadrada.

Este bloque obtiene la raíz cuadrada de un vector de entrada. La entrada es el vector de 32 bits en punto flotante y la salida es un vector de 32 bits en punto flotante con el resultado de esta operación.

2.2.9 Bloque de memoria RAM.

El bloque de memoria RAM actúa como buffer, almacenando los valores de magnitud de la FFT de cada muestra que provienen de la sección de operaciones. Este bloque es necesario porque el tiempo de procesamiento de la FFT y del bloque de operaciones es superior al tiempo que se tarda el sistema en trasmitir cada valor de magnitud hacia el computador mediante el módulo de transmisión serial RS232.

2.2.10 Bloque de transmisión RS232.

Este bloque se encarga de transmitir al computador vía puerto serial cada uno de los valores de magnitud de la FFT que se obtienen de los módulos anteriores. Para realizar la transmisión de estos valores, el bloque de transmisión RS232 primero adecua los vectores que contienen el valor de magnitud de la FFT al protocolo de transmisión serial, debido a que estos vectores de magnitud son vectores de 32 bits de longitud y el protocolo puede transmitir únicamente 8 bits de datos en cada envió.

2.2.11 Bloque aplicación PC.

El bloque denominado aplicación PC representa el computador en el que se reciben las muestras de la magnitud de la FFT vía puerto serial, para que mediante MATLAB se realice el despliegue de la señal a través de una gráfica de la magnitud de la FFT que se obtuvo en la FPGA. Este bloque también representa la aplicación que se desarrolla en MATLAB para efectuar tareas adicionales sobre la gráfica de la señal que permitan realizar aplicaciones de poliscopio.

2.2.12 Bloque de control.

Este bloque consiste en una implementación en VHDL que controla y coordina la ejecución de los componentes del sistema. Se encarga de realizar la conexión y sincronización entre los demás bloques, interactuando con las señales que recibe de ellos y enviando señales de control para establecer la secuencia de datos en el sistema.

2.3 Implementación del prototipo del sistema de análisis de señales tipo poliscopio.

Para desarrollar el sistema de análisis de señales de tipo poliscopio utilizando FPGA se implementan en VHDL cada uno de los bloques que se muestran en la figura 2.1. En esta sección se presentan los detalles de implementación de cada uno de estos bloques, se describe su funcionamiento y se exponen las consideraciones que se tuvieron en cuenta en el proceso de construcción de cada uno de ellos.

2.3.1 ADC.

El circuito de captura analógica que incorpora la Spartan 3A Starter kit consiste de un preamplificador programable LTC6912-1 que escala la señal analógica de entrada conectada a la interfaz J22 de la placa. La salida del preamplificador está conectada a un ADC LTC1407. Estos dos integrados son programados y controlados de manera serial desde la FPGA mediante el estándar SPI[18]. En la figura 2.2 se muestra una sección de la Spartan 3A donde se aprecia el circuito de captura analógica con el conector J22 de entrada.

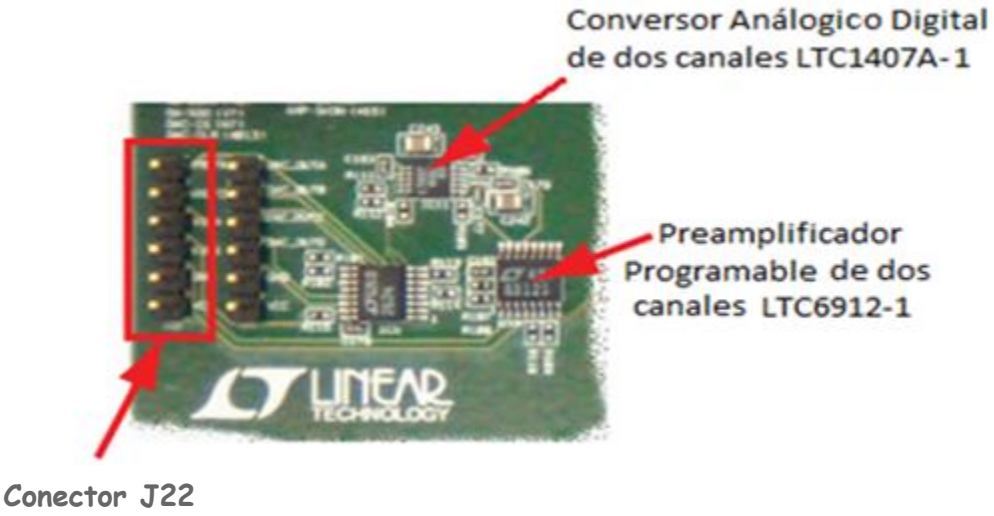

Figura 2.2. Circuito de captura analógica de la Spartan 3A.

Este circuito de captura analógica convierte una señal analógica de voltaje sobre los terminales VINA o VINB en una representación digital de 14 bits que llamaremos D[13:0], de acuerdo con Ec.2.1.

$$D[13:0] = \text{Ganancia} \times \frac{(V_{in} - 1.65)}{1.25} \times 8192 \qquad \text{Ec. 2.1}$$

[18] SPI, Serial Peripheral Interface – interfaz de periférico serial, es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus SPI permite controlar casi cualquier componente de electrónica digital que acepte un flujo de bits serie regulado por un reloj.

En la Ec.2.1 la ganancia es un valor cargado en el preamplificador programable.

El máximo rango de voltaje del ADC es de $\pm$1.25V centrado alrededor de un voltaje de referencia de 1.65V, por lo tanto 1.25 aparece en el denominador de Ec.2.1 para normalizar la entrada analógica de acuerdo a este valor. De esta manera la máxima representación será de 8192.

Para el manejo del ADC existen varias señales que realizan interfaz entre el circuito de captura analógica y la FPGA, a continuación se listan las señales que interactúan entre estos dos.

- **SPI_MOSI** (Serial data: Master Output, Slave Input.): esta señal es un vector de 8 bits que permite configurar la ganancia del preamplificador dual.

- **AMP_CS**: Esta señal es activa en bajo, cuando está en cero se programa la ganancia del preamplificador, cuando retorna a 1 esta configuración queda establecida.

- **SPI_SCK:** Reloj del bus SPI. Es el reloj que se le pasa al circuito de captura analógica.

- **AMP_SHDN:** Esta señal es activa en alto, sirve para apagar y hacer reset al circuito de captura analógica.

- **AD_CONV:** Esta señal es activa en alto, inicia el proceso de conversión.

- **ADC_OUT o SPI_MISO:** Esta señal es una salida de datos serial, entrega de manera serial los 14 bits de cada canal que contienen la representación de la señal analógica de entrada.

El circuito de captura analógica funciona de la siguiente forma:

- Primero es necesario configurar la ganancia del preamplificador contenida en SPI_MOSI, esto se hace cuando la señal AMP_CS está en nivel bajo.
- Cuando la señal AMP_CS vuelve a nivel alto la configuración de ganancia de los dos canales del preamplificador queda establecida. Con esto concluye la configuración del preamplificador.
- Luego de esto se pone la señal AD_CONV en alto para iniciar el proceso de conversión por parte del ADC.
- Los valores de conversión para ambos canales se entregan de manera serial en la señal ADC_OUT o SPI_MISO, por lo tanto es necesario considerar los tiempos requeridos para recibir los 28 bits correspondientes a los dos canales, más los retardos por configuración y respuesta del circuito.

En la figura 2.3 se presenta el bloque que se implementó en VHDL para el manejo del ADC de la Spartan 3A Starter Kit. Aquí se puede observar cada uno de los puertos que utiliza este componente. A continuación se describirán los más importantes. Es de resaltar que en la figura 2.3 las entradas al bloque se ubican a la izquierda y las salidas se ubican a la derecha.

CLK50: Entrada, es el reloj de la FPGA, funciona a 50MHZ
RESET: Entrada, Se utiliza para el reinicio del bloque ADC, devuelve los procesos a su estado inicial.
btn: Entrada, se utiliza para indicar el inicio de operación del ADC, permite configurar el preamplificador.
SPI_MISO: Entrada, sirve para recibir los valores que corresponden a cada muestra, se reciben bit a bit los 28 bits correspondientes a los dos canales del ADC.
Muestra: Salida, indica con un valor alto cuando una muestra en el canal 1 está completa.
AD_CONV: Salida, indica al ADC que debe capturar una nueva muestra**.**
AMP_CS: Salida, indica al ADC que se va a configurar el preamplificador.
SPI_MOSI: Salida, se utiliza para pasar el valor de ganancia al ADC.
SPI_SCK: Salida, reloj para el ADC configurado a la frecuencia de muestreo.
ADCO: Salida, entrega un vector con el valor de la muestra proveniente del ADC.

Los otros puertos de salida sirven para deshabilitar otros componentes de la FPGA que también utilizan el bus SPI.

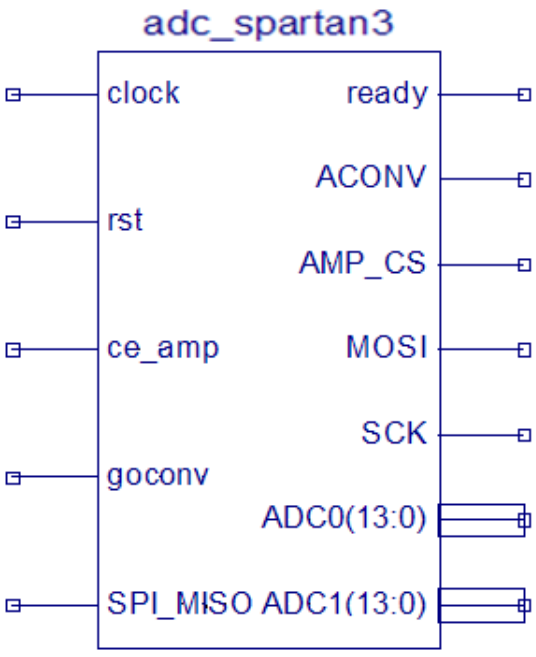

Figura 2.3. Bloque de ADC implementado en VHDL

El control de este circuito de captura analógica se realiza mediante el diagrama de estados que se muestra en la figura 2.4 para la etapa de configuración del preamplificador y en la figura 2.5 para la etapa de conversión por parte del ADC.

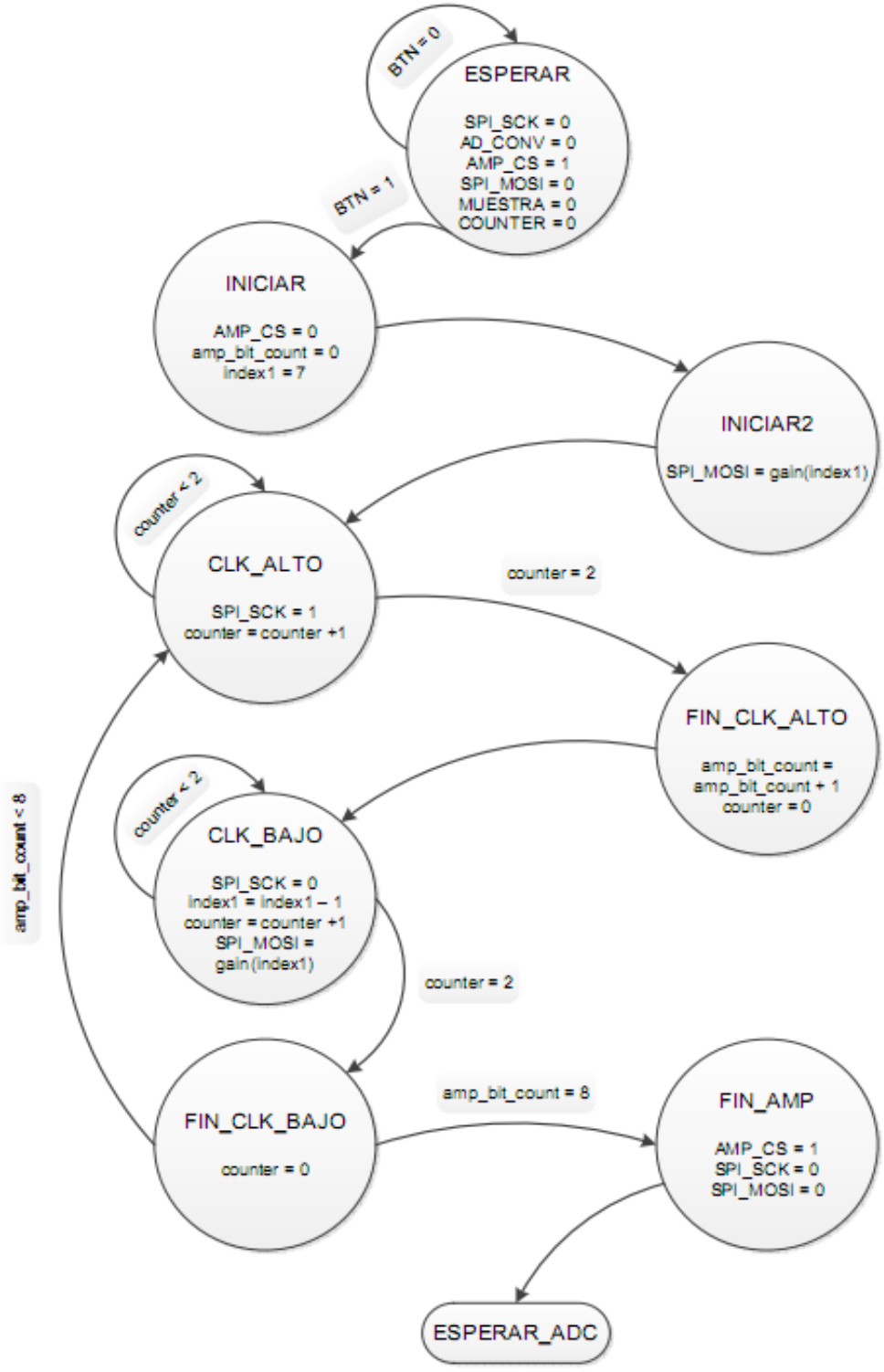

Figura 2.4. Diagrama de estados para configuración del preamplificador.

En la figura 2.4, se observa que el diagrama de estado para la etapa de configuración del preamplificador presenta 8 estados. En el primer estado, ESPERAR, se inician las señales involucradas en el control del preamplificador y del ADC, la señal AMP_CS se pone en 1 debido a que es activa en bajo. La señal muestra es una señal bandera utilizada para indicar que una muestra del ADC esta lista. Counter es un contador de uso general que se utiliza por ejemplo para conservar el estado actual en la máquina de estados.

Cuando la señal BTN toma un valor alto, se pasa al siguiente estado que es INICIAR. En este estado la señal AMP_CS pasa a nivel bajo para indicar que se va a configurar la ganancia del preamplificador. La señal amp_bit_count se fija en cero para que apunte a la primera posición del vector de configuración de ganancia, esta señal toma como máximo valor siete, para así garantizar que el proceso de pasar bits de configuración de ganancia solo se realiza ocho veces. La señal index1 toma valor siete para fijarse en la última posición del vector de configuración de ganancia SPI_MOSI porque en la configuración del preamplificador es necesario enviar primero el MSB y en último lugar en LSB. Realizadas estas asignaciones se pasa al estado siguiente.

EL siguiente estado es INICIAR2, aquí únicamente se asigna un valor contenido en el vector de ganancia determinado por index1, a la señal SPI_MOSI. Luego de esto se pasa al estado CLK_ALTO.

En el estado CLK_ALTO se genera un retardo de dos ciclos de reloj con el fin de garantizar que se cumpla la especificación de tiempo requerida por el preamplificador para almacenar un bit del vector de configuración de ganancia. Luego de que se cumple este retardo se pasa al estado FIN_CLK_ALTO.

En FIN_CLK_ALTO la señal amp_bit_count aumenta en 1 para indicar que se ha realizado el proceso una vez, la señal counter es puesta en cero y se pasa al estado CLK_BAJO.

En CLK_BAJO la señal SPI_SCK se fija en bajo, index1 disminuye en 1 para pasar el siguiente valor de configuración de ganancia, counter aumenta en 1 para generar un retardo y en SPI_MOSI se almacena el siguiente valor del vector de ganancia gain, determinado por index1. Una vez se han cumplido dos retardos se pasa al siguiente estado.

En FIN_CLK_BAJO el contador es puesto en cero y se pasa al siguiente estado, si amp_bit_count es menor que ocho quiere decir que aún no se han pasado todos los valores del vector de ganancia a la señal SPI_MOSI por lo tanto se requiere realizar el proceso de nuevo, así que se regresa al estado CLK_ALTO porque no es necesario realizar otra vez la iniciación de señales, y con esto concluye el cliclo de reloj. Si la señal amp_bit_count es igual a ocho quiere decir que ya se pasaron a SPI_MOSI todos los valores del vector de configuración de ganancia y por tanto es necesario finalizar la configuración del preamplificador colocando la señal AMP_CS en alto, lo que se hace en el siguiente estado, que en este caso es FIN_AMP.

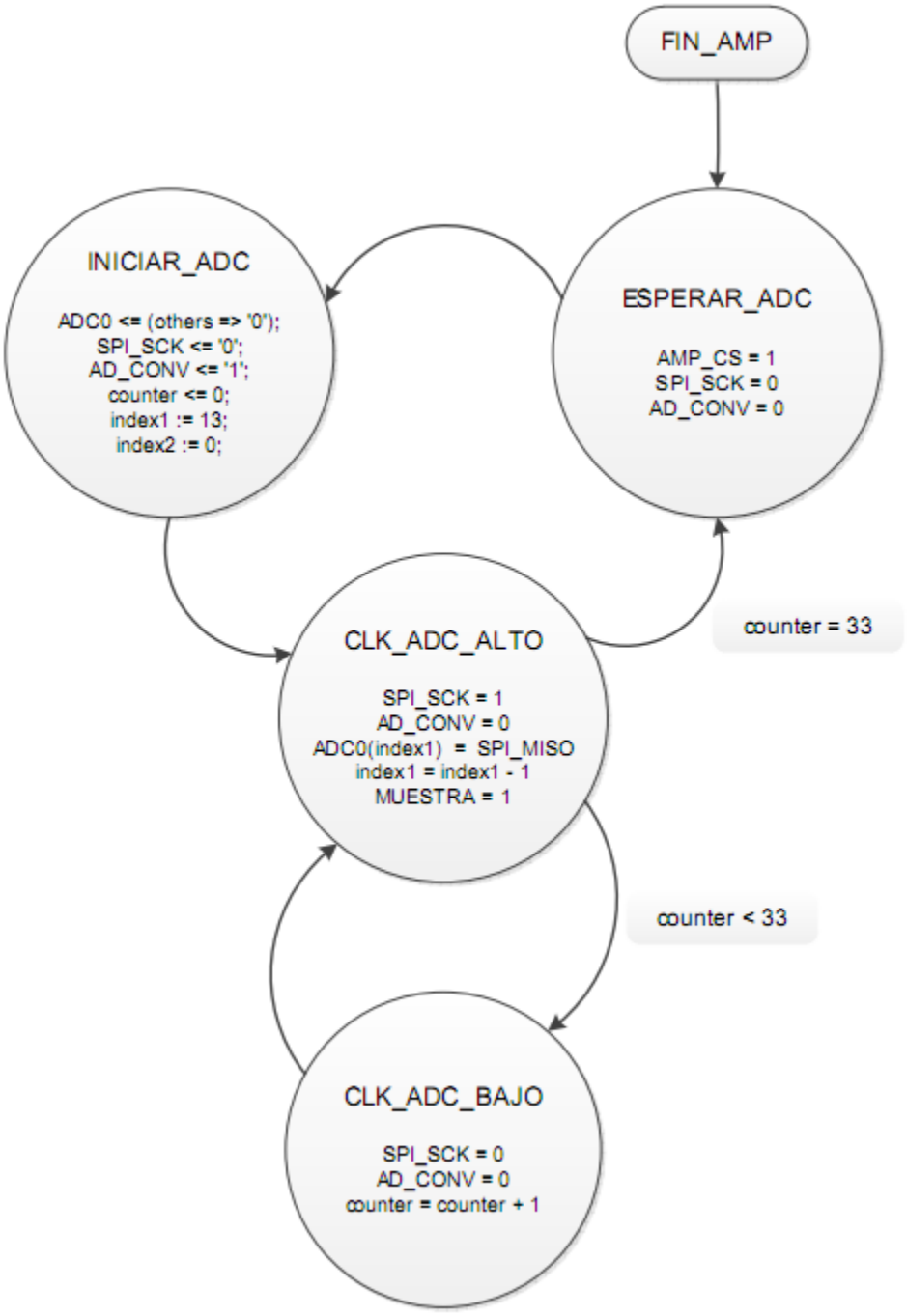

Figura 2.5. Diagrama de estados para el ADC.

Como se observa en la figura 2.5, una vez concluida la configuración del preamplificador el estado siguiente es ESPERAR_ADC, en este estado se fija AMP_CS en alto para indicar que el preamplificador ha sido configurado, SPI_SCK y AD_CONV se ponen en nivel bajo y se pasa al estado siguiente.

El siguiente estado es INICIAR_ADC, en este estado el vector ADC0 que contendrá las muestras de la señal del canal 1 del ADC se inicia con todos sus valores en cero, SPI_SCK se sigue en cero y AD_CONV se fija a nivel alto para indicar que se empieza a realizar la conversión por parte del ADC. Counter se fija en cero y la señal index1 se fija en 13 para indicar que se van a capturar los 14 bits correspondientes a una muestra del canal 1. Index2 se fija en cero porque no se van a realizar capturas desde el canal 2 del ADC.

El estado siguiente es CLK_ADC_ALTO, en este estado SPI_SCK se pone en nivel alto para generar el ciclo de reloj, AD_CONV se pone en nivel bajo porque ya se indicó que se iba a capturar una muestra, ADC0 almacena en la posición index1 el valor de bit que le entrega SPI_MISO y se disminuye en uno el valor de index1. Para pasar al estado siguiente se examina la señal counter, si es menor de 33 entonces se va a CLK ADC_BAJO para que aumente el contador y siga capturando datos de muestra, y si counter es igual a 33 quiere decir que ya se almacenaron todos los bits de muestras y se puede iniciar un nuevo proceso de conversión pasando al estado ESPERAR_ADC. Es importante señalar que la señal MUESTRA toma valor alto cuando counter tiene valor 17 indicando que ya están listos los 14 bits del vector de muestra de señal. Se espera a que tenga valor 17 y no 14 como se puede suponer al considerar que el ADC almacena muestras de 14 bits, debido a que al inicio del proceso, cuando la señal ADC_CONV genera un pulso, transcurren tres ciclos de reloj hasta que SPI_MISO entregue un bit de un vector de muestra.

El siguiente estado es CLK_ADC_BAJO, aquí la señal SPI_SCK toma valor bajo, AD_CONV se conserva en cero y counter aumenta en 1.

El periodo de muestreo consiste en el tiempo transcurrido entre una muestra y otra, que es de 1.38us. Que se traduce en una frecuencia de muestreo de 724Khz. Esto se puede ver en la figura 2.6 donde se muestra una simulación del ADC y se marcan los tiempos de interés.

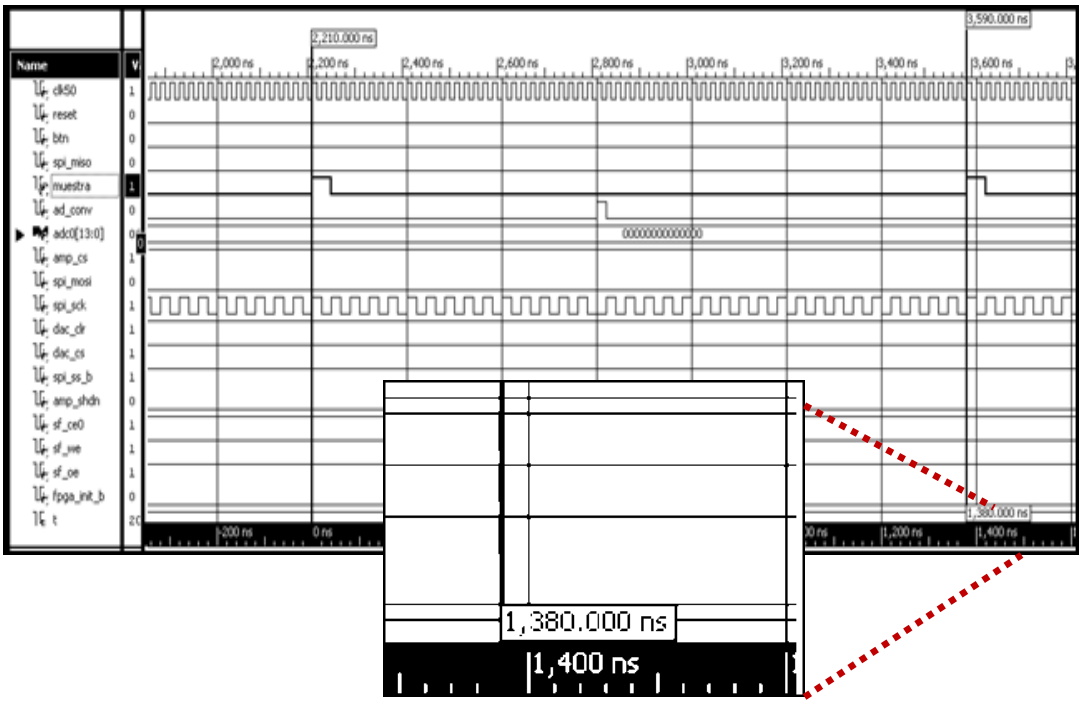

Figura 2.6. Simulación del ADC, diagrama de tiempos para observar el periodo entre muestras.

2.3.2 Memoria FIFO.

El bloque de memoria FIFO almacena las muestras provenientes del ADC conservando el orden para que la señal conserve su forma. Consiste en un conjunto de registros de 14 bits cada uno, debido a que esta es la longitud de palabra que maneja el ADC para cada muestra. La profundidad de la memoria FIFO, es decir, el número de registros que agrupe, depende del tamaño de la transformada rápida de Fourier que se utilice. En la figura 2.7 se muestra el bloque de la memoria FIFO implementado en VHDL, aquí se observan las entradas y las salidas del componente. Las entradas se ubican en la izquierda y las salidas se ubican en la derecha. A continuación se describen las entradas y salidas más importantes del componente de memoria FIFO.

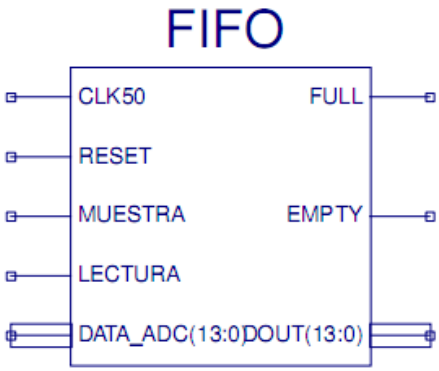

Figura 2.7. Bloque FIFO implementado en VHDL.

CLK50: Entrada, es el reloj de la FPGA, funciona a 50MHz.
RESET: Entrada, se utiliza para el reinicio del bloque FIFO, devuelve los procesos a su estado inicial.
MUESTRA: Entrada, recibe un valor alto para indicar a la FIFO que se debe preparar para recibir una muestra del ADC.
LECTURA: Entrada, recibe un valor alto para indicar a la FIFO que se ponga en modo lectura.
DATA_ADC: Entrada, recibe el vector de datos del ADC que corresponde a una muestra de la señal de entrada.
FULL: Salida, devuelve un valor alto cuando la memoria está llena.
EMPTY: Salida, pone un valor alto a la salida siempre que la memoria haya sido leida completamente o haya sido reiniciada.
DOUT: Salida, entrega a la salida el vector de datos correspondiente a una muestra de la señal de entrada.

En la figura 2.8 se presenta un diagrama de flujo para explicar el funcionamiento de la memoria FIFO. Aquí se observa que la memoria FIFO inicia su operación con las señales WRITE_EN_aux y READ_EN_aux con un valor bajo, se tiene que counter aumenta en uno con cada ciclo de reloj y cuando llega a dos, la memoria FIFO se pone en ESPERAR_ESCRIBIR. La señal counter espera dos ciclos de reloj para garantizar que la memoria FIFO sea inicializada de manera correcta pues en un primer momento, las señales EMPTY y FULL tienen el mismo valor y si no se da el tiempo necesario para que tomen sus estados correspondientes, se tendrá un mal funcionamiento del componente.

Continuando con el diagrama de la figura 2.5, cuando se tiene un valor alto en el puerto de entrada denominado MUESTRA quiere decir que existe una muestra lista por parte del ADC para ser almacenada en la FIFO, así que se verifica el estado de FULL, si esta en uno significa que la memoria está llena y se puede leer su contenido, así que se pasa a ESPERAR_LEER, si esta en cero pasa a ESCRIBIR y se pone WRITE_EN_aux en uno. En la FASE ESCRIBIR se verifica el estado de FULL, si esta en uno se va a ESPERAR_LEER y si esta en cero se devuelve a ESPERAR_ESCRIBIR.

En la fase ESPERAR_LEER se verifica el estado de EMPTY, si es cero significa que la memoria no está vacía, así que se examina el estado de LECTURA. Si el estado de LECTURA es alto se pasa a LEER, aquí se lleva READ_EN_aux a nivel alto y se lee la memoria hasta que EMPTY tenga valor alto, es decir hasta que la memoria este vacía. Si el Estado de LECTURA es un nivel bajo se regresa a ESPERAR_LEER.

En ESPERAR_LEER, si el estado de EMPTY es uno quiere decir que la memoria está vacía y se regresa a ESPERAR_ESCRIBIR.

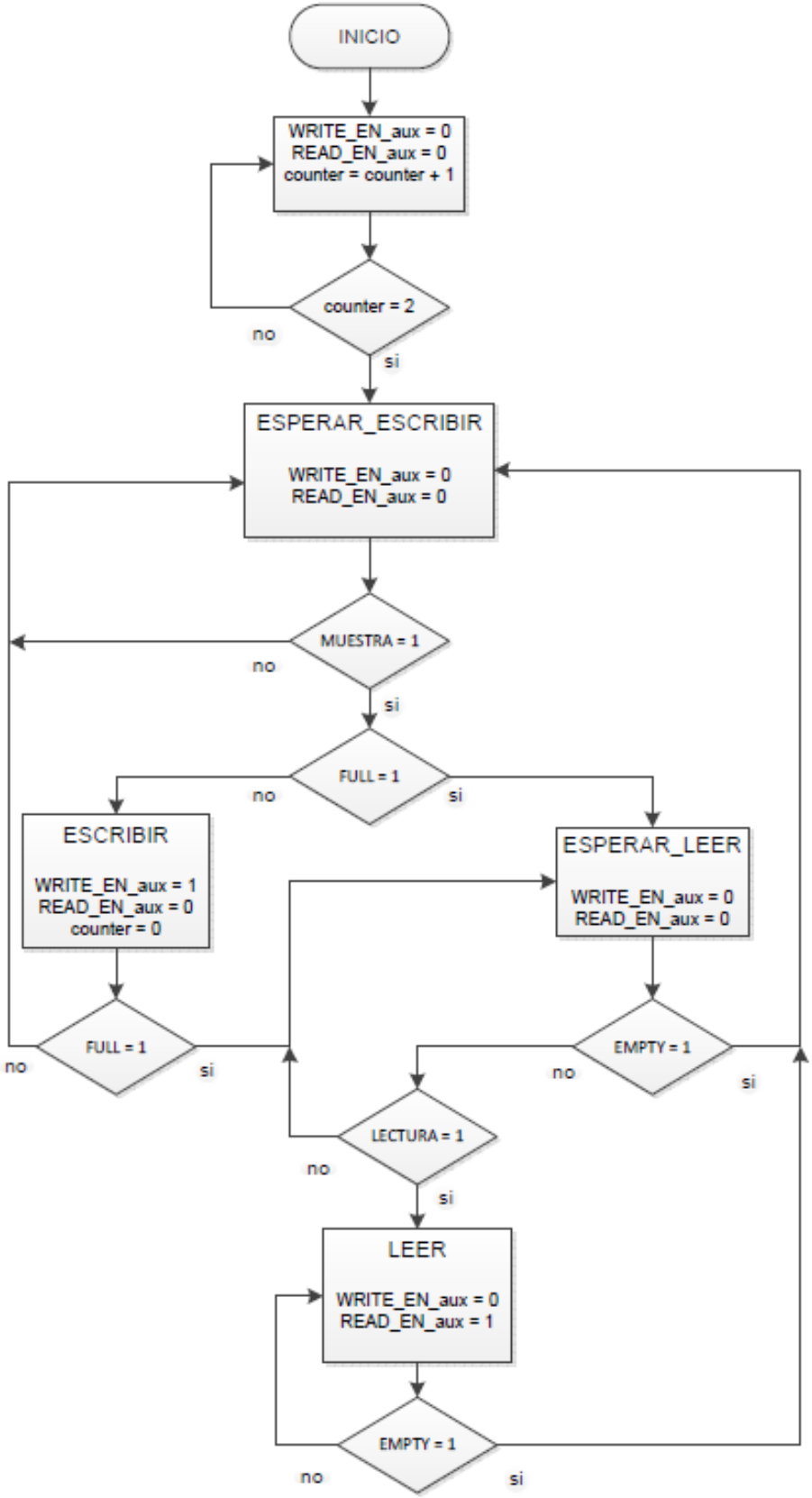

Figura 2.8. Diagrama de flujo memoria FIFO.

2.3.3 Bloque de FFT.

Este bloque denominado FFT realiza una transformada rápida de Fourier a partir de las muestras de la señal de entrada. Su implementación consiste en un CORE IP (Intellectual Property, Nucleo con Propiedad Intelectual) desarrollado por Xilinx Inc.

Un CORE IP consiste en una estructura que describe un componente cuyos parámetros de diseño pueden ser cambiados antes de la síntesis pero su código HDL no está disponible. Un CORE IP provee cierta flexibilidad pero está definido y optimizado para una familia específica de dispositivos programables.

La gran ventaja del uso de un CORE en el diseño sobre FPGA, además de que son diseños ya probados y depurados, es que aumenta en gran medida la rapidez de desarrollo de un prototipo, entregando en unos pocos pasos un componente que puede tomar meses en implementarse desde cero. Por otra parte, la desventaja más significativa del uso de un CORE radica en que, si bien presenta una estructura flexible en cuanto a la configuración de muchas de sus características, existen parámetros que son fijos, por lo tanto el diseño tiene que adaptarse a estas condiciones. Además, el estar sujeto a una licencia privada, limita mucho su uso en la mayoría de los diseños, y como están diseñados para un dispositivo específico son una opción poco utilizada en el desarrollo de proyectos sobre FPGA.

El CORE IP que se utiliza es el Fast Fourier Transform v7.1 propiedad de Xilinx Inc., el proceso para generar un CORE IP mediante el ISE Core Generator Tool se puede consultar en el Anexo A de este documento.

La configuración elegida para el CORE IP de la FFT se muestra y se justifica a continuación.

Al generar el CORE IP de la FFT aparece una ventana donde se seleccionan: el tamaño de la transformada, el reloj que utilizara, el tipo de arquitectura para procesar la mariposa. En la figura 2.9 se muestra la primera ventana de configuración.

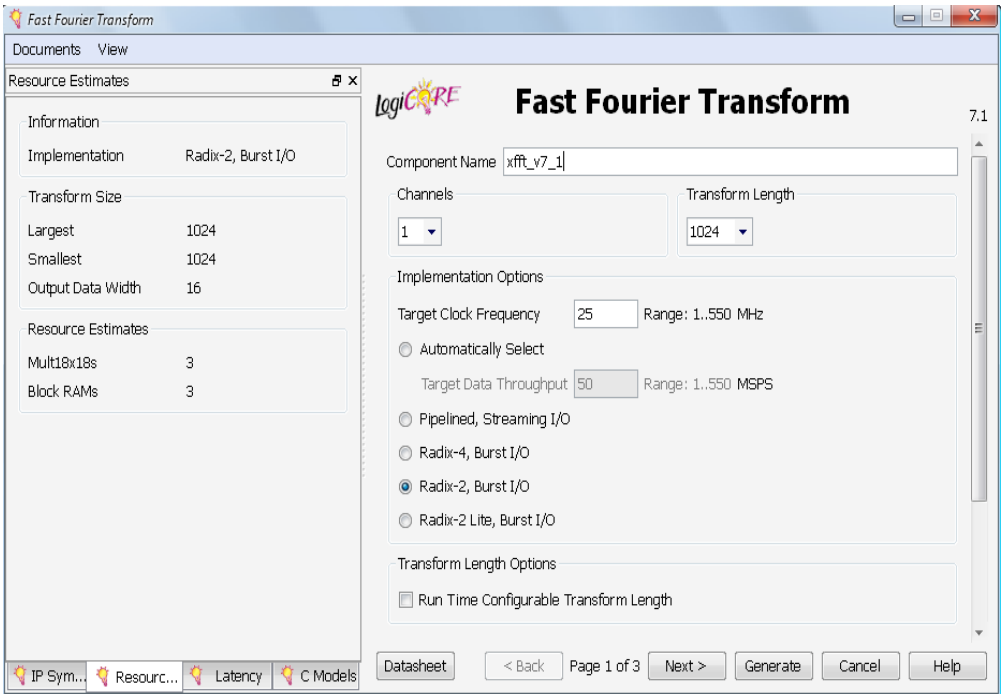

Figura 2.9. Configuración del CORE FFT. Paso 1.

Aquí se fija el tamaño de la FFT en 1024 muestras. El número de canales se fija en 1, este parámetro indica cuantas transformadas se llevarán a cabo al interior del bloque. El reloj de la transformada se establece en 25 MHZ para poder realizar el empalme con los demás componentes del sistema.

Se elige la arquitectura Radix-2 Burst I/O debido a las limitaciones de hardware con que cuenta el Spartan 3A. Dicha arquitectura procesa un arreglo de datos mediante la sucesión de pasos en los cuales el algoritmo efectúa mariposas de raíz-2 que recogen dos números complejos y regresa dos números complejos de mayor tamaño. Una arquitectura superior como Radix-4 Burst I/O implica un excesivo uso de multiplicadores y no quedarían recursos suficientes para el resto del sistema. Una arquitectura en Pipelined Streaming supera la cantidad de multiplicadores con los que cuenta la Spartan 3A así que sería imposible su implementación en esta FPGA. Por último, la arquitectura de Radix-2 Lite, Burst I/O presenta un pobre desempeño comparada con Radix-2 Burst I/O.

La siguiente ventana de configuración se presenta en la figura 2.10. En esta ventana se establece el tipo de dato para la entrada y salida de la FFT, la longitud de palabra para cada muestra, las opciones de escalado y redondeo, los pines adicionales, el orden de la salida y el tiempo de retardo para la lectura inicial de datos o tiempo de offset.

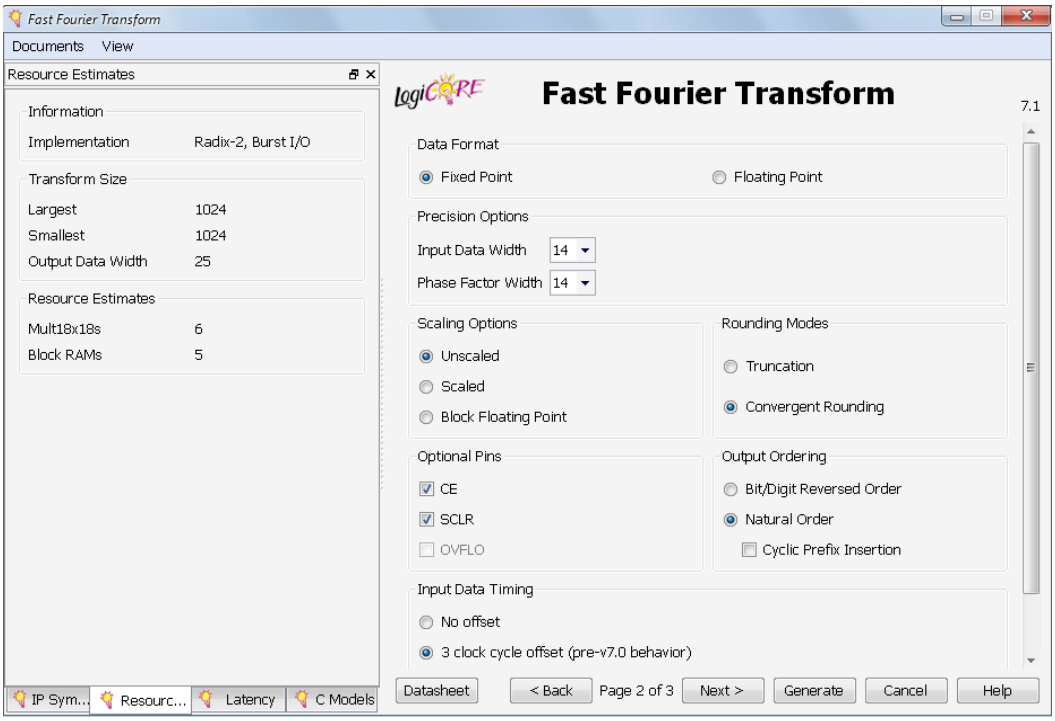

Figura 2.10. Configuración del CORE FFT. Paso 2.

Se elige el tipo de dato como punto fijo pues las muestras de la señal de entrada provenientes del ADC y almacenadas en la memoria FIFO están en este formato en representación de complemento a dos. El tamaño de cada muestra o longitud de palabra se fija en 14 bits debido a que el ADC entrega cada muestra en un vector de 14 bits.

En las opciones de escalamiento se escoge sin escalamiento, porque cuando se utiliza la opción de escala, se tiene una programación de escalamiento en un factor de 1, 2, 4 u 8 en cada etapa, si la escala es insuficiente se producirá un desbordamiento ya que la salida de la mariposa crecerá más allá del rango dinámico. Esta escala se aplica a cada etapa del algoritmo, lo que hace más lento el procesamiento de la FFT, y además no existe un método definitivo para el cálculo de esta escala. Igualmente, cuando se utiliza escalamiento, se realiza un redondeo en cada etapa de la mariposa y como se tiene que el número de etapas es $\log_2 N$, si N es 1024 entonces serán 10 etapas, así que el redondeo introducirá un error en cada etapa, que irá aumentando en la etapa siguiente, disminuyendo la precisión de todo el sistema.

En los modos de redondeo se opta por redondeo convergente debido a que este hace un redondeo hacia abajo mientras que el truncamiento hace un recorte del exceso de bits, lo que se traduce en pérdida de información.

En la sección de pines opcionales se añade el pin CE o pin de habilitación de operación de la FFT, con el fin de evitar que la FFT realice operaciones en instantes no válidos para evitar el consumo innecesario de recursos. El puerto SCLR se habilita para permitir un

reset síncrono de la FFT una vez se ha realizado la operación sobre un conjunto de muestras.

El ordenamiento de la salida se escoge en orden natural, porque no se quiere alterar la disposición con la que venía la señal a lo largo del sistema.

Por último, en input data timing, se selecciona un offset de tres ciclos de reloj para realizar una correcta sincronización entre la memoria FIFO y la entrada de carga de datos a la FFT.

En la última ventana de configuración del CORE FFT se escoge el tipo de memoria RAM que se utilizara para almacenar las muestras y se selecciona el método de optimización de resultados de la FFT. En la figura 2.10 se muestra este último paso de configuración de la FFT.

En el tipo de memoria se elige la opción de bloques de memoria RAM ya que ocupa menos área del FPGA y su síntesis requiere de menor tiempo. En las opciones de optimización se opta por usar cuatro multiplicadores para optimizar el rendimiento.

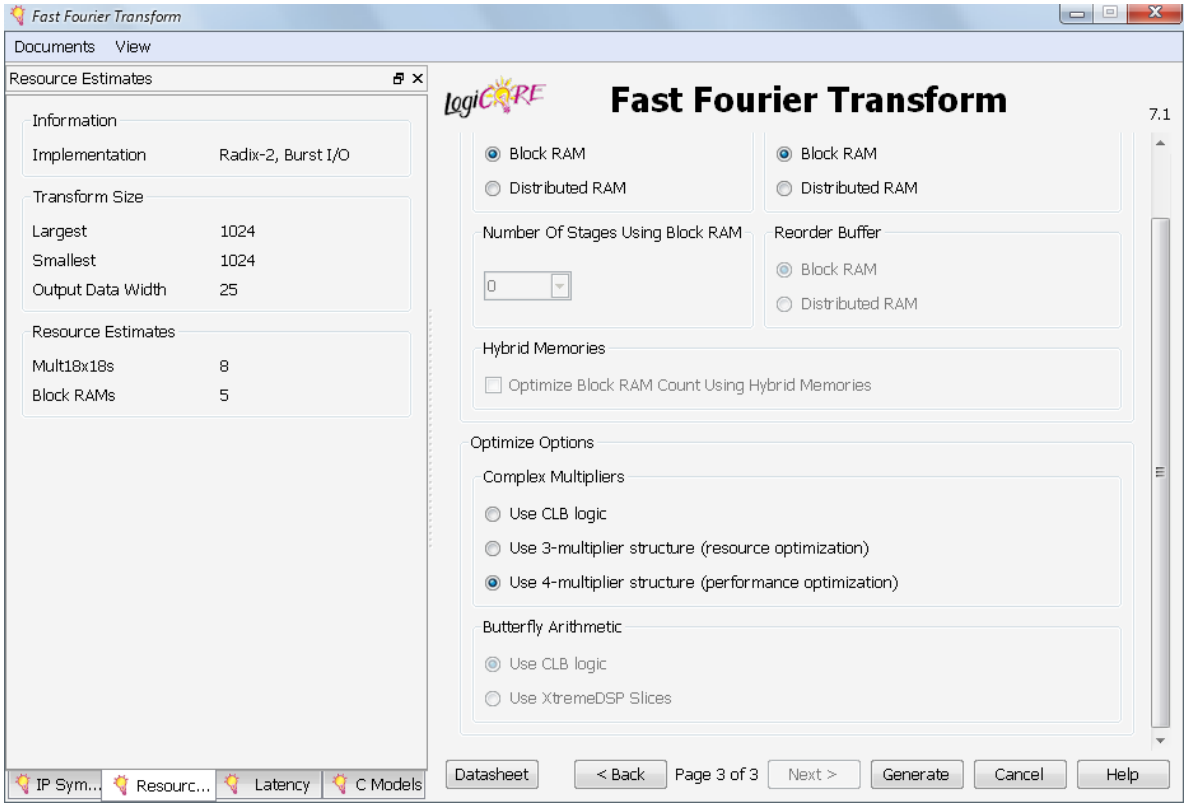

Figura 2.11. Configuración del CORE FFT. Paso 3

En la figura 2.12 se muestra un diagrama del CORE FFT generado. Aquí se observan las entradas a la izquierda y las salidas a la derecha.

A continuación se explican los puertos utilizados en este CORE. Es de resaltar que estos puertos generados son para este caso específico pues el bloque de la FFT genera los puertos dependiendo de las opciones de configuración utilizadas en la etapa de generación del CORE IP.

XN_RE: Es la entrada para la parte real

XN_IM: Es la entrada para la parte imaginaria, se utiliza cuando se quiere calcular la transformada inversa o IFFT, si se quiere una transformada directa se pasan ceros a esta entrada.

START: Sirve para dar comienzo a la transformada.

UNLOAD: Se establece en uno para indicar que se van a descargar los datos.

FWD_INV: Señal que indica el tipo de transformada a realizar, 1 para realizar una transformada directa y 0 para realizar una transformada inversa.

FWD_INV_WE: Habilita el uso de FWD_INV. Es activa en alto.

SCLR: Es activa en alto, sirve para realizar un reset síncrono sobre el bloque FFT.

CE: Habilita el reloj de la FFT, es activo en alto.

CLK: Reloj de la FFT.

XK_RE: Componente real de la transformada de Fourier.

XK_IM: Componente imaginaria de la transformada de Fourier.

XN_INDEX: Índice de los datos de entrada. Indica en que registro de la FFT está.

XK_INDEX: Índice de los datos de salida.

RFD: Read For Data. Activa en alto, indica cuando la FFT está lista para que se carguen datos.

BUSY: Indica con un valor alto cuando la FFT está realizando alguna operación.

DV: Esta señal se pone en alto cuando existe un dato válido en la salida.

EDONE: señal de casi listo, se pone en alto un instante antes de que la operación de cálculo de la FFT haya sido realizada.

DONE: Se pone en alto para indicar que el cálculo de la transformada se ha completado.

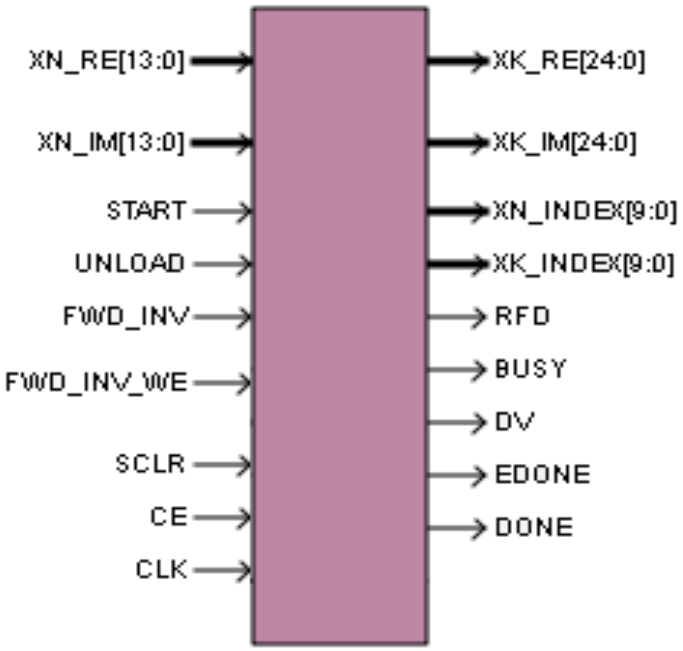

Figura 2.12. Modulo CORE FFT. Puertos de entrada y salida.

Para realizar el control de operaciones y facilitar la interconexión con los demás componentes del sistema de análisis de señales de tipo poliscopio, el CORE FFT se encapsula en un control que denominamos **fft.** En la figura 2.13 se muestra el bloque de este control con las entradas en la izquierda y las salidas en la derecha.

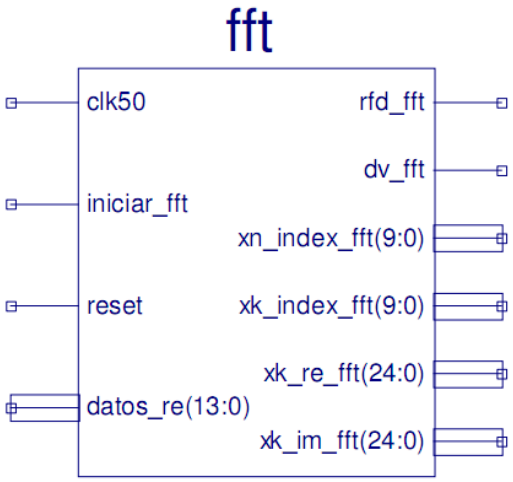

Figura 2.13. Bloque Control fft. Entradas y salidas.

A continuación se realiza una descripción de los puertos del bloque de control de la FFT.

clk50: Este puerto es la entrada de reloj de 50MHZ de la FPGA.

iniciar_FFT: Este puerto de entrada recibe un valor alto cuando se requiere que la FFT entre en operación.

reset: Está conectado al reset del CORE FFT y cumple la misma función, reinicia el bloque completamente.

datos_re: Este puerto de entrada recibe el vector de datos proveniente de la memoria fifo.

rfd_fft: Está conectado a RFD del CORE FFT asi que en la salida pone un valor alto cuando el CORE FFT está listo para que se carguen datos.

dv_fft: Está conectado a DV del CORE FFT y pone un valor alto cuando se tienen datos válidos a la salida de la FFT.

xn_index_fft: Índice para la entrada. Índica la posición de la muestra de entrada en el que esta la FFT

xk_index_fft: Índice para la salida. Indica el número de dato de salida en el que esta la FFT.

xk_re_fft: Salida parte real de la transformada de la FFT, es un vector en complemento a dos de 24 bits de longitud.

xk_im_fft: Salida parte imaginaria de la transformada de la FFT, es un vector en complemento a dos de 24 bits de longitud.

El funcionamiento de la FFT dentro del sistema de análisis de señales tipo poliscopio se explica mediante el diagrama mostrado en la figura 2.14. En este diagrama se han llamado las variables con una nomenclatura en letra minúscula debido a que no son los puertos del CORE FFT ni tampoco los puertos del bloque de control fft sino que son variables auxiliares de control para facilitar la asignación de estados, la escritura y lectura sobre los puertos de los componentes. Sin embargo las variables asociadas a un puerto llevan un nombre que permite relacionarlas con este.

Siguiendo el diagrama de la figura 2.14, el inicio de la operación de la fft se realiza poniendo ce_fft en alto para activar el reloj del bloque, las demás variables de control se ponen en cero. Luego de esto se verifica el estado de busy para confirmar que no se esté realizando ninguna operación sobre la FFT. Si la FFT está ocupada se espera a que se desocupe para continuar, cuando la FFT está libre se examina el estado de iniciar, se realiza un reset si se requiere que la FFT entre en operación, es decir si iniciar esta en uno. Si no es así no se avanza al siguiente estado.

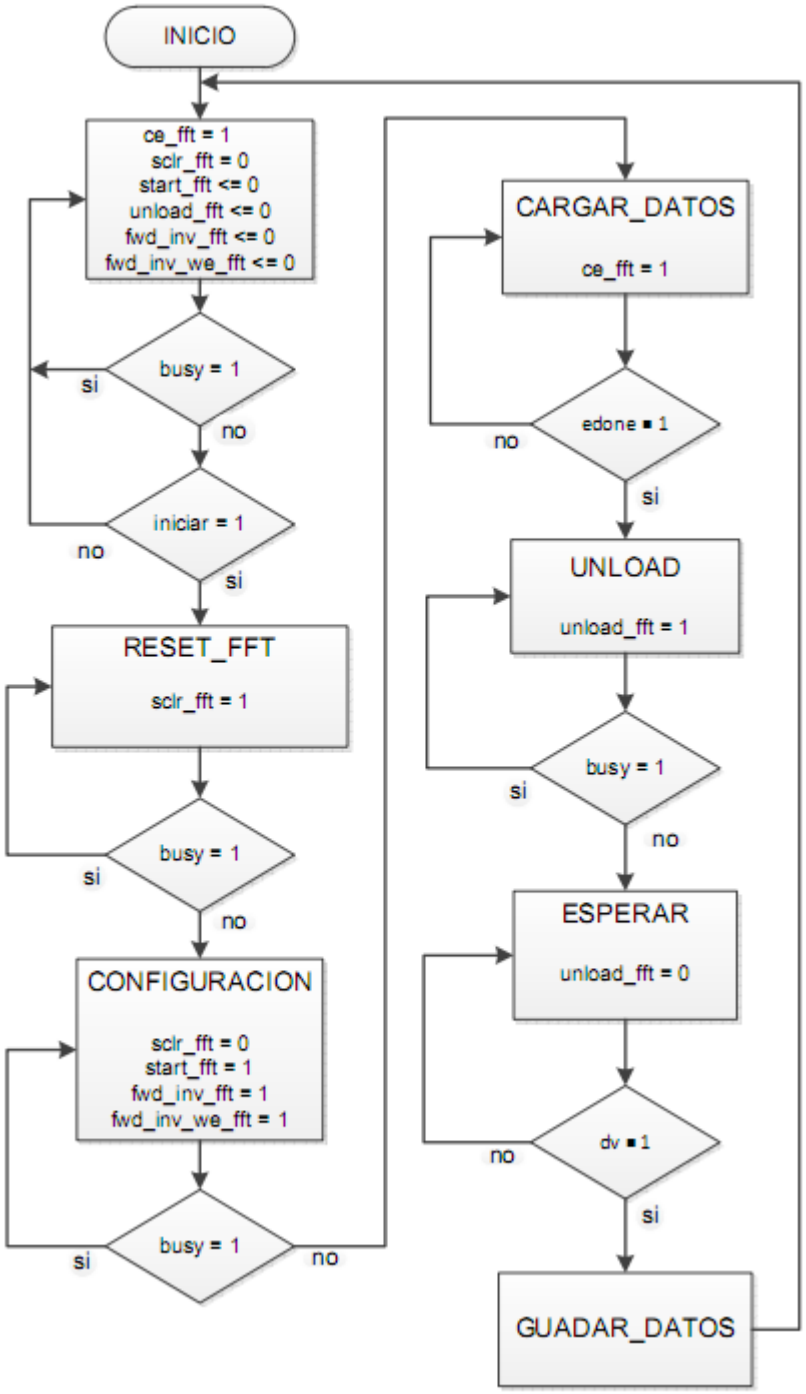

Figura 2.14. Diagrama de funcionamiento del bloque fft.

Una vez se ha realizado el reset para garantizar que la FFT este vacía, se verifica que no esté ocupada mediante la variable busy, si la FFT no está ocupada, se fija en transformada directa el tipo de transformada a realizar utilizando las variables fwd_inv_fft en un valor alto y fwd_inv_we_fft en alto para habilitar la anterior.

Para continuar se verifica que la FFT no esté realizando alguna operación, examinando el estado de la variable busy; si la FFT está libre se procede a cargar los datos de entrada, para esto se habilita el reloj de la FFT mediante la variable ce_fft puesta en un valor alto.

Una vez cargados los datos, se espera hasta que la señal edone tenga valor uno, es decir hasta que la operación de cálculo de la FFT este casi lista. Cuando se indica que la operación está casi lista, se indica al bloque FFT que se van a descargar los datos poniendo un uno en unload_fft. Para continuar, una vez más se verifica que busy valga cero. En la etapa ESPERAR se regresa unload_fft a nivel bajo y se espera a que dv_fft tome valor uno, es decir a que se tengan datos válidos listos para ser leídos, cuando esto sucede se procede a guardar los datos y por último se regresa a la etapa de inicio.

En cada etapa del proceso se verifica el estado de busy para poder continuar, debido a que esta señal pone uno lógico cuando la FFT está realizando una operación y si se ordena alguna instrucción al bloque cuando este está ocupado, la instrucción no se ejecuta y se genera un error.

Una vez concluido el proceso los datos son guardados en una memoria RAM que se ubica a la salida del bloque FFT. Ambas componentes, la parte real y la parte imaginaria se concatenan para ser almacenadas en una sola memoria.

2.3.4 Bloque RAM.

Este bloque almacena los datos provenientes del bloque FFT. La parte imaginaria ha sido concatenada a la parte real, con el fin guardar estas en un solo registro, buscando ahorro de recursos.

La memoria RAM consiste de un conjunto de registros cuya profundidad es igual al tamaño de la FFT y la longitud de palabra por registro es dos veces el tamaño del vector de salida para una componente de la FFT. La lectura y escritura de la memoria se realiza aumentando un contador que recorre uno a uno los registros, cuando se realiza la operación de lectura existe una señal que se establece en valor alto, cuando la operación es de escritura esta misma señal se configura en valor bajo.

La salida del bloque de memoria RAM se conecta al bloque que realiza la conversión

En la figura 2.15 se muestra un esquema del componente de memoria RAM, aquí se detallan las entradas y salidas de este módulo. Las entradas se ubican en la izquierda del bloque y las salidas en la derecha.

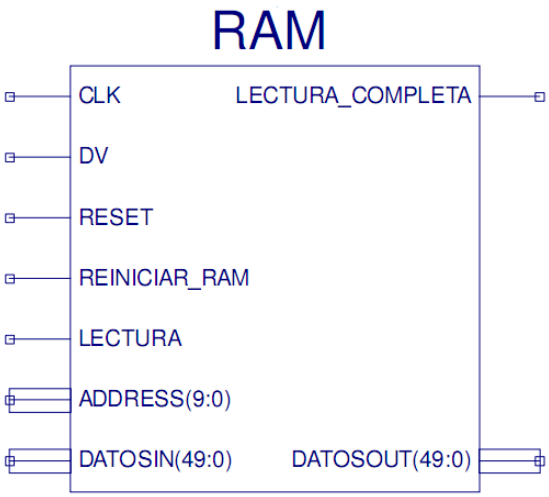

Figura 2.15. Bloque de Memoria RAM.

La descripción de los puertos presentes en el componente se hace a continuación.

CLK: Es la entrada de reloj de la memoria

DV: Es una entrada que indica que se tiene un dato valido, esta señal viene del bloque FFT y sirve para que la RAM se ponga en modo de escritura.

RESET: Sirve para inicializar la memoria RAM. Es activo en alto

REINICIAR_RAM: Coloca todas las señales de control de la memoria en la posición inicial.

LECTURA: Un valor alto indica que la memoria está en modo lectura, un valor bajo indica que se está en espera.

ADDRESS: Es el vector de dirección, esta entrada corresponde al index del bloque FFT.

DATOSIN: Es la entrada de datos provenientes de la FFT.

LECTURA_COMPLETA: Este valor se pone en uno cuando se ha realizado la lectura de todas las posiciones de la memoria RAM, cuando esta señal esta en uno el control general pone la entrada LECTURA en cero.

DATOSOUT: Es la salida de los datos almacenados en la memoria.

En la figura 2.16 se muestra el diagrama de secuencia de la memoria RAM para ilustrar su funcionamiento.

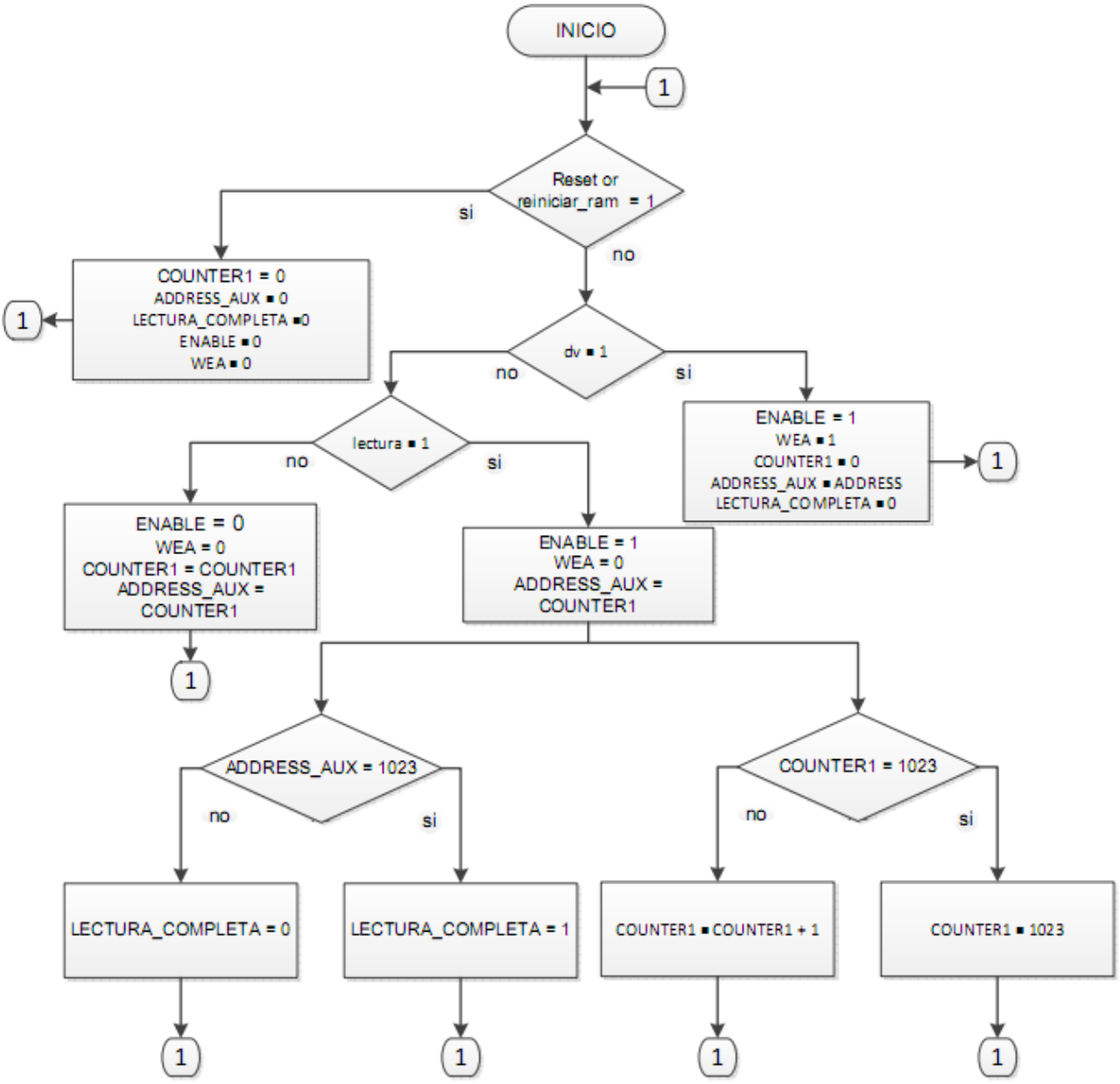

Figura 2.16. Diagrama de Secuencia de la Memora RAM.

Cuando se inicia la operación de la memoria RAM se examina el estado de reset y de reiniciar_ram, si alguna de estas señales tiene un valor alto, las señales de control de la memoria RAM son establecidas en cero y se retorna a la posición de inicio. Si alguna de estas dos señales esta en cero, entonces se inspecciona el valor de dv.

Si la señal dv vale uno, se habilita el proceso que realiza la escritura sobre la RAM, la señal ENABLE se lleva a alto para habilitar el cambio de valor sobre el puerto WEA que es el que permite con un valor alto la escritura en la RAM y con un valor bajo la lectura. En el proceso de escritura ADRESS recibe el valor de index de la FFT y lo pasa a una variable auxiliar llamada ADRRES_AUX que está conectada al puerto de la memoria que permite aumentar la posición de registro. Cabe aclarar que el index de la FFT va aumentando conforme se entregan los resultados de la transformada.

Si la señal dv vale cero, se examina el contenido de la señal lectura. Si lectura es cero entonces el contador que recorre las direcciones no aumenta, si la señal de lectura es uno se pone ENABLE en uno y se habilita la lectura haciendo WEA igual a uno, en este caso el contador COUNTER aumenta en uno si es menor a 1023 como se puede ver en la bifurcación de no, del condicional que se ubica en la parte inferior derecha del grafico de la figura 2.16. El uso del contador que recorre las direcciones en el modo lectura de la memoria RAM es necesario debido a que una vez se ha realizado el proceso de escritura utilizando el index de la FFT se requeria el uso de otra señal para recorrer las direcciones en el modo lectura. En la bifurcación de si, del condicional que se ubica en la parte inferior izquierda del grafico de la figura 2.16 se observa que cuando se ha recorrido toda la memoria en el modo lectura, es decir, cuando ADRESS_AUX es igual a 1023, el puerto de LECTURA COMPLETA TOMA valor uno para que el control principal envíe un cero al puerto LECTURA y la operación de lectura finalice.

Los datos de salida de la memoria RAM se deben conectar a los bloques que realizan conversión de complemento a dos a punto flotante, pero como esta salida corresponde a la concatenación de la parte real y la parte imaginaria de la FFT se debe utilizar un vector auxiliar para dividir de nuevo cada una de estas componentes, que después de ser divididas se pasan por separado a los bloques de conversión, uno para parte real y otro para parte imaginaria.

2.3.5 Bloque de conversión de complemento a dos a punto flotante.

Este bloque se implementa mediante un CORE IP que realiza diferentes operaciones con vectores en formato de punto flotante. En este caso para realizar la operación requerida el CORE IP se configura para que reciba como entrada un vector en punto fijo en representación de complemento a dos y entregue en la salida un vector de 32 bits en representación de punto flotante.

La salida de este bloque se conecta al bloque de multiplicación por sí mismo para empezar el proceso de obtención de magnitud de la FFT.

En la documentación del Xilinx CORE Generator y en el datasheet del CORE Floating point utilizado no se especifica el método por el cual se realiza esta conversión alguna de

las operaciones que se pueden efectuar mediante este bloque, sin embargo, sí se menciona cuáles son los pasos para generar el CORE según la operación que se quiere realizar.

A continuación se presenta el proceso para generar el CORE que realice la conversión de punto fijo a punto flotante.

En la figura 2.17 se muestra la primera ventana de configuración del CORE Floating point. Aquí se selecciona el tipo de operación que va a realizar el bloque. Para este caso seleccionamos una operación de Fixed to Float, es decir, conversión de punto fijo a punto flotante.

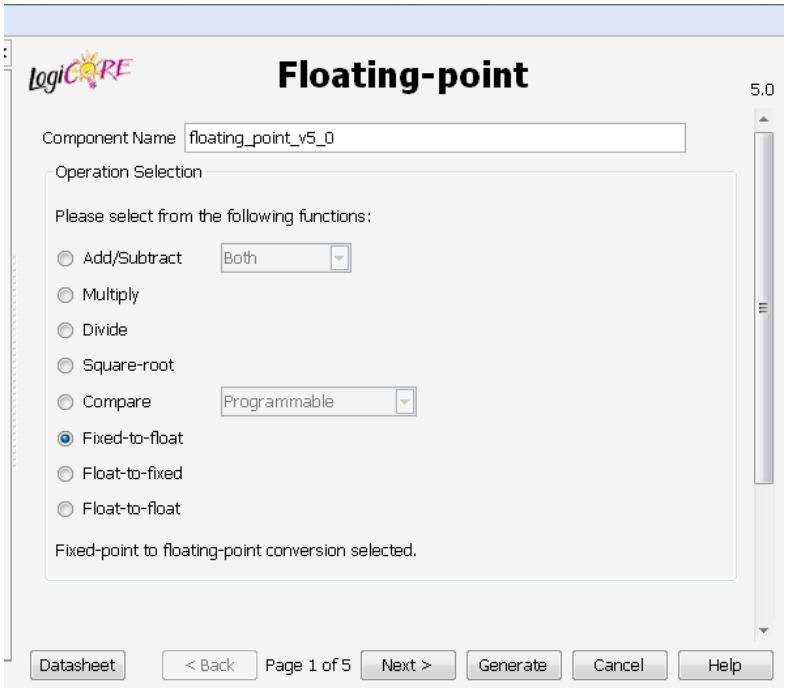

Figura 2.17. Generación del CORE Floating Point. Paso 1.

En la figura 2.18 se muestra el segundo paso para la generación del CORE Floating Point. Aquí se puede seleccionar la precisión de la representación de punto fijo de la entrada. Esta precisión se selecciona como custom con tamaño 25 bits para el entero y cero bits para la fracción.

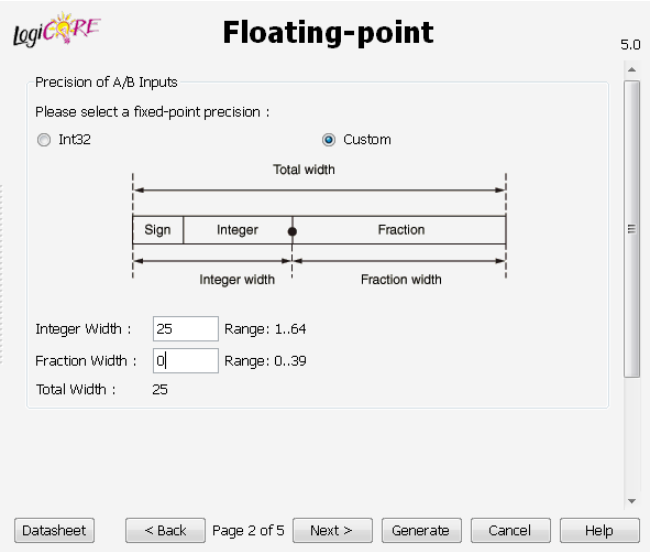

Figura 2.18. Generación del CORE Floating point. Paso 2.

En el tercer paso de configuración del CORE se selecciona la precisión del vector de punto flotante de salida, esta se elige en precisión simple. En la figura 2.19 se muestra esta selección.

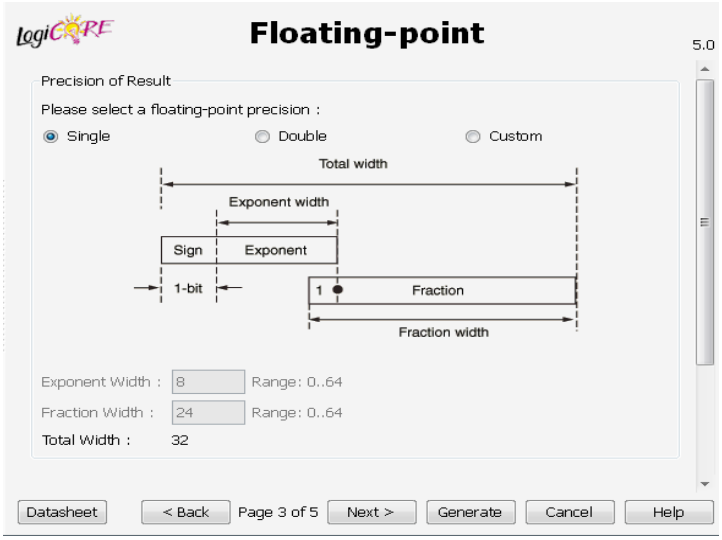

Figura 2.19. Generación del CORE Floating point. Paso 3.

El último paso es seleccionar puertos de control opcionales para el bloque del CORE. Aquí se seleccionan OPERATION_ND que recibe un uno para indicar al bloque que hay un nuevo dato a la entrada, RDY que se pone en alto cuando hay un dato valido a la salida del bloque, SCLR que es un reset síncrono y CE que habilita el reloj del bloque.

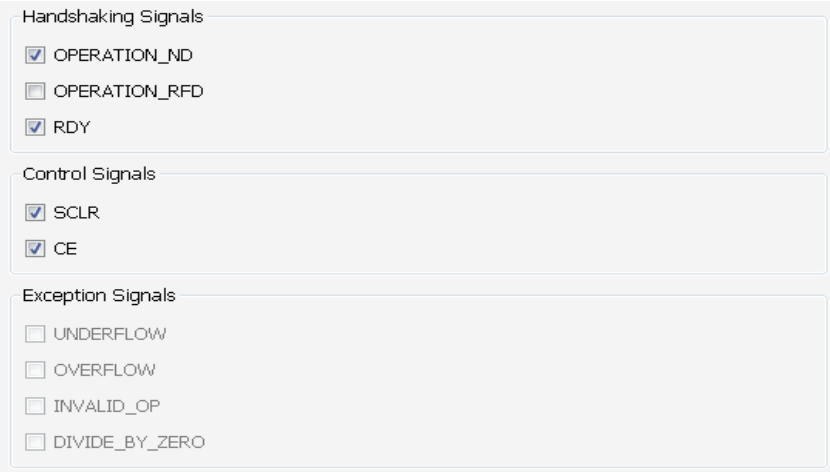

Figura 2.20. Generación del CORE Floating point. Paso 4.

2.3.6 Bloque de multiplicación por sí mismo.

El bloque de multiplicación por sí mismo se realiza mediante el CORE de Floating point explicado anteriormente. La entrada consiste en cada componente de la FFT, parte real y parte imaginaria, en punto flotante y la salida es un vector de 32 bits con el resultado de la operación.

El proceso para generar el CORE es idéntico al que se describió para el bloque anterior, exceptuando el primer paso donde se selecciona la operación a realizar, que en este caso se escoge en Multiply como se verifica en la figura 2.21.

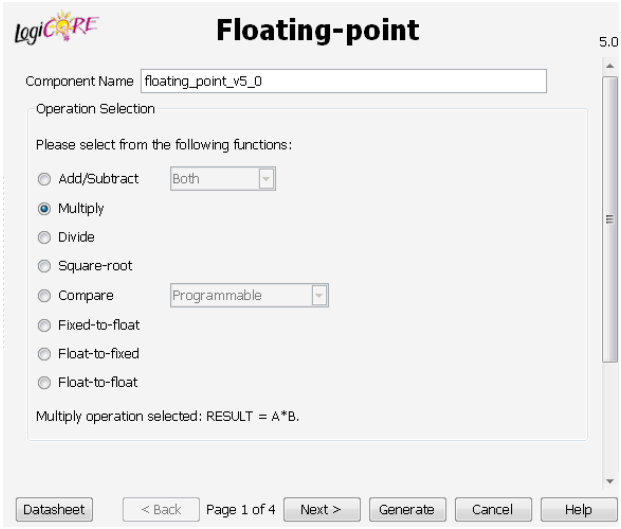

Figura 2.21. Generación del CORE Floating point. Multiplicación.

2.3.7 Bloque de suma.

Este bloque realiza la operación de suma entre las dos componentes ya elevadas al cuadrado. Su implementación se realiza utilizando el CORE de Floating point y los pasos para generar este CORE son los mismos que los del bloque de conversión de complemento a dos a punto flotante, la única diferencia consiste en el primer paso donde se especifica el tipo de operación que realiza el bloque, aqui se escoge Add o suma. Esto se puede verificar en la figura 2.22 donde se muestra la primera etapa de configuración del CORE para este módulo.

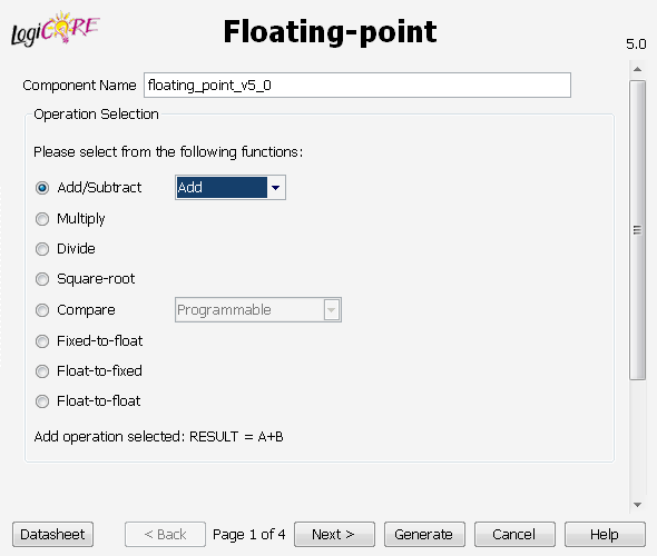

Figura 2.22. Generación del CORE Floating Point. Suma.

2.3.8 Bloque de raíz cuadrada.

El bloque de raíz cuadrada realiza esta operación sobre la suma de las componentes de la FFT una vez han sido elevadas al cuadrado. La entrada de este módulo corresponde a un vector de punto flotante de precisión simple que proviene de la salida del bloque de suma y la salida, que se conecta a una memoria RAM, es también un vector en punto flotante.

La implementación de este bloque también se realiza utilizando el CORE de Floating point que se empleó en los bloques de operaciones anteriores y su proceso de configuración es el mismo que se describió anteriormente en el bloque de conversión de complemento a dos a apunto flotante, la única diferencia, como en los casos anteriores, es que en la primer paso donde se define qué operación realiza el bloque, se especifica que se efectuara una operación de raíz cuadrada. En la figura 2.23 se muestra la configuración de este primer paso para generar el CORE de raíz cuadrada.

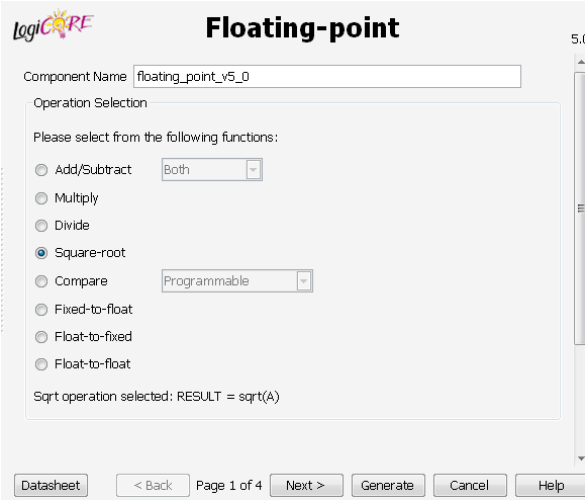

Figura 2.23. Generación del CORE Floating point. Raíz cuadrada.

En la figura 2.24 se muestra un esquema del componente generado utilizando el CORE Floating Point. A la derecha se muestran las salidas y a la izquierda las entradas del bloque.

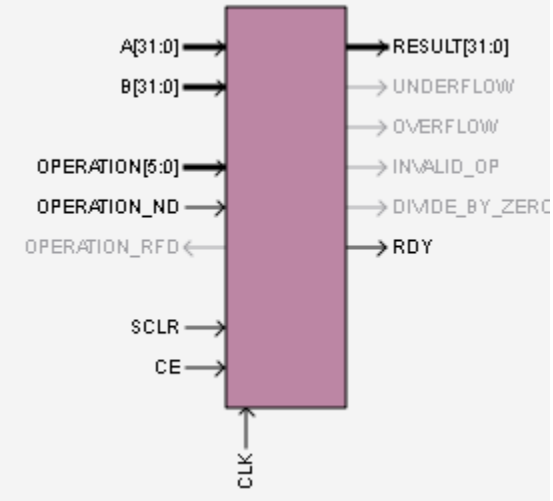

Figura 2.24. Bloque del CORE Floating point.

En la figura 2.25 se presenta un diagrama de flujo que ilustra el funcionamiento de todo el modulo funcional que realiza las operaciones sobre los datos de salida de la FFT, desde la conversión de complemento a dos a punto flotante, hasta la operación de raíz cuadrada. Se agrupan los componentes en un solo modulo para facilitar la explicación de su funcionamiento.

En la figura 2.25 se presenta una convención especial para las señales, los nombres terminados en ND corresponden, de acuerdo como se explicó en la sección de

descripción de implementación del bloque de conversión de complemento a dos a punto flotante, al puerto de entrada OPERATION_ND, el cual recibe un valor alto cuando se pone un nuevo dato a la entrada de un bloque de operación. Igualmente, los nombres de señales terminados con RDY corresponden al puerto de salida RDY, el cual toma un valor alto cuando existe un dato valido en la salida un bloque de operación.

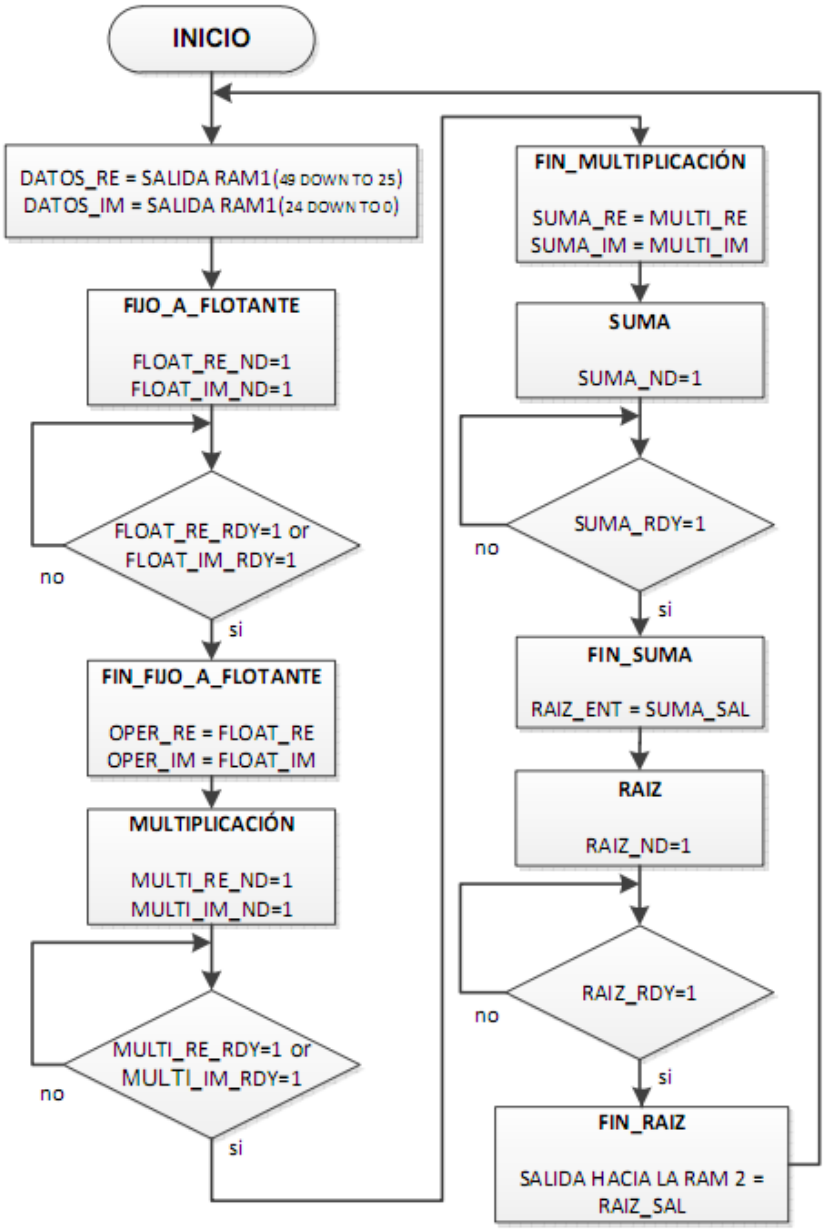

Figura 2.25. Diagrama de flujo, bloque de operaciones en punto flotante.

El proceso del módulo de operaciones en punto flotante comienza cuando se recibe un par de datos desde la memoria RAM que almacena las salidas de la FFT. Como se trata de dos componentes, parte real y parte imaginaria, hay dos bloques de conversión, uno

para cada componente. En este momento las señales ND de estos bloques de conversión se ponen en valor alto para indicar que existe un dato en la entrada.

Mientras cada entrada es procesada las señales de RDY de cada bloque permanecen en cero y cuando la conversión ha finalizado las señales de RDY toman valor uno, en este momento se pasa a la sección de FIN_FIJO_A_FLOTANTE, donde se asignan las salidas de este bloque a los operadores del bloque siguiente.

El siguiente paso es pasar los datos ya en punto flotante a los bloques de multiplicación. Cada bloque de MULTIPLICACIÓN recibe dos operadores, que en este caso son la misma componente, es decir, la parte real es la entrada de los dos puertos de bloque de multiplicación número uno y la parte imaginaria es la entrada de los dos puertos del bloque de multiplicación número dos. Así se consigue elevar cada componente al cuadrado.

Siguiendo la figura 2.25, se observa que igual que en el bloque anterior, cuando se recibe una entrada las señales ND se ponen en uno y para pasar a la sección de FIN_MULTIPLICACION se espera a que las señales RDY tomen valor uno, es decir, se espera a que las operaciones en el bloque hayan conluido y se tenga un dato valido en su salida.

En la sección FIN_MULTIPLICACION se asignan las salidas de los bloques de multiplicación a los operadores del bloque suma que es el siguiente bloque.

En el bloque suma se reciben dos operadores que son las salidas de los dos bloques de multiplicación. Al igual que en los bloques anteriores, la señal ND de este bloque se pone en uno para indicar que se tiene un nuevo dato a la entrada y se espera hasta que la señal RDY se ponga en alto, es decir cuando la operación se ha realizado, para pasar a la sección FIN_SUMA, donde se asigna la salida del bloque al operador del siguiente bloque.

El bloque que sigue es el de RAÍZ CUADRADA. El funcionamiento es el mismo que el de los bloques de operación descritos anteriormente. Primero, cuando se recibe un dato a la entrada la señal ND se pone en uno. Luego se examina el estado de la señal RDY para verificar que la operación ha finalizado y si es así, se asigna la salida del bloque a la entrada del bloque siguiente que es la memoria RAM 2.

Un punto importante que es necesario resaltar es que las señales de ND que se mencionan a lo largo de la descripción del módulo de operaciones en punto flotante, deben ser mantenidas en alto durante mínimo 20ns para que el bloque respectivo sea notificado de que existe un dato de entrada valido.

2.3.9 Bloque de RAM 2.

Este bloque recibe el dato de magnitud y lo almacena para luego ser transmitido al computador mediante el bloque de transmisión RS232.

El funcionamiento de este componente es el mismo que el del bloque RAM que se ubica a la salida de la FFT, el cual fue descrito anteriormente. La diferencia es que la señal que habilita la escritura sobre la RAM 2 no es la señal dv como en ese caso sino que se trata de una señal asociada al puerto RDY del bloque que realiza la operación de raíz cuadradam y la señal que habilita la lectura es una señal proveniente del bloque RS232 que indica que se va a transmitir un dato. Para más detalles sobre el funcionamiento del bloque RAM se puede consultar la figura 2.16 y la descripción relacionada a esta figura.

2.3.10 Bloque de transmisión RS232.

Este bloque se encarga de transmitir al computador los datos de magnitud de la FFT almacenados en la RAM 2. Su implementación de realiza mediante un registro 11 bits de acuerdo al protocolo de transmisión serial, donde ocho bits son para datos y los tres bits restantes son de control. Este registro ha sido encapsulado en un bloque que permita hacer el control de la transmisión y la configuración de la velocidad de transmisión.

El bloque implementado presenta una entrada para un vector de 32 bits que corresponde a la salida de la memoria RAM 2. Este vector es divido en cuatro vectores, cada uno de ocho bits que corresponden a los datos para un registro en transmisión serial. A cada vector de ocho bits se le asignan los bits de control para completar el registro de 11 bits y se transmiten uno a uno hacia el computador.

2.3.11 Bloque aplicación PC.

Este bloque corresponde a una Interfaz gráfica de Usuario o GUI desarrollada en MATLAB que se encarga de recibir los datos de magnitud de la FFT provenientes de la FPGA, normalizarlos teniendo en cuenta el tamaño de la transformada y el factor de conversión del ADC, para realizar una curva de voltaje contra frecuencia de la señal de entrada del sistema de análisis de señales de tipo poliscopio.

El programa MATLAB permite también realizar aplicaciones de poliscopio mediante un menú llamado Aplicación e incorpora una barra de herramientas con botones para zoom, leyenda y cursor, que facilitan la inspección de la gráfica presentada.

Considerando que es importante comparar la señal de entrada al circuito de prueba con la señal de salida de este circuito, la aplicación desarrollada permite visualizar las dos graficas en una misma interfaz para mayor comodidad del usuario.

Es importante aclarar que el sistema de análisis de señales de poliscopio no genera ningún tipo de señal, ni a la entrada ni a la salida del circuito de prueba, tampoco se encarga de realizar la lectura o adecuación de estas señales. La función del sistema de análisis de señales de poliscopio es realizar el análisis de una señal que se le entregue con el fin de obtener su curva de voltaje contra frecuencia.

En la figura 2.26 se presenta una captura de pantalla de la aplicación generada en MATLAB. En esta figura se pueden observar las características antes mencionadas.

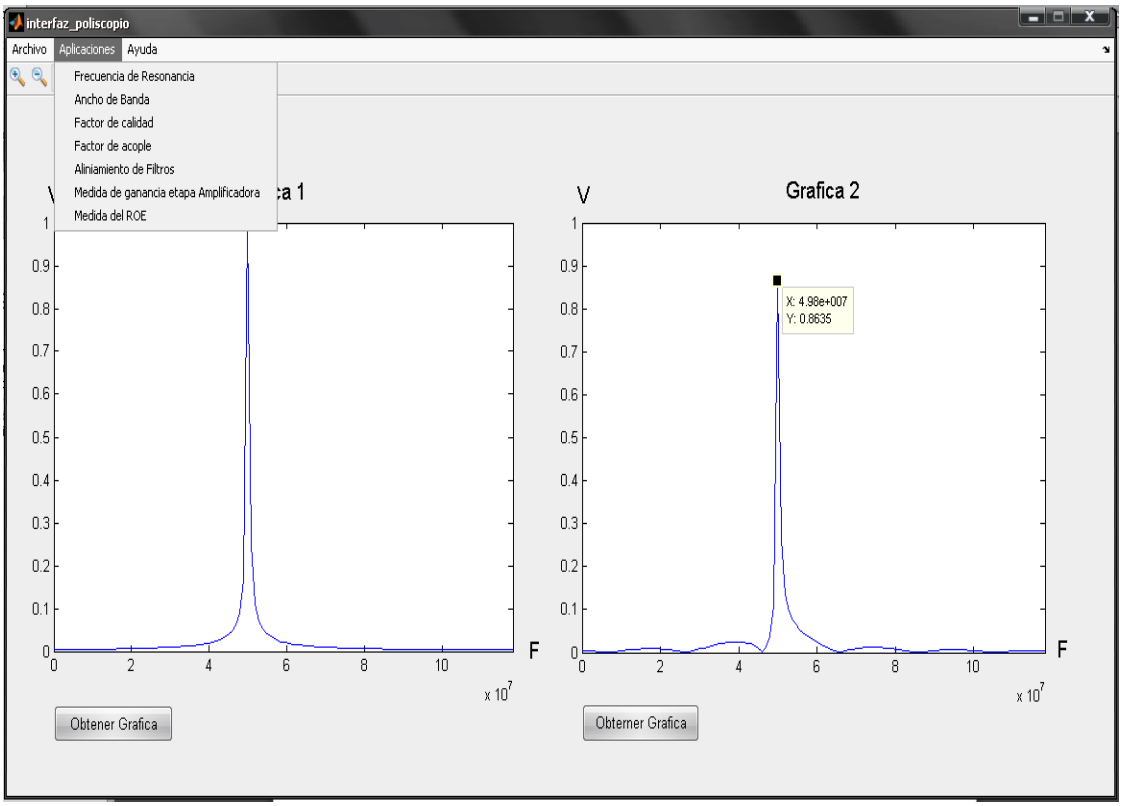

Figura 2.26. Sistema de análisis de señales de tipo poliscopio. Aplicación en MATLAB.

Capítulo III Pruebas y resultados.

En este capítulo se presentarán las pruebas efectuadas sobre cada uno de los módulos del prototipo del sistema de análisis de señales de tipo poliscopio y sobre cada una de las estructuras funcionales que se definirán con el fin de facilitar las pruebas y depuración del prototipo. También se presentaran y se discutirán los resultados obtenidos y se realizaran las observaciones pertinentes.

3.1 Plan de pruebas.

Como se explicó en el capítulo 2, la implementación del prototipo se realiza teniendo en cuenta el diagrama modular presentado en el diseño, asimismo el plan de pruebas se define sobre cada uno de los módulos del prototipo. Sin embargo, es necesario agrupar varios componentes de manera conveniente cuando se realizan los ensayos, por ejemplo, sobre el conversor analógico a digital, sobre el bloque que realiza la transformada rápida de Fourier o sobre el módulo RS232 de trasmisión de datos hacia el computador.

El plan de pruebas definido consta de las etapas que se mencionan a continuación:

- Simulaciones sobre cada módulo funcional una vez concluida la implementación en VHDL. Estas simulaciones se realizan utilizando la herramienta ISIM que provee el entorno integrado de diseño ISE de Xilinx y los archivos test brench, donde se definen las señales que sirven de estímulo y la duración de cada estimulo. Mediante los resultados arrojados en estas simulaciones se comprueba el diagrama de tiempos que necesita cada componente y las salidas que se obtienen de acuerdo a las entradas que se asignan en cada caso.

- Simulaciones conjuntas sobre bloques agrupados donde se puede comprobar la interconexión entre bloques, el tiempo de respuesta de cada bloque y las señales que se obtienen a la salida de los componentes agrupados de manera global. Estas simulaciones también se realizan con ayuda de los archivos test brench definidos para estos casos y de igual manera se utiliza el ISE de Xilinx.

- Comparación de los resultados obtenidos probando los módulos del sistema de análisis de señales tipo poliscopio y los resultados a partir de otros sistemas como por ejemplo diseños de prueba de Xilinx y las gráficas de la transformada de una señal conocida obtenida utilizando MATLAB.

- Pruebas Físicas sobre los módulos implementados y sobre el sistema en General.

3.2 Pruebas sobre el prototipo del sistema de análisis de señales de poliscopio.

A continuación se presentaran las pruebas realizadas sobre cada uno de los módulos implementados y sobre las estructuras funcionales definidas.

3.2.1 Pruebas de simulación sobre los módulos implementados.

3.2.1.1 Pruebas sobre el bloque ADC.

Para efectuar las pruebas del conversor analógico a digital LTC1407A y del preamplificador LTC6912 de la Spartan 3A Starter kit se realizó una simulación en ISIM del ISE 12.3 de Xilinx, con el fin de verificar el diagrama de tiempos especificado en el datasheet del kit de desarrollo Spartan 3A [13]. En la figura 3.1 se puede observar el diagrama de tiempos especificado en este datasheet y el diagrama de tiempos que arroja la simulación. Aquí se puede ver como se corresponden con gran exactitud en las señales SPI_MOSI, SPI_MISO, AMP_CS, AD_CONV, AMP_SHDN, SPI_SCK, que son las señales que controlan la configuración del ADC y del preamplificador y también controlan los ciclos de adquisición de datos por parte del conversor.

La señal btn corresponde a un estímulo externo, en nuestro caso el pulso generado por un botón, la primera vez que se genera este pulso inicia la configuración del preamplificador poniendo la señal AMP_CS en nivel bajo y 40ns después aparece el primer ciclo de reloj en SPI_SCK. La duración de este ciclo de reloj es de 160 ns. Luego de esto, se carga la configuración de ganancia del preamplificador, la cual es enviada a través de la señal SPI_MOSI y en nuestro caso se trata del vector 00010001 y corresponde a una ganancia de -1, que permite niveles de voltaje en la entrada desde 0.4V hasta 2.9V. Con esto concluye la configuración del preamplificador y la señal AMP_CS regresa a nivel alto.

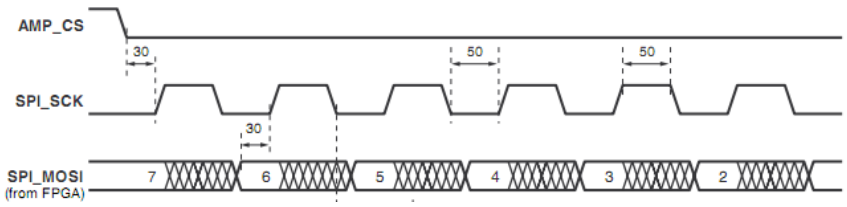

Figura 3.1. Diagrama de tiempos preamplificador LTC6912.

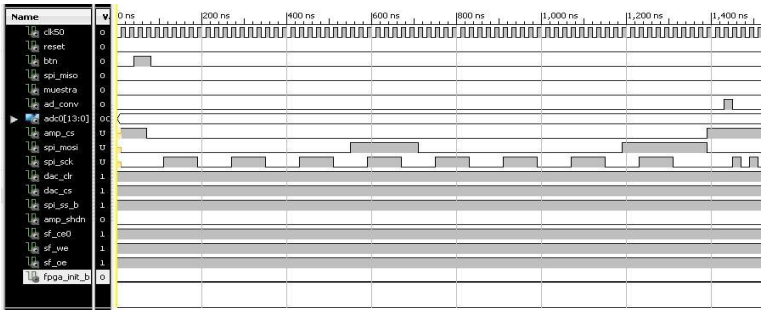

Figura 3.2. Diagrama de tiempos de la simulación del preamplificador.

Después de configurar el preamplificador, la señal AD_CONV es activada durante un ciclo del reloj de 50MHZ de la FPGA, es decir durante 40ns. Esta señal indica que se inicia el ciclo de adquisición de datos por parte del conversor analógico a digital. Es de resaltar que una vez configurado el preamplificador la señal SPI_SCK se configura a 40ns por ciclo con el fin de obtener la máxima frecuencia de muestreo posible en el ADC teniendo en cuenta el reloj de 50MHZ con el que cuenta el kit de desarrollo Spartan 3A. Por otra parte cuando se ha concluido la adquisición de datos, la señal muestra se pone en alto indicando que el ADC ha tomado una muestra válida para ser almacenada, esta muestra se pasa a la FPGA de manera serial mediante la señal SPI_MISO.

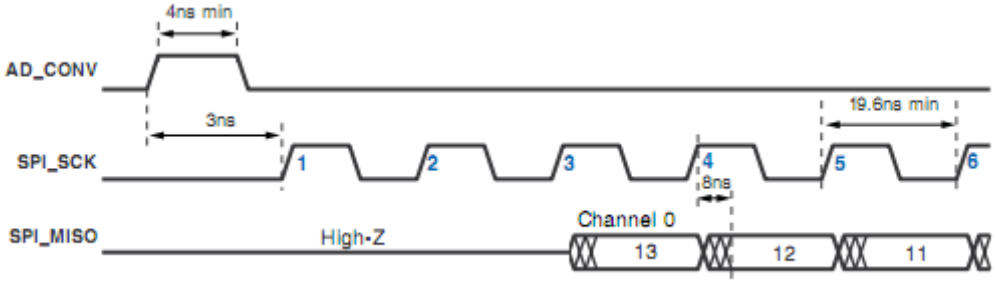

Figura 3.3. Diagrama de tiempos ADC LTC1407A.

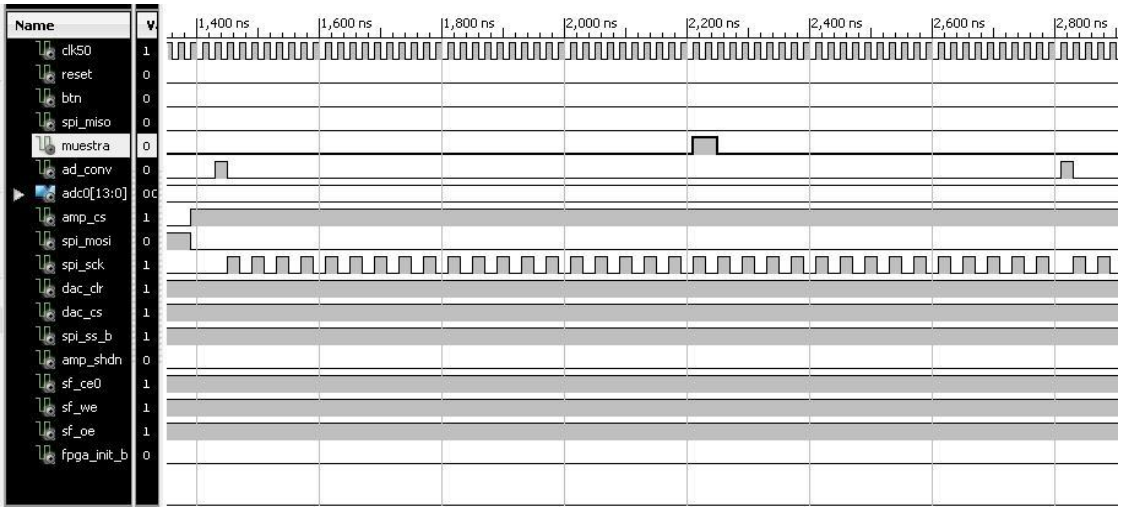

Figura 3.4. Diagrama de tiempos de la simulación del conversor analógico a digital.

3.2.1.2 Pruebas sobre la memoria FIFO.

La simulación de la memoria FIFO se realiza cargando un archivo .DAT con valores preestablecidos en el puerto de entrada durante el ciclo de escritura y se verifica que en el puerto de salida aparezcan estos valores durante el ciclo de lectura. Cuando la memoria se llena completamente la señal FULL se pone en alto. Solo es posible escribir datos cuando la señal WRITE_EN se pone en alto y solo es posible leer los datos cuando la señal READ_EN se pone en alto. Cuando los datos son leídos completamente la señal EMPTY se pone en ALTO. LA memoria se fija en un tamaño de 16 registros para facilitar las pruebas y la visualización del diagrama de tiempos. Los datos cargados en la memoria mediante el archivo .DAT son los siguientes: 1,-7986,3554,6404,-6404,-3554,7986,0 -7986,3554,6404,-6404,-3554,7986,0,-7986. En la Figura 3.5 se muestra el diagrama de tiempos del ciclo de escritura y en la Figura 3.6 se muestra el diagrama de tiempos del ciclo de lectura de la memoria FIFO.

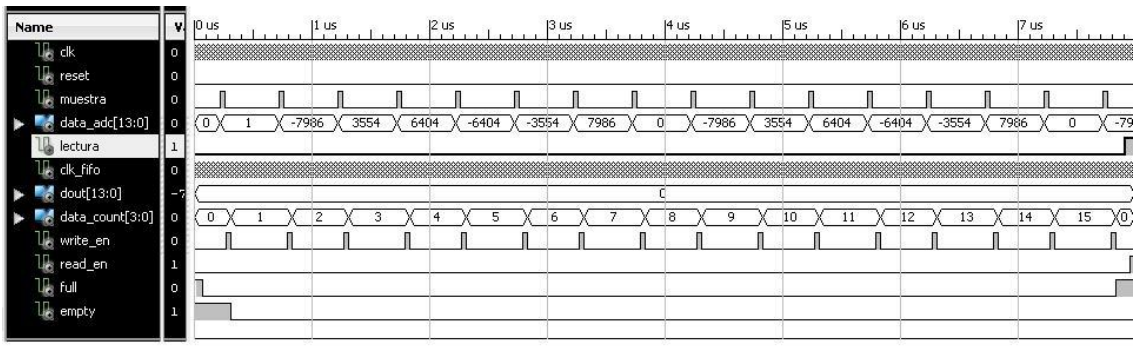

Figura 3.5. Diagrama de tiempos del ciclo de escritura de la memoria FIFO.

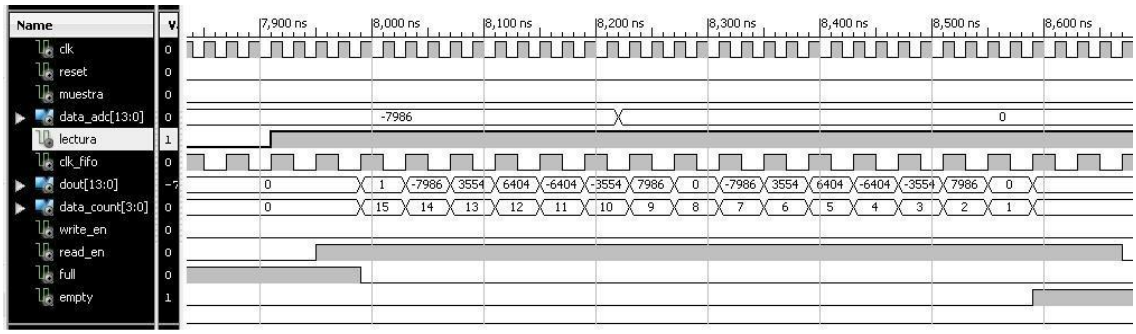

Figura 3.6. Diagrama de tiempos del ciclo de lectura de la memoria FIFO.

3.2.1.3 Pruebas sobre la FFT.

Para realizar la prueba de simulación sobre el bloque que realiza la FFT se define esta con 16 muestras, luego se carga un archivo .DAT con las muestras que se quieren llevar al dominio de la frecuencia y se comparan los valores que se obtienen en la salida del bloque con los valores que arroja realizar una transformada rápida de Fourier en

MATLAB. Los valores de la salida del bloque FFT se guardan en un archivo .DAT para la parte real y un archivo .DAT para la parte imaginaria, para así poder llevarlos a MATLAB.

En la figura 3.7 se observa el diagrama de tiempos de la FFT, aquí se puede ver las señales que se utilizan en cada puerto. El reloj con el que funciona la FFT es de 25MHz. La señal CE_FFT tiene que estar en alto para habilitar cualquier operación de la FFT. La configuración se inicia llevando la señal INICIAR_FFT a un nivel alto, luego, según recomendación del datasheet del CORE FFT [13], se borran todos los registros del bloque. A continuación la señal START_FFT se pone en alto por un ciclo de reloj y con esto concluye la configuración. Dado el caso en que se tuvieran que utilizar señales de escalamiento, estas deberían introducirse en este periodo de configuración, estas señales serian el vector de escalamiento y la señal que habilita el ingreso del vector de escalamiento.

Después de la etapa de configuración la señal RFD_FFT se pone en nivel alto indicando que se pueden cargar los datos en el bloque FFT. Una vez los datos han sido cargados en la FFT la señal BUSY_FFT se pone en nivel alto indicando que el bloque está ocupado y que no se pueden realizar operaciones hasta que esta señal se ponga de nuevo en nivel bajo.

En la Figura 3.8 se presenta el diagrama de tiempos del datasheet del Core FFT. En esta figura se puede observar el comportamiento anteriormente descrito y se verifica que el funcionamiento de la implementación que se realizo está acorde con lo que se demanda en el Datasheet.

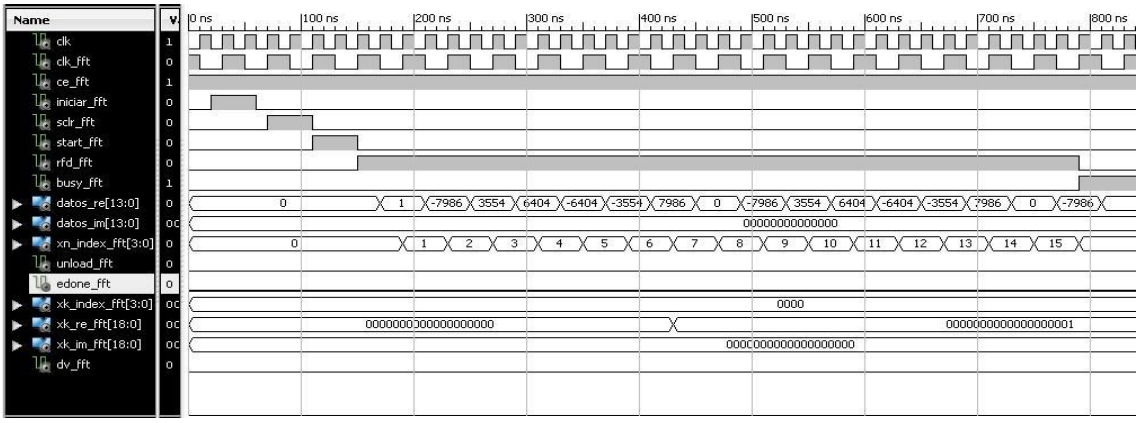

Figura 3.7 Configuración e Inicio del bloque FFT.

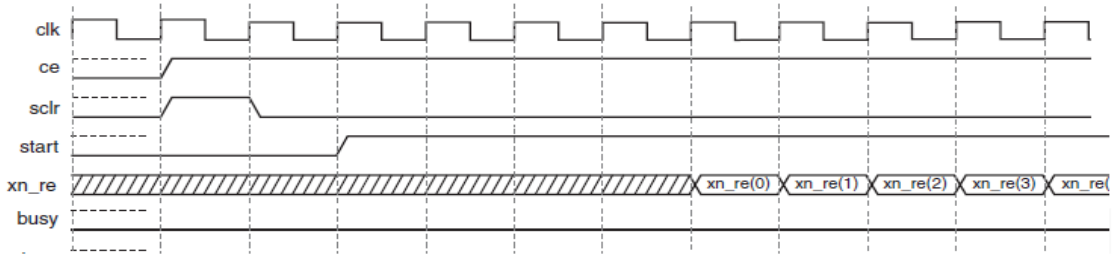

Figura 3.8. Diagrama de tiempos del datasheet CORE FFT.

Cuando el Bloque FFT ha terminado de procesar las muestras la seña EDONE_FFT se pone a nivel alto durante un ciclo de reloj indicando que un ciclo después los datos de salida de la FFT pueden ser descargados. La señal busy vuelve a nivel bajo y para obtener los datos es necesario poner la señal UNLOAD_FFT en alto. Los datos empiezan a aparecer con cada ciclo de reloj en los puertos, parte real y parte imaginaria, y solo serán válidos si la señal DV_FFT está en nivel alto. En la Figura 3.9 se puede observar lo anteriormente descrito.

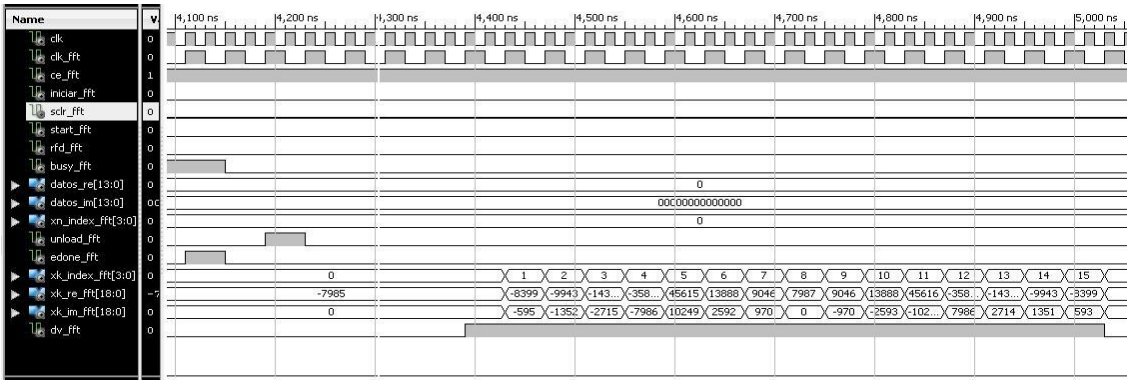

Figura 3.9. Diagrama de Tiempos descarga de datos bloque FFT.

En la tabla 3.1 se presentan la parte real y la parte imaginaria de la transformada rápida de Fourier con los datos que se utilizaron en la simulación y los datos que se obtienen al realizar una FFT en MATLAB sobre las mismas muestras.

Numero de Muestra	Dato Entrada	Parte Real Simulación	Parte Imaginaria Simulación	Parte Real MATLAB	Parte Imaginaria MATLAB
1	1	-7985	0	-7985	0
2	-7986	-8399	-595	-8397	-593
3	3554	-9943	-1352	-9942	-1351
4	6404	-14319	-2715	-14318	-2713
5	-6404	-35887	-7986	-35887	-7986
6	-3554	45615	10249	45616	10250
7	7986	13888	2592	13888	2593
8	0	9046	970	9047	970
9	-7986	7987	0	7987	0
10	3554	9046	-970	9047	-970
11	6404	13888	-2593	13888	-2593
12	-6404	45616	-10250	45161	-10250
13	-3554	-35887	7986	-35887	7986
14	7986	-14318	2714	-14318	2713
15	0	-9943	1351	-9942	1351
16	-7986	-8399	593	-8397	593

Tabla 3.1 Resultados de la simulación de la FFT y resultados de FFT en MATLAB.

De la tabla 3.1 podemos observar que prácticamente los resultados son los mismos obtenido en la simulación del Core FFT.

Para obtener la FFT de las muestras utilizadas en la simulación en MATLAB se crea un archivo .m y dentro del archivo se escribe:

X= [1 -7986 3554 6404 -6404 -3554 7986 0 -7986 3554 6404 -6404 -3554 7986 0 -7986];

Y=FFT(X)

3.2.1.4 Pruebas sobre el modulo funcional de operaciones en C2 [19] y punto flotante.

Como se especificó en el capítulo anterior, para realizar una curva voltaje contra frecuencia se necesita el módulo de la señal en el dominio de la frecuencia, así que en el diseño modular del sistema se definen una serie de bloques que realizan las operaciones necesarias para obtener este módulo. Para verificar su funcionamiento estos bloques se agrupan en una estructura funcional con el fin de facilitar las pruebas y la visualización de la información que se obtiene.

Los bloques agrupados en este modelo funcional se muestran en la figura 3.10, aquí se puede observar que se agruparon los siguientes bloques:

- Los bloques que transforman de C2 a punto flotante.
- Los bloques de multiplicación en punto flotante.
- El bloque de suma.
- El bloque que obtiene la raíz cuadrada.

Figura 3.10. Modulo para obtener magnitud de la señal en el dominio de la frecuencia.

La prueba sobre este módulo consiste en una simulación que permite cargar una serie de números en complemento a dos mediante un archivo .DAT y procesarlos a través del bloque conjunto, exportar el resultado a otro archivo .DAT y este archivo se lleva a MATLAB para realizar una conversión de estos valores a números en formato de Entero con signo.

[19] C2: Complemento a dos

Por otro lado, en MATLAB se realiza el mismo proceso del bloque conjunto, utilizando un script que obtiene la magnitud de los valores en el dominio de la frecuencia contenidos en el archivo .DAT de entrada, mediante la instrucción abs(); y al final se presentan los dos resultados.

En la tabla 3.2 se muestran los datos correspondientes al archivo .DAT de entrada. En la Tabla 3.3 se exponen los resultados obtenidos tanto en MATLAB como en la simulación del bloque funcional, se puede observar que los valores obtenidos a través de la implementación en VHDL son similares a los valores obtenidos utilizando MATLAB.

Parte real	Parte imaginaría	Representación en C2 parte real	Representación en C2 parte Imaginaria
-7985	0	1111110000011001111	0000000000000000
-8399	-595	1111011111100110001	1111111110110101101
-9943	-1352	1111011100100101001	1111111101010111000
-14319	-2715	1111001000000010001	1111111010101100101
-35887	-7986	1101110011111010001	1111100000011001110
45615	10249	0010110010001011111	0000101000000001001
2592i	2592	0000110110010000000	0000001010001000000
9046	970	0000100011010101110	0000000011110010100
7987	0	0000011111001100111	0000000000000000000
9046	-970	0000100011010101110	1111111100001101110
13888	-2593	0000110110010000000	1111110101110111111
45616	-10250	0010110010001100000	1111010111111101110
-35887	7986	1101110011111010001	0000011111001100110
-14318	2714	1111001000000010010	0000001010100110100
-9943	1351	1111011001001010001	0000000101010001111
-8399	593	1111011111001100001	0000000010010100001

Tabla 3.2. Datos de entrada para prueba del bloque de operaciones en punto flotante.

En la figura 3.11 se observa el diagrama de tiempos del bloque de operaciones en punto flotante, la señal listo se pone en nivel alto cuando el bloque ha terminado de realizar las operaciones sobre un dato de entrada, este dato de entrada corresponde a dos valores en complemento a dos, una parte real y una parte imaginaria. Aquí se puede comprobar que el tiempo de procesamiento para cada dato de entrada es de 1 microsegundo.

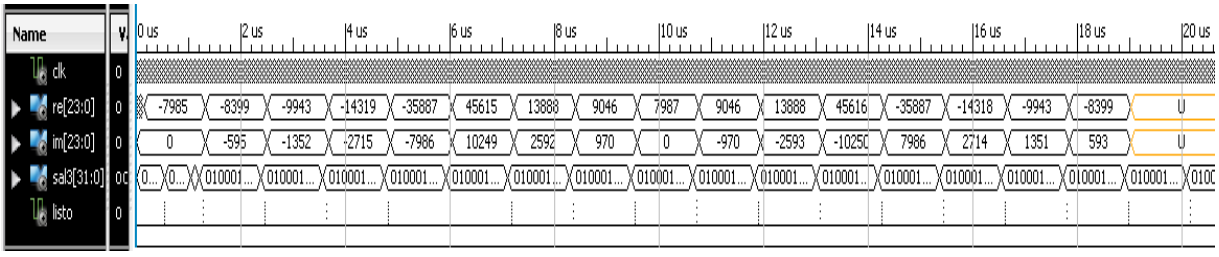

Figura 3.11. Diagrama de tiempos del bloque de operaciones en punto flotante.

El Script de MATLAB que convierte punto flotante a entero con signo es el siguiente:

```
flotante='01000101111110011000100000000000010001100000001110010000001100100100011000011100110010011111111001000110011000111011100001111100010001110000111100111001101010101000111001101101010000000111001010001100101110010111000101111111001101100000000000100011000001110001001110110111001000110010111001011111111111100101000111001101101010000010110101001000111000011111001110011010101010000110011000111011001111001110011000100011000011100110010010111010001000110000000111000111010001111001'
```

```
f=length(flotante);
t= f/32;
for r=1:t
        c=0;
        for r1=((32*(r-1))+1):32*r,
                if c==0,
                        x1=flotante(r1);
                        c=1;
                else
                        x1 = [x1, flotante(r1)];
                end;
        end;
x(1,r)=typecast( uint32( bin2dec(x1) ), 'single');
end;
```

El Script obtención de magnitud en MATLAB es el siguiente:

```
x=[-7985 -8399-595i -9943-1352i -14319-2715i -35887-7986i 45615+10249i 13888+2592i
9046+970i 7987 9046-970i 13888-2593i 45616-10250i -35887+7986i -14318+2714i -
9943+1351i -8399+593i];
y1=abs(x)
```

3.2.1.5 Pruebas sobre el módulo de trasmisión RS232.

Para probar el funcionamiento del módulo de transmisión RS232 se realiza un código en VHDL que trasmita una secuencia de números, en el cual se pueden observar las señales y el diagrama de tiempos que maneja el bloque. En la figura 3.12 se muestra el resultado de la simulación en un diagrama de tiempo, aquí se observa que las señal rs232_dce_tx trasmite el numero 4 cuya representación en binario es 100, así que, de acuerdo al protocolo serial del rs232, la señal de trasmisión está en un valor alto y cambia de estado cuando se empieza a trasmitir, en ese momento se envía el bit start y a continuación los 8 bits de datos, desde el LSB hasta el MSB, en este caso como lo muestra la figura 3.12, se envía el número 4, es decir 0000100 en binario, pero como se especifica en el

protocolo de trasmisión para el rs232 se envía primero el ultimo bit y en último lugar el primer bit, así que el orden es 00100000. Luego de enviar los datos se envía el bit de stop que es un 1.

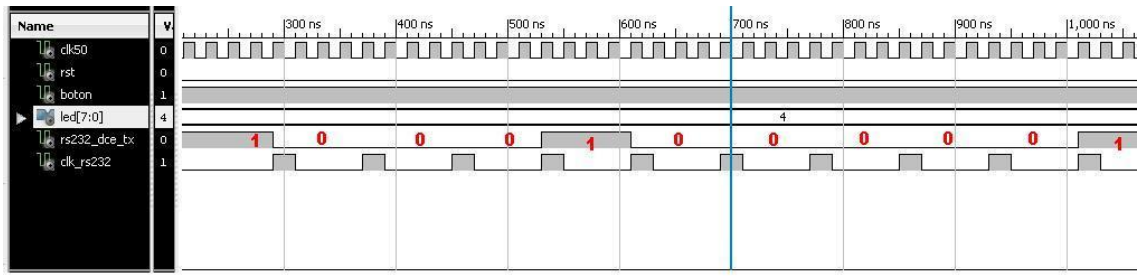

Figura 3.12 Simulación sobre el módulo de trasmisión serial.

3.2.2 Pruebas físicas.

3.2.2.1 Prueba sobre el ADC.

Para esta prueba se dispuso de un generador de señales conectado en una de las entradas del conversor análogo-digital y mediante una implementación del conversor digital-analógico se obtiene la señal reconstruida en un osciloscopio. El esquema de conexión se muestra en la figura 3.13.

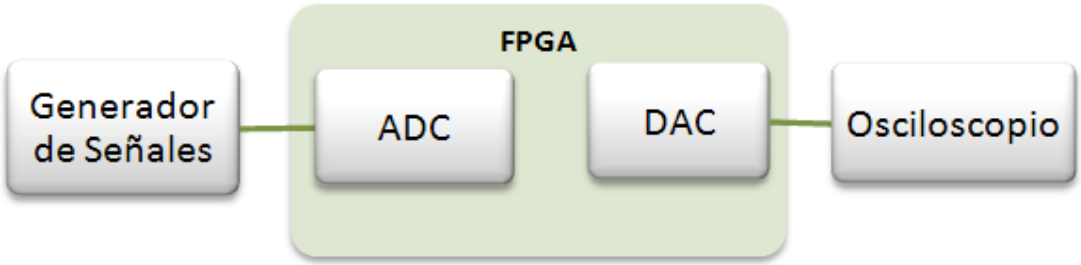

Figura 3.13 Esquema de conexión para prueba del ADC.

En las figura 3.14 se muestra el generador y la configuración puesta, en la figura 3.15 se muestra en el osciloscopio, la señal de entrada al ADC, en la figura 3.16 se muestra en el osciloscopio la señal de salida del DAC. El generador está ajustado en 1 voltio pico, la frecuencia se varía desde 0Hz hasta 200Khz y el osciloscopio muestra una señal de 0.8 voltios a estas frecuencias. Se observa que hay una leve disminución en el voltaje de la señal que muestra el osciloscopio, esto se debe a que el ADC y el DAC además de tener diferentes valores de voltaje de referencia (El ADC tiene 1.65V y el DAC 3.3V) tienen también diferente resolución, la del ADC es de 14bits y la del DAC es de 12bits. Es

importante señalar que la entrada del ADC debe estar entre 0.4 V y 2.9 V pues estos son los niveles de voltaje que se especifican en el Datasheet para este dispositivo de referencia LTC1407A.

Figura 3.14. Configuración.

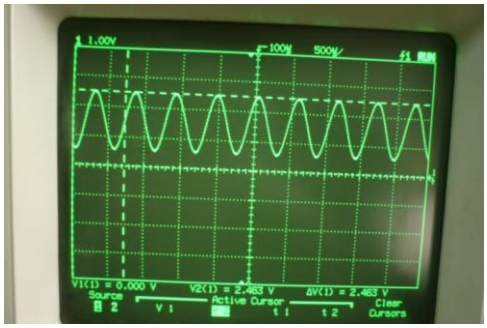

Figura 3.15. Señal de entrada al ADC de la FPGA.

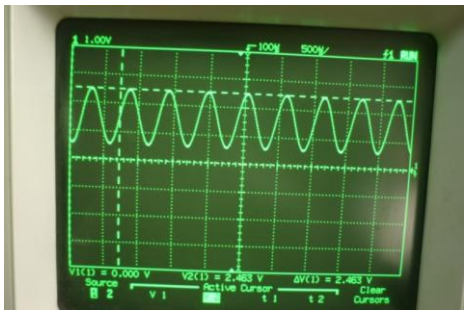

Figura 3.16. Señal de salida del DAC de la FPGA.

3.2.2.2 Pruebas sobre la memoria FIFO.

Para esta prueba se define una memoria FIFO de 16 registros, es necesario integrar una memoria ROM y el módulo de trasmisión RS232. Se implementa en VHDL un control que

sea capaz de cargar un archivo .COE[20] en la memoria ROM y que pase los valores que este archivo contiene uno a uno en la memoria FIFO para que una vez ésta se ha llenado, se trasmitan al computador los valores almacenados en ella utilizando el módulo de trasmisión RS232. En el computador se verifica que los registros trasmitidos son los mismos valores contenidos en el archivo .COE que se carga en la memoria ROM.

En la figura 3.17 se muestra un esquema de la conexión de componentes de la implementación que se realiza para esta prueba. En la figura 3.18 observa una captura de pantalla del archivo .COE, aquí se pueden ver los valores que este archivo contiene. En la figura 3.19 se observa una captura de pantalla de los valores recibidos en el computador mediante el software de monitoreo de puerto serie.

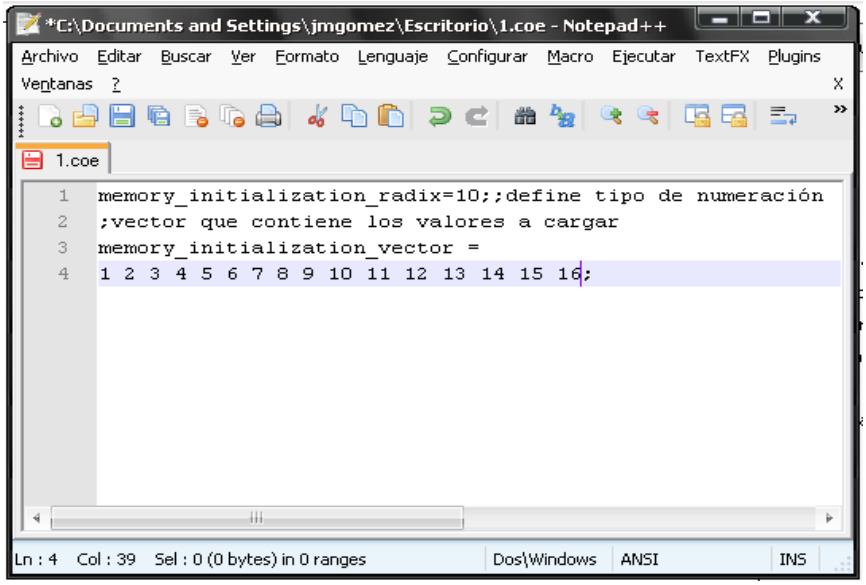

Figura 3.17. Archivo .COE para prueba de la FIFO.

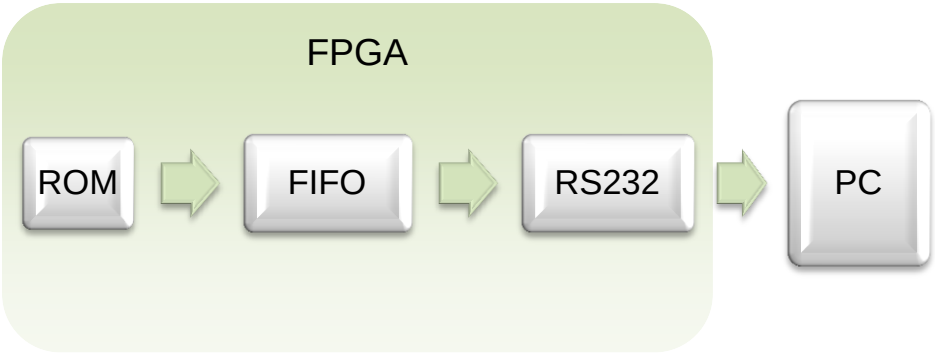

Figura 3.18. Esquema de conexión para prueba sobre memoria FIFO.

[20] la extensión de archivo .COE se utiliza para los archivos de coeficientes y valores predeterminad en diseños de lógica digital, que son eficientes en el uso de dispositivos silicio para su adecuado desempeño.

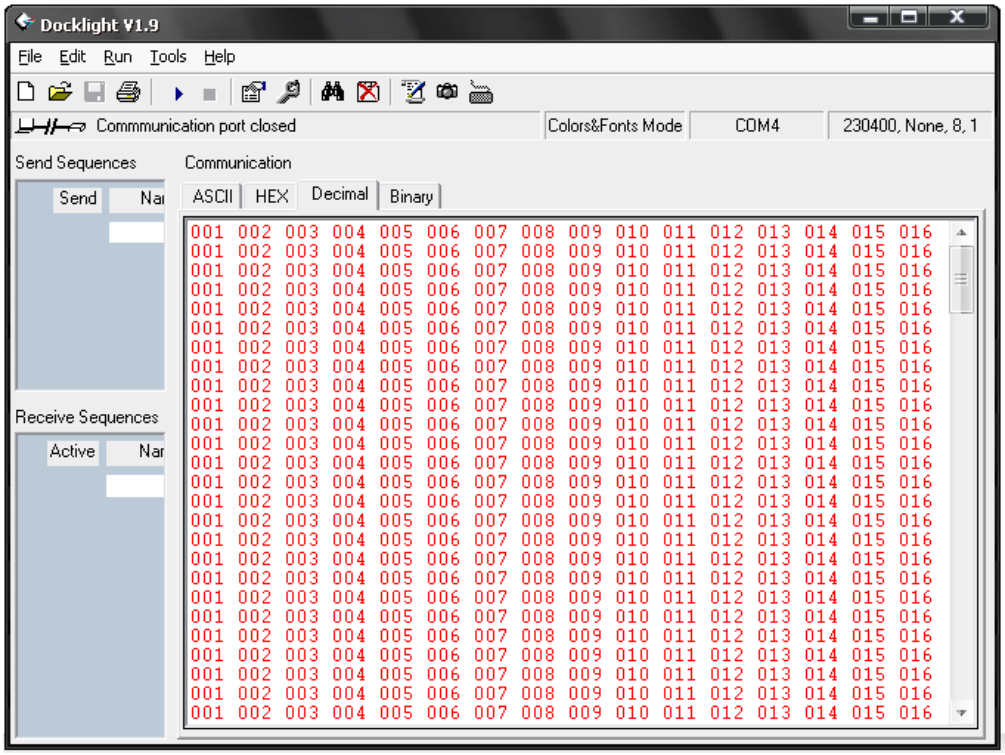

Figura 3.19. Valores recibidos en el computador desde la memoria FIFO implementada en la FPGA.

3.2.2.3 Pruebas sobre FFT.

Para efectuar las pruebas sobre el bloque que realiza la trasformada rápida de Fourier se realiza una implementación en VHDL de un CORE de la FFT con un tamaño definido en 16 muestras. Se cargan los valores que la FFT contendrá en una memoria ROM mediante un archivo .COE, estos valores se pasan a una memoria FIFO pues se requiere que el bloque de la FFT reciba las muestras secuencialmente. La salida de la FFT corresponde a una parte real y a una parte imaginaria, estas salidas se pasan a una memoria RAM para después ser trasmitidas al computador utilizando el módulo de trasmisión serial RS232. En el computador, se obtiene su magnitud utilizando MATLAB y se compara el resultado con la magnitud que se consigue al procesar los mismos valores de entrada contenidos en el archivo .COE utilizando un script de MATLAB que obtiene la FFT de estos valores y su magnitud.

En la Tabla 3.3 se recogen los resultados obtenidos en la prueba anteriormente descrita. En la primera columna de esta tabla están las muestras de entrada contenidas en el archivo .COE, en la segunda columna se muestran los valores que se trasmiten al computador mediante la interfaz RS232. En la tercera columna están los valores de magnitud para cada par de salidas de la FFT obtenidos desde la implementación en la FPGA. En la cuarta columna se presentan los valores de magnitud para los mismas muestras de entrada contenidas en el archivo .COE pero procesados completamente utilizando MATLAB. En la Tabla se puede verificar que los valores obtenidos mediante la

implementación en la FPGA de la FFT son muy cercanos a los valores obtenidos utilizando MATLAB.

Dato entrada	Salida FFT en FPGA	Magnitud salida FFT en FPGA	Magnitud FFT en MATLAB
0	-7986+ 0i	7986	7986
-7986	-7987+ 2667i	8420,51412	8419,22295
3554	-7986+ 6075i	10034,0331	10034,1366
6404	-7986+ 12191i	14573,8354	14573,4381
-6404	-7986+ 35888i	36765,8094	36765,8094
-3554	-7986 - 46067i	46754,0874	46752,633
7986	-7986 - 11653i	14126,8753	14126,7343
0	-7985 - 4359i	9097,31312	9097,77453
-7986	-7986 + 0i	7986	7986
3554	-7985+ 4359i	9097,31312	9097,77453
6404	-7986 + 11653i	14126,8753	14126,7343
-6404	-7986 + 46067i	46754,0874	46752,633
-3554	-7986 - 35888i	36765,8094	36765,8094
7986	-7986 - 12191i	14573,8354	14573,4381
0	-7986 - 6075i	10034,0331	10034,1366
-7986	-7987 - 2667i	8420,51412	8419,22295

Tabla 3.3. Resultados obtenidos en la prueba de la FFT en la FPGA y resultados obtenidos en MATLAB.

En la Figura 3.20 se muestra una captura de pantalla del archivo .COE utilizado para esta prueba. En la figura 3.21 se muestra una captura de pantalla del script de MATLAB que obtiene la magnitud de la salida de la FFT implementada en la FPGA.

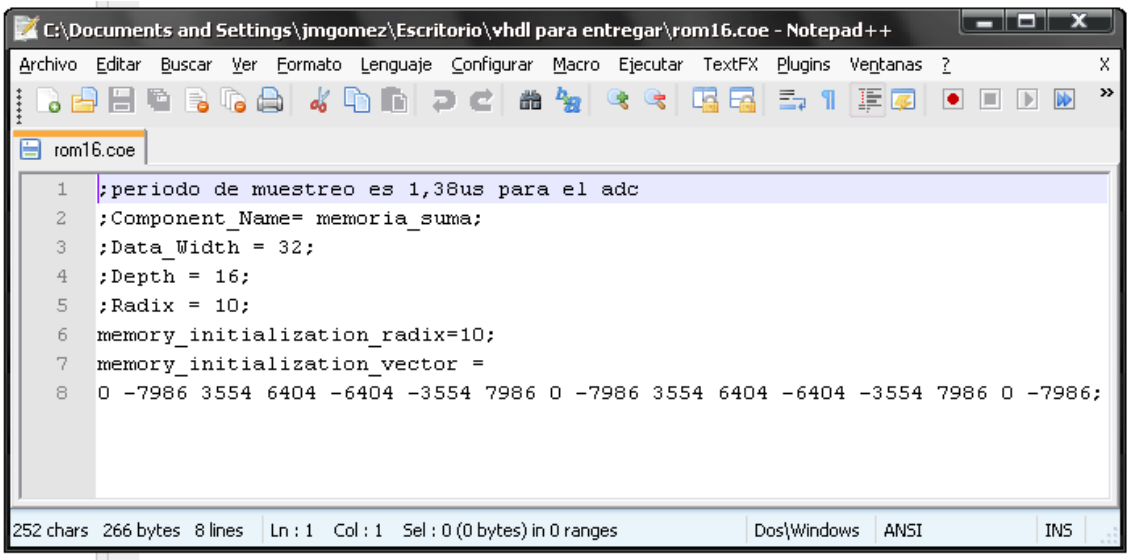

Figura 3.20. Archivo .COE, datos de entrada para prueba sobre la FFT.

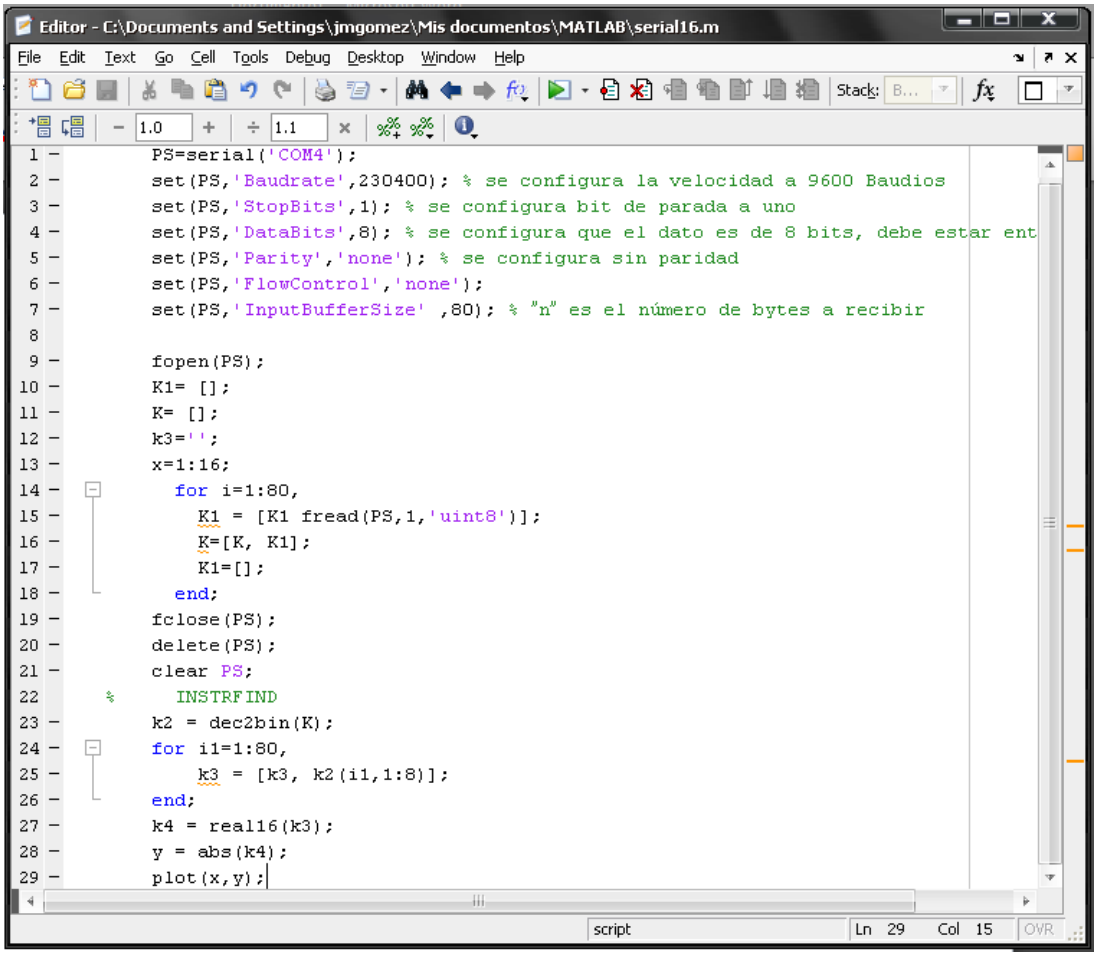

Figura 3.21. Script de Matlab para cálculo de magnitud en prueba sobre la FFT.

3.2.2.4 Pruebas sobre el módulo de operaciones en punto flotante.

Para realizar la prueba sobre este módulo se realiza una implementación en VHDL que permita cargar valores desde un archivo .COE para que sean procesados por el modulo y las salidas que entrega el bloque son transmitidas hacia el computador utilizando el módulo de transmisión RS232. En el computador estos valores son comparados con los valores que se obtienen al realizar el mismo proceso utilizando MATLAB.

En la Figura 3.22 se presenta el archivo .COE de datos de entrada para la prueba sobre este bloque. En la Figura 3.23 se muestran los valores recibidos desde la FPGA. En la tabla 3.4 se realiza una comparación entre los valores obtenidos mediante la implementación en la FPGA y el script en MATLAB, aquí se observa que los resultados en ambos casos son muy similares.

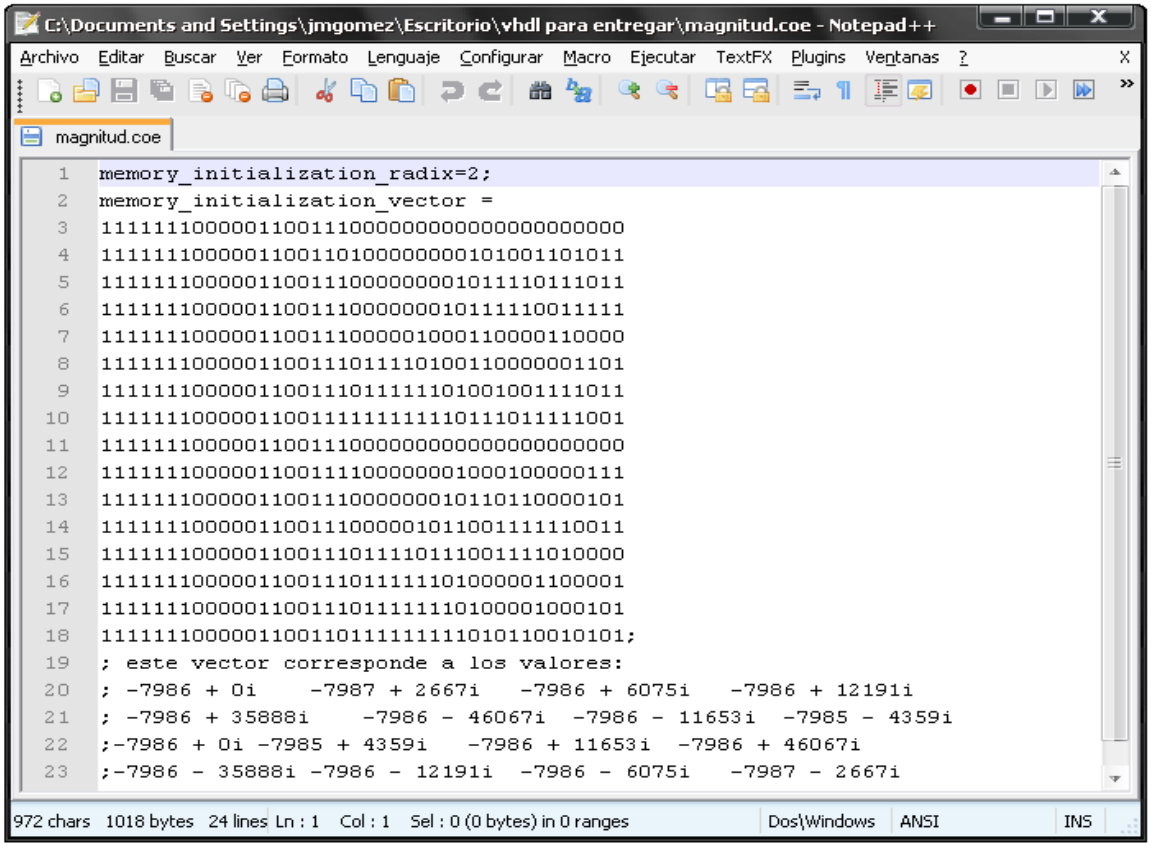

Figura 3.22. Archivo .COE de entrada para la prueba sobre el bloque de operaciones en punto flotante

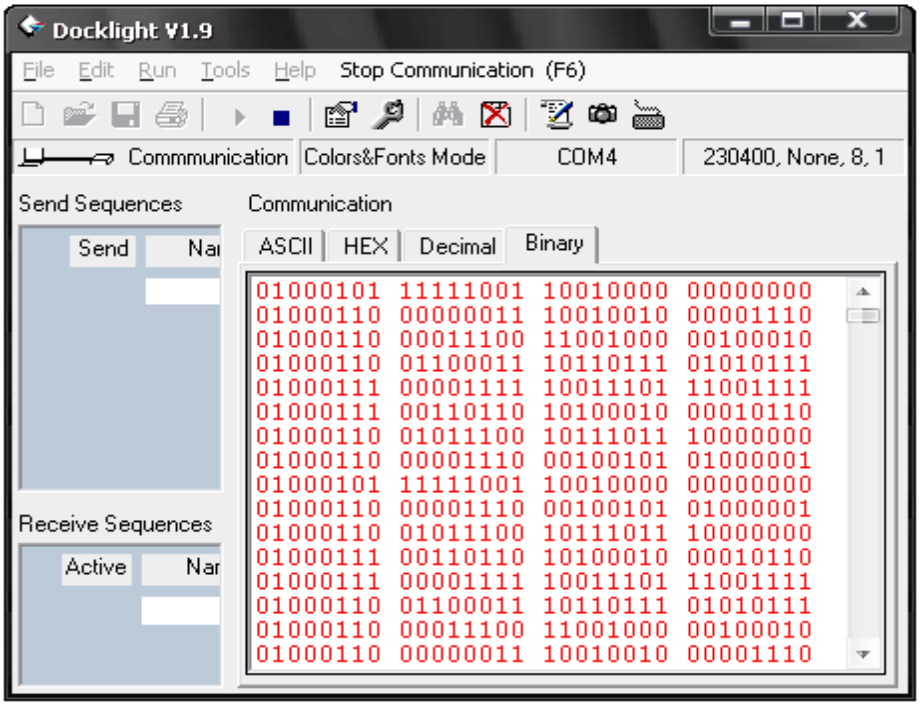

Figura 3.23. Valores recibidos en el computador desde la FPGA

Datos del .COE	Valores recibidos en el PC, generados por la fpga	Valores obtenidos en MATLAB
-7986+ 0i	7986	7986
-7987+ 2667i	8420,514	8420,51411732087
-7986+ 6075i	10034,03	10034,0331372784
-7986+ 12191i	14573,83	14573,8353565559
-7986+ 35888i	36765,81	36765,8093886154
-7986- 46067i	46754,09	46754,0873614276
-7986 – 11653i	14126,88	14126,8752737468
-7985 – 4359i	9097,313	9097,31311981730
-7986+ 0i	7986	7986
-7985+ 4359i	9097,313	9097,31311981730
-7986+ 11653i	14126,88	14126,8752737468
-7986+ 46067i	46754,09	46754,0873614276
-7986- 35888i	36765,81	36765,8093886154
-7986- 12191i	14573,83	14573,8353565559
-7986- 6075i	10034,03	10034,0331372784
-7987- 2667i	8420,514	8420,51411732087

Tabla 3.4. Valores Comparativos de la prueba sobre el bloque de operaciones en punto flotante.

3.2.2.5 Pruebas sobre el módulo de trasmisión RS232.

Para esta prueba se realizó un módulo en VHDL que con un contador que genere números consecutivos, estos valores son enviados al computador a través del módulo RS232 implementado para el sistema de análisis de señales tipo poliscopio utilizando un cable tipo DB9[21]. En el computador los datos son recibidos utilizando un software de monitoreo para el puerto serial y se verifica que los valores recibidos sean consistente con los valores enviados desde la FPGA, es decir que se conserve la secuencia que está enviando el contador.

En la figura 3.24 se muestra el resultado de la prueba realizada, se observa que los datos recibidos en el software de monitoreo son una secuencia de valores consecutivos como se especificó que iba funcionar el módulo de prueba.

[21] El conector **DB9** (originalmente *DE-9*) es un conector analógico de 9 clavijas utilizado principalmente para conexiones en serie, ya que permite una transmisión asíncrona de datos según lo establecido en la norma RS-232 (RS-232C).

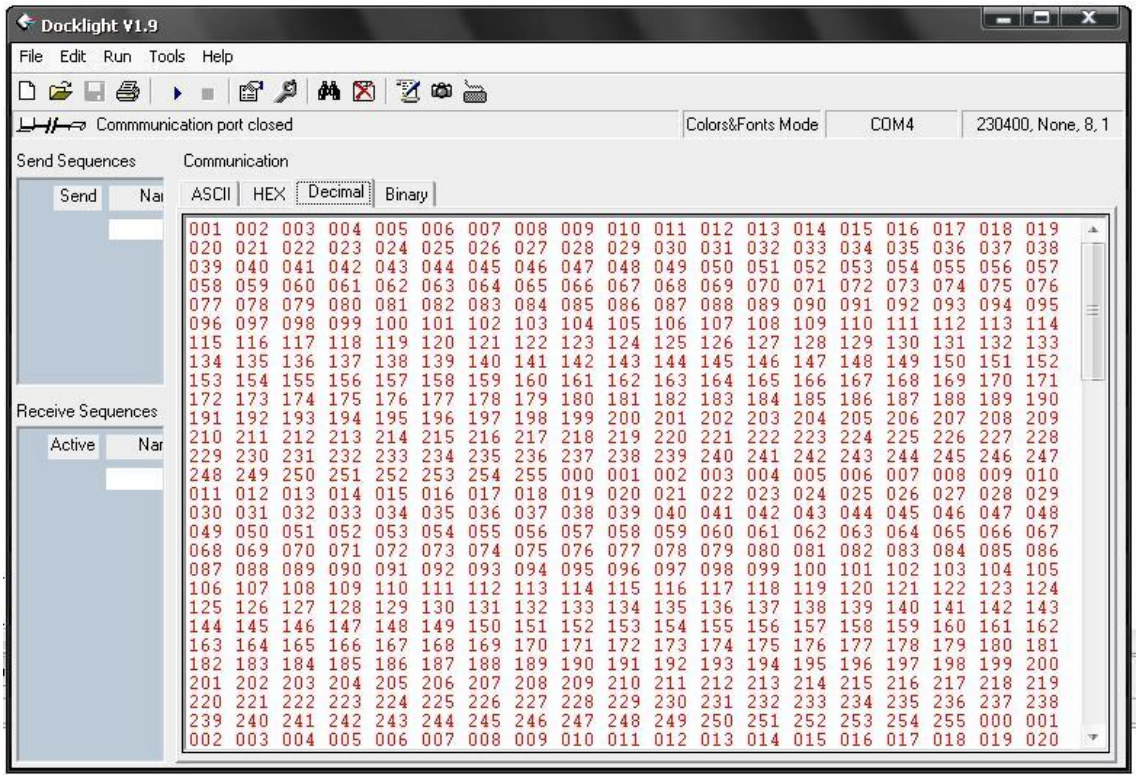

Figura 3.24. Datos recibidos desde el modulo de trasmisión RS232

3.2.3 Pruebas físicas para escalamiento del sistema.

Ahora que se han realizado las pruebas sobre los módulos del sistema de análisis de señales, implementado en la FPGA, es necesario determinar hasta qué punto se pueden aumentar sus capacidades sin degradar el rendimiento. Para esto, se aumenta progresivamente el tamaño de las memorias y el tamaño de la FFT en potencias de dos como es permitido y se realiza una prueba del sistema cada vez, comparando el resultado obtenido utilizando la FPGA con el resultado que se obtiene utilizando MATLAB.

La comparación se realiza mediante el examen de la gráfica de la salida del sistema implementado sobre la FPGA versus la gráfica de la magnitud de la FFT realizada en MATLAB.

Para ambos casos se carga el mismo archivo de valores de entrada. En la FPGA se carga un archivo .COE con las muestras de una señal sinusoidal de amplitud 1 voltio y de frecuencia 50 KHZ. Esta señal es muestreada a 724 637.6812 Hz que es la misma frecuencia de muestreo a la cual está configurado el ADC en la FPGA para esta implementación. Además se realiza una multiplicación por 8192 puesto que éste es el valor de resolución del ADC. Estas muestras son generadas utilizando un script de MATLAB que llamamos senosoidal.m cuya captura de pantalla se muestra en la figura 3.25. Los resultados son trasmitidos al computador mediante el módulo de trasmisión RS232 y son recibidos en MATLAB para ser procesados directamente mediante el script

soperaciones.m[22], que realiza la conversión de punto flotante a decimal con signo y obtiene la gráfica a partir de estos datos. En MATLAB estos datos son generados utilizando el mismo vector que se obtiene de la función senosoidal.m con el fin de conseguir como ya se mencionó, los mismos valores de entrada.

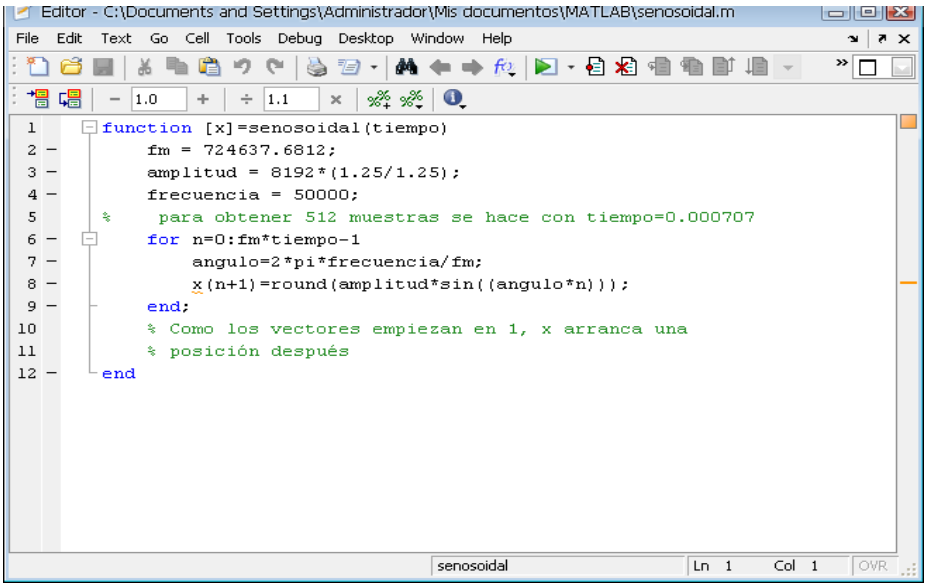

Figura 3.25. Archivo senosoidal.m para generar muestras de una señal sinusoidal.

Es importante señalar que en este documento se registran las pruebas realizadas con tamaño de transformada de 128 muestras en adelante, debido a que para un número menor de muestras el resultado del sistema es exactamente igual al resultado obtenido en MATLAB.

3.2.3.1 Prueba del sistema procesando 128 muestras.

Para el sistema de análisis de señales de tipo poliscopio implementado sobre la FPGA utilizando una FFT de 128 muestras (ver figura 3.26) se obtiene un resultado exactamente igual al obtenido en MATLAB con el mismo número de muestras, esto se comprueba en la figura 3.27 donde se comparan ambas graficas superponiendo una con otra. Para mayor claridad, en la figura 3.28 se muestra la gráfica de la salida del sistema implementado sobre la FPGA y en la figura 3.29 la gráfica obtenida en MATLAB para el mismo número de muestras.

[22] Este script se incluye en el CD que se adjunta con este documento.

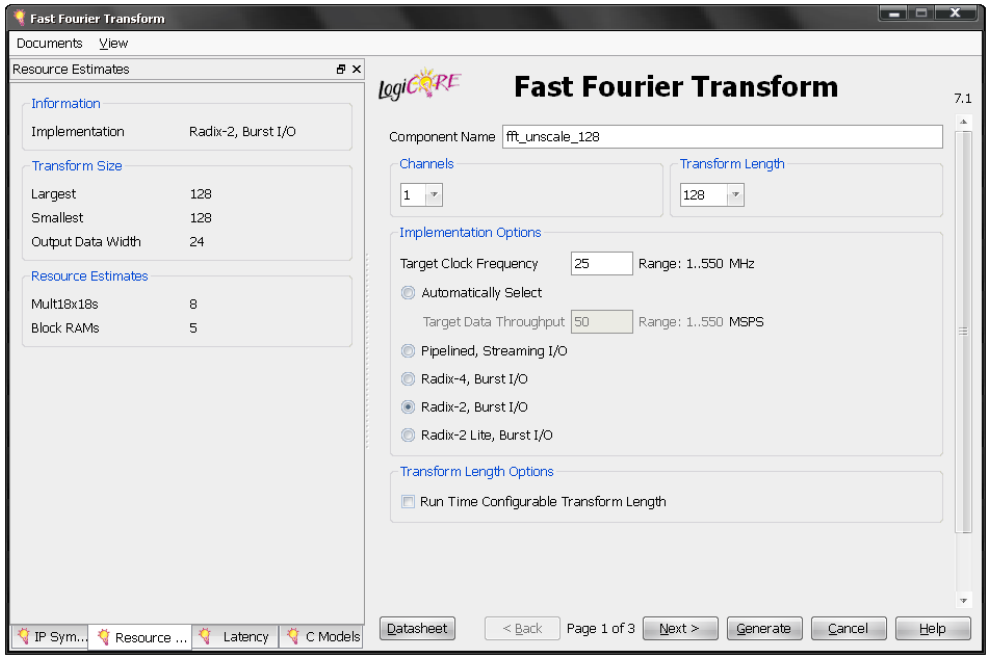

Figura 3.26. CORE FFT con N=128.

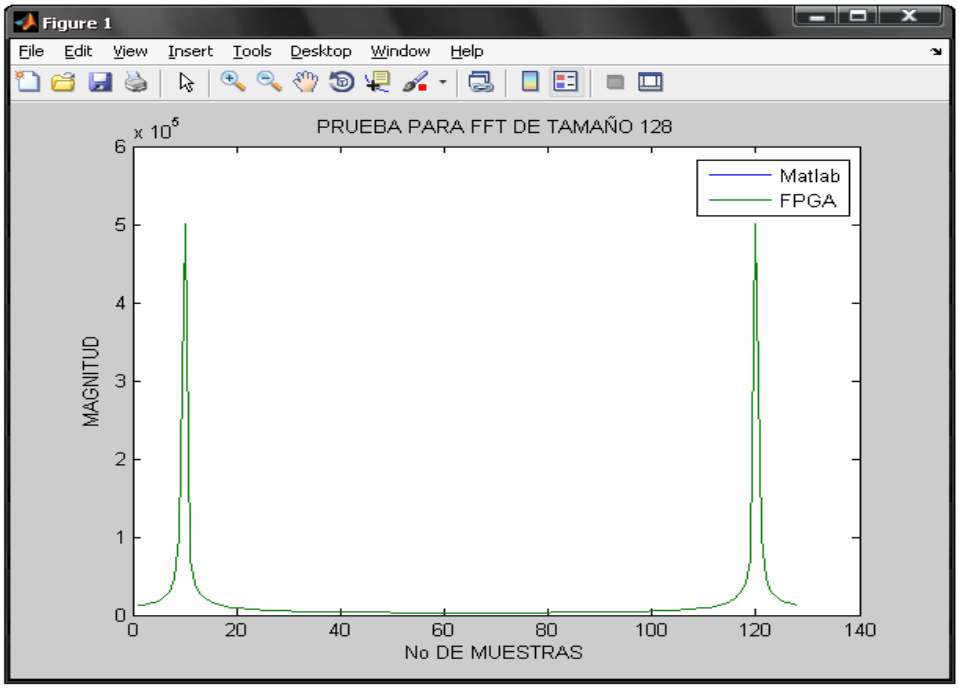

Figura 3.27. Comparación graficas MATLAB y FPGA para FFT de 128 muestras.

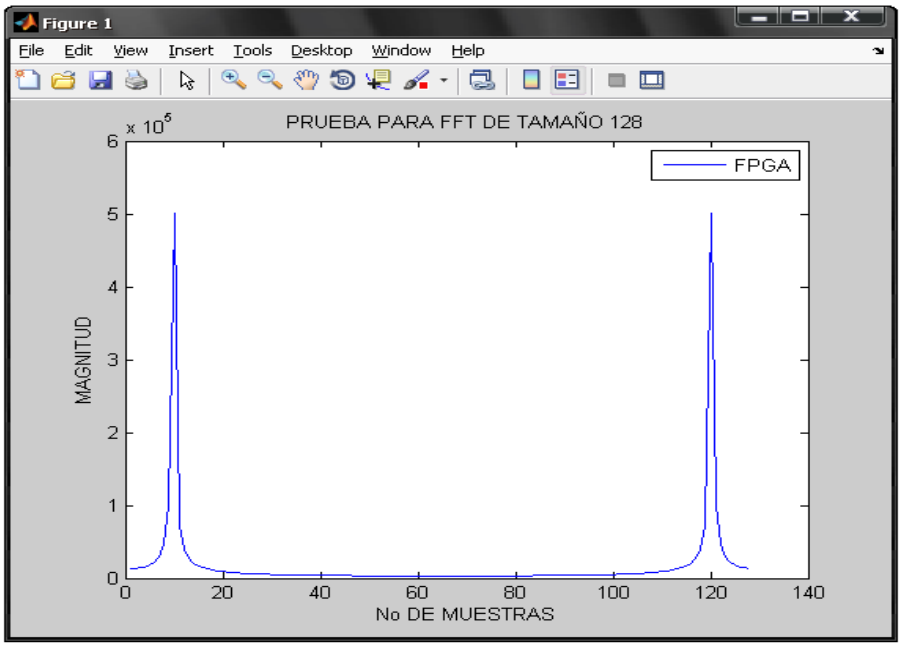

Figura 3.28. Magnitud de FFT de 128 muestras en FPGA.

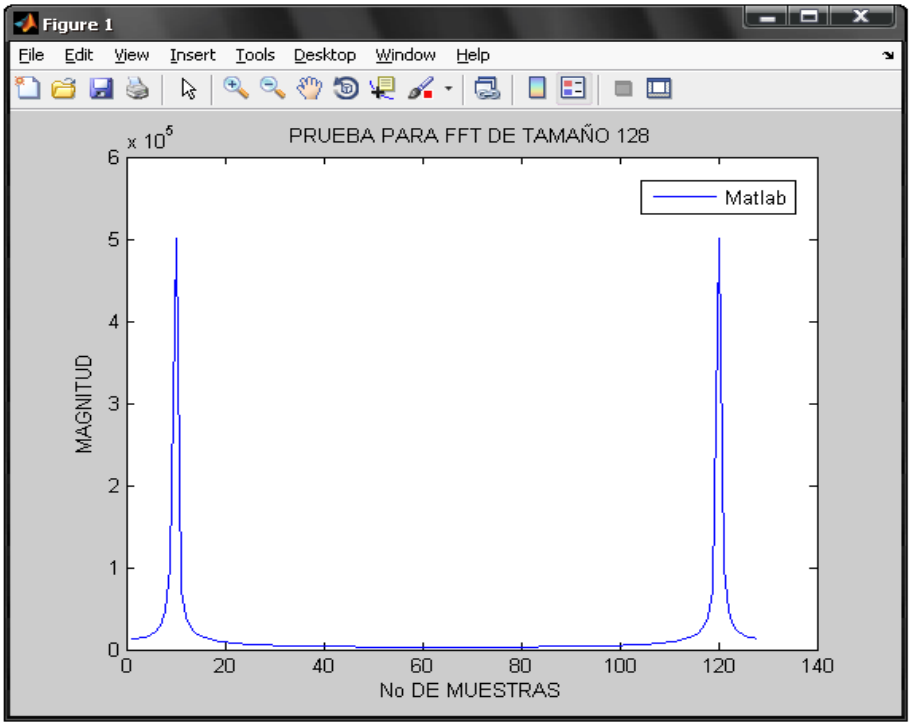

Figura 3.29. Magnitud de FFT de 128 muestras en MATLAB.

3.2.3.2 Prueba del sistema procesando 256 muestras.

En este caso se realiza el mismo procedimiento que en el caso anterior, en la figura 3.30 se muestra una captura de pantalla de la generación del CORE, en la Figura 3.31 se muestra la comparación de la FFT obtenida en MATLAB con la obtenida en la FPGA, en la figura 3.32 se presenta la gráfica que se obtiene utilizando los valores de magnitud

procesados en la FPGA y la figura 3.33 corresponde a la gráfica de la magnitud de la FFT utilizando MATLAB. Para este caso ambos resultados son idénticos.

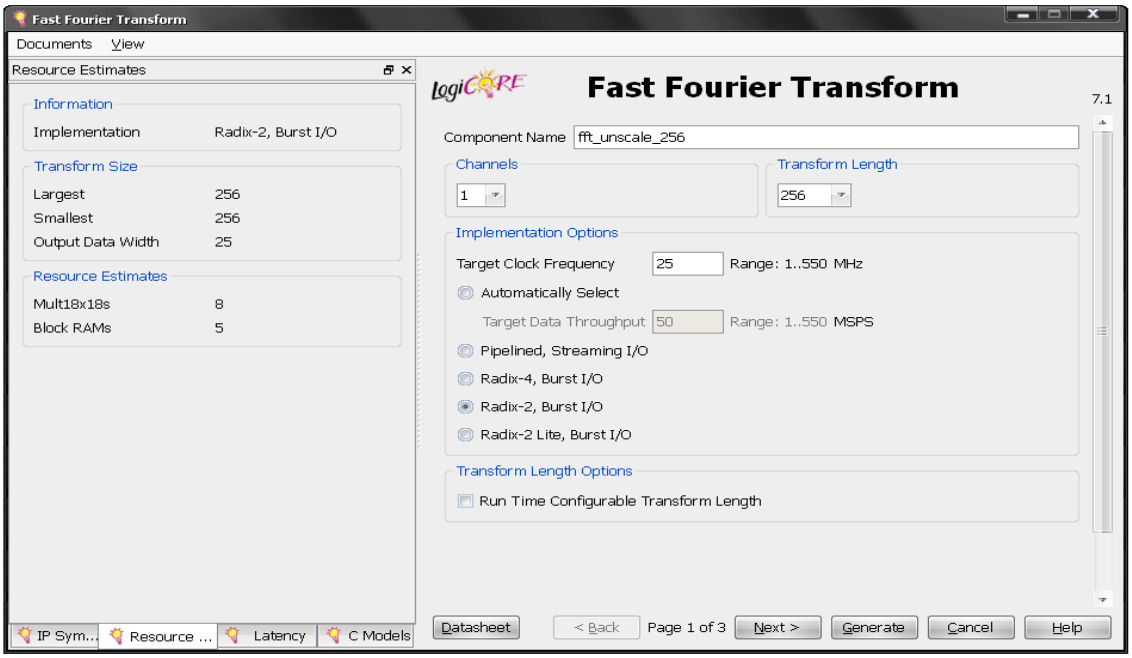

Figura 3.30. CORE FFT con N=256.

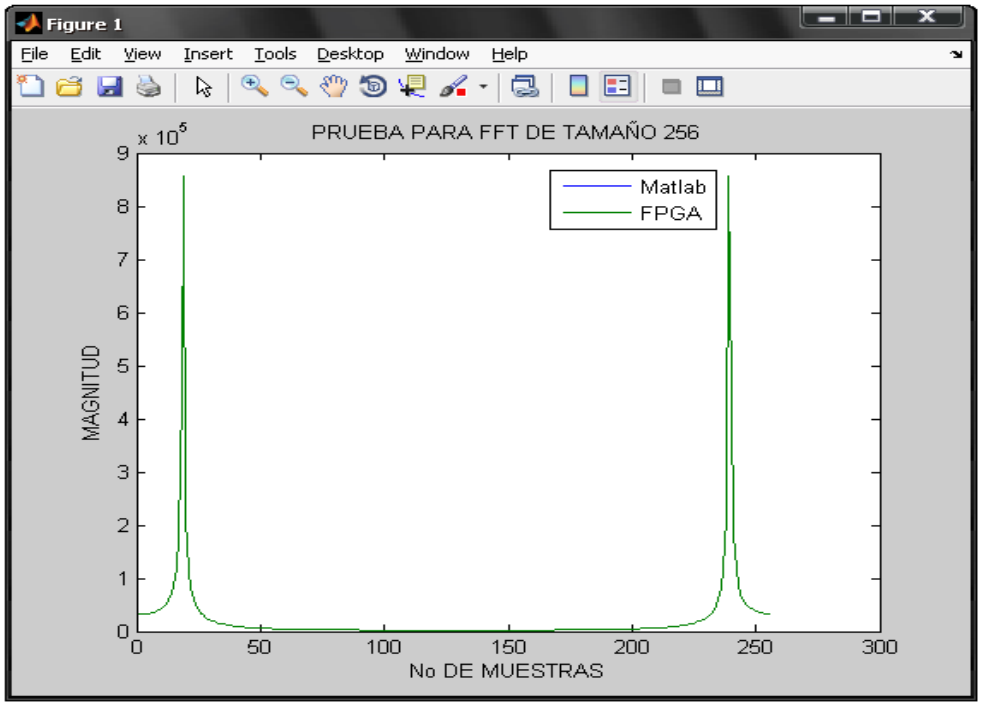

Figura 3.31. Comparación graficas MATLAB y FPGA para FFT de 256 muestras.

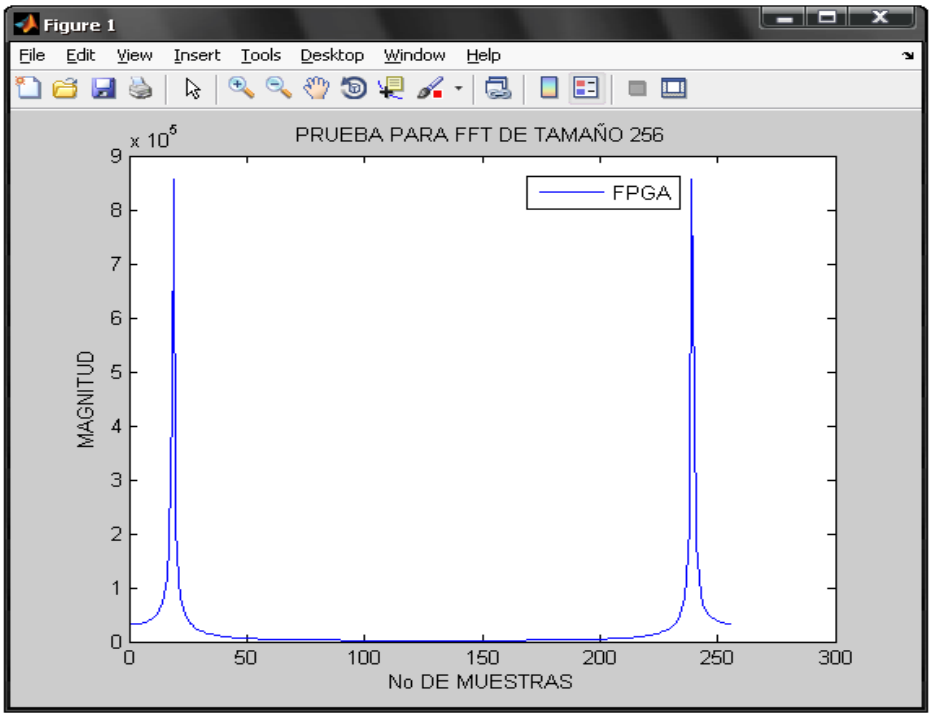

Figura 3.32. Magnitud de FFT de 256 muestras en FPGA.

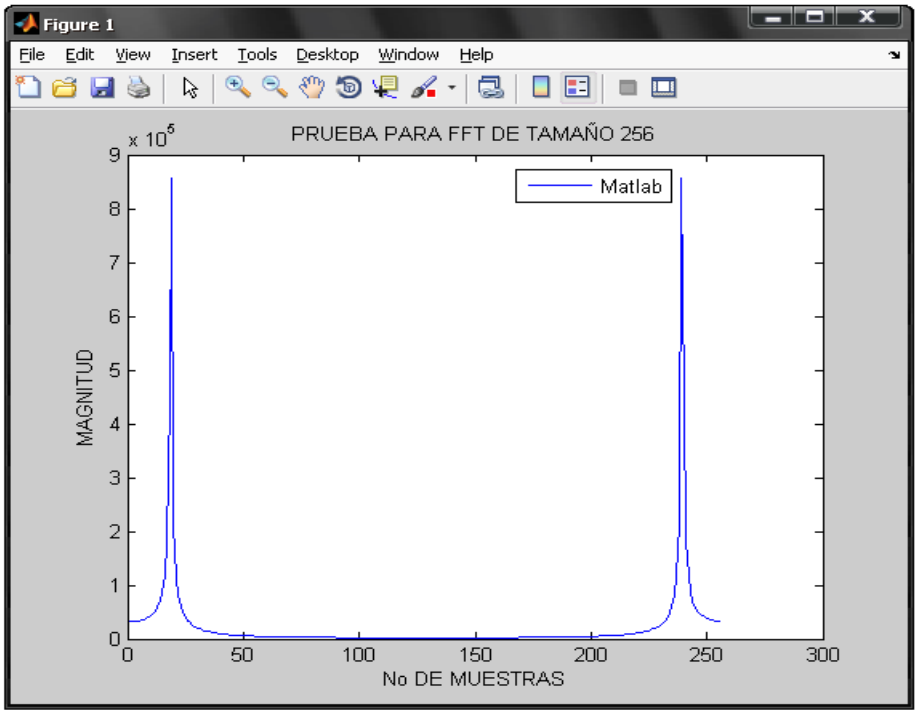

Figura 3.33. Magnitud de FFT de 256 muestras en MATLAB.

3.2.3.3 Prueba del sistema procesando 512 muestras.

Para la prueba del sistema con 512 muestras se realiza el mismo procedimiento que en el caso anterior, en la figura 3.34 se muestra una captura de pantalla de la generación del Core, en la Figura 3.35 se muestra la comparación de la FFT obtenida en MATLAB con la obtenida en la FPGA, en la figura 3.36 se presenta la gráfica que se obtiene utilizando los valores de magnitud procesados en la FPGA y la figura 3.37 corresponde a la gráfica de la magnitud de la FFT utilizando MATLAB. Para este caso ambos resultados son muy similares.

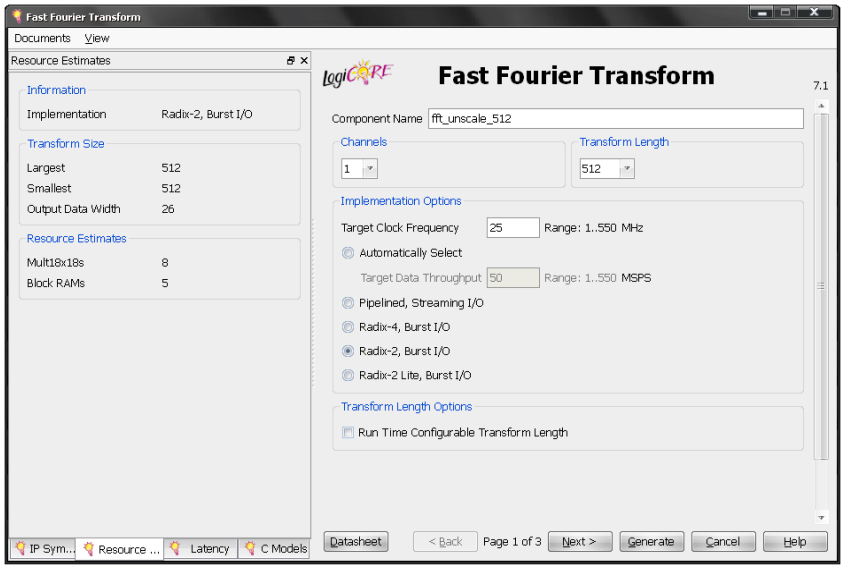

Figura 3.34. CORE FFT con N=512

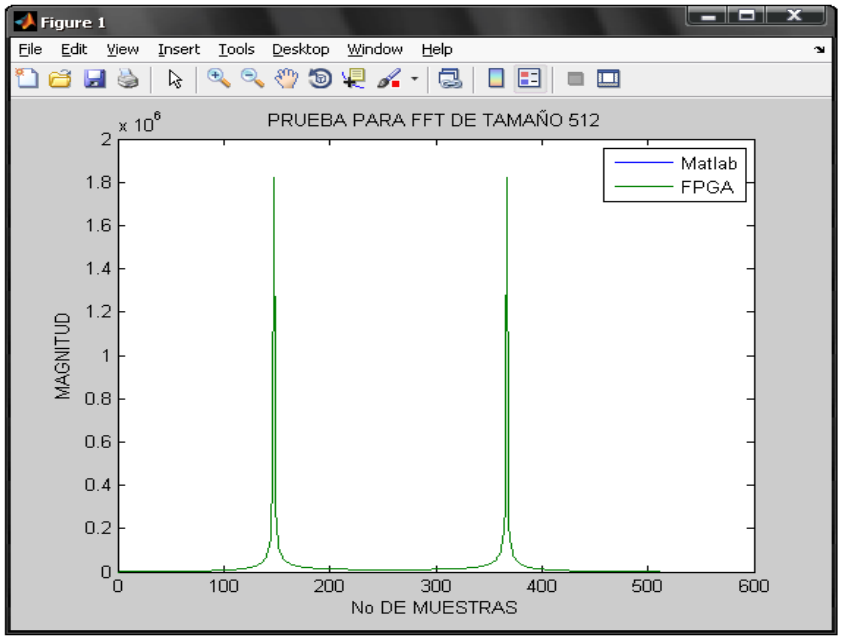

Figura 3.35. Comparación Graficas MATLAB y FPGA para FFT de 512 muestras.

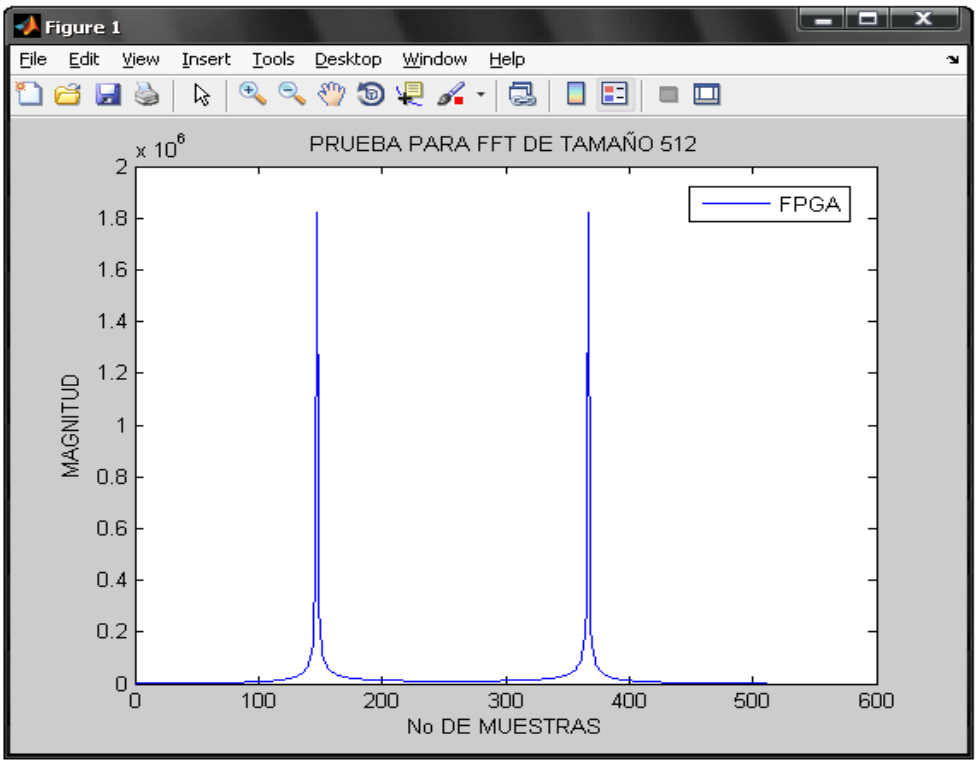

Figura 3.36. Magnitud de FFT de 512 muestras en FPGA

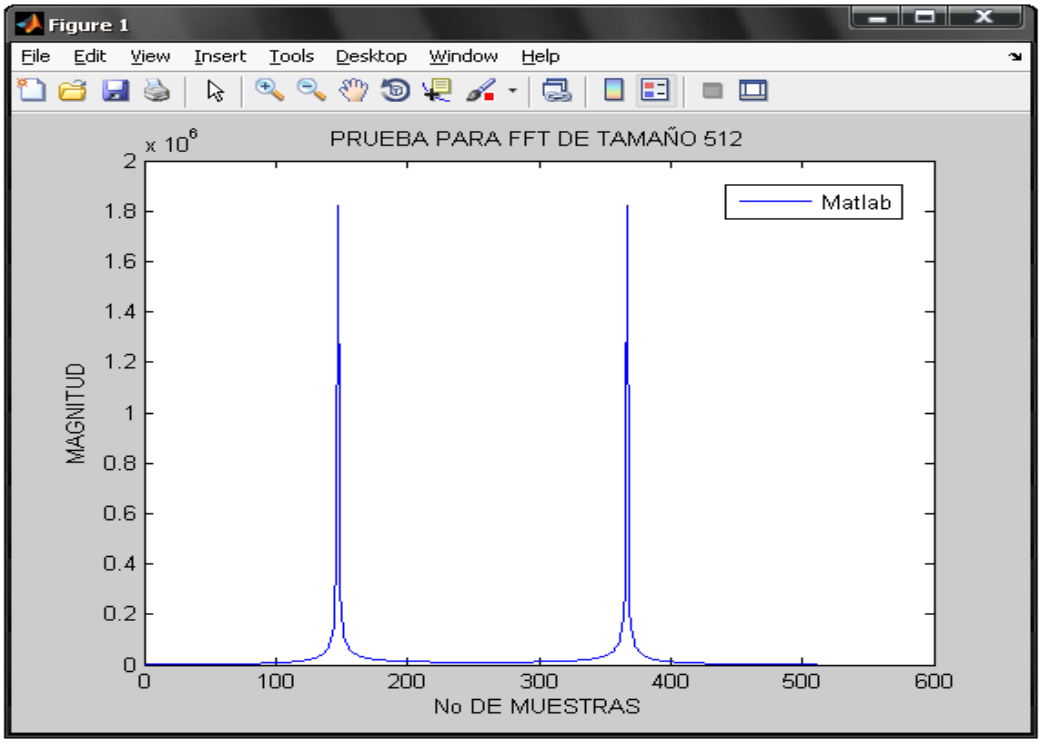

Figura 3.37. Magnitud de FFT de 512 muestras en MATLAB

3.2.3.4 Prueba del sistema procesando 1024 muestras.

Para realizar la prueba del sistema con 1024 muestras se procede de igual forma que para el caso de 512 muestras. En la figura 3.38 se muestra una captura de pantalla de la generación del CORE, en la Figura 3.39 la comparación de la FFT obtenida en MATLAB con la obtenida en la FPGA, en la figura 3.40 se presenta la gráfica que se obtiene utilizando los valores de magnitud procesados en la FPGA y la figura 3.41 corresponde a la gráfica de la magnitud de la FFT utilizando MATLAB, en este caso también se obtienen resultados muy similares con ambos sistemas. Sin embargo, no se continua escalando el sistema debido a que con este tamaño de transformada el uso de recursos de la FPGA (Slices, LUTS, bloques de RAM, multiplicadores, etc.) aumenta considerablemente alcanzando sus límites, lo cual es un riesgo porque el sistema podría tener un mal desempeño o no podría funcionar al momento de procesar señales. En la figura 3.42 se muestra el resumen de uso de recursos para esta implementación, aquí se observa que el porcentaje de uso de recursos de la FPGA está entre el 70% para Slices ocupados, 90% para el uso de multiplicadores, 80% para bloques de RAM. Entre otros.

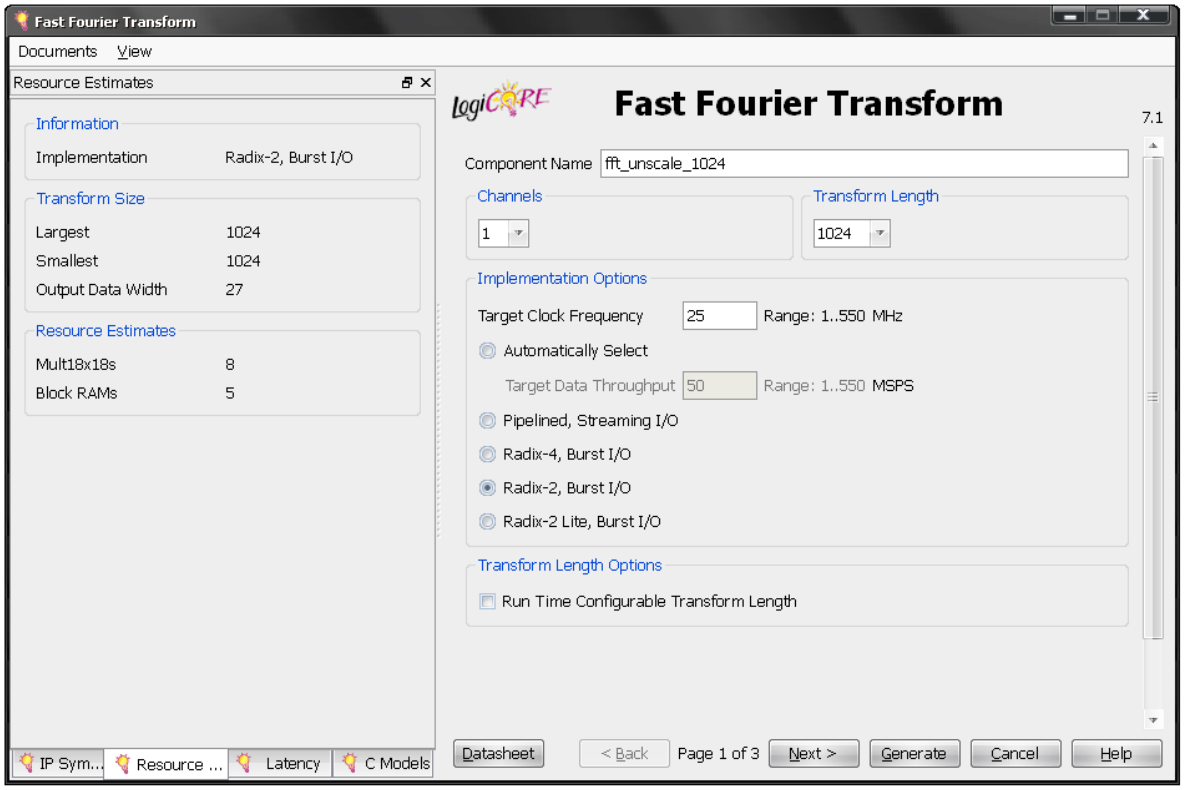

Figura 3.38. CORE FFT con N=1024.

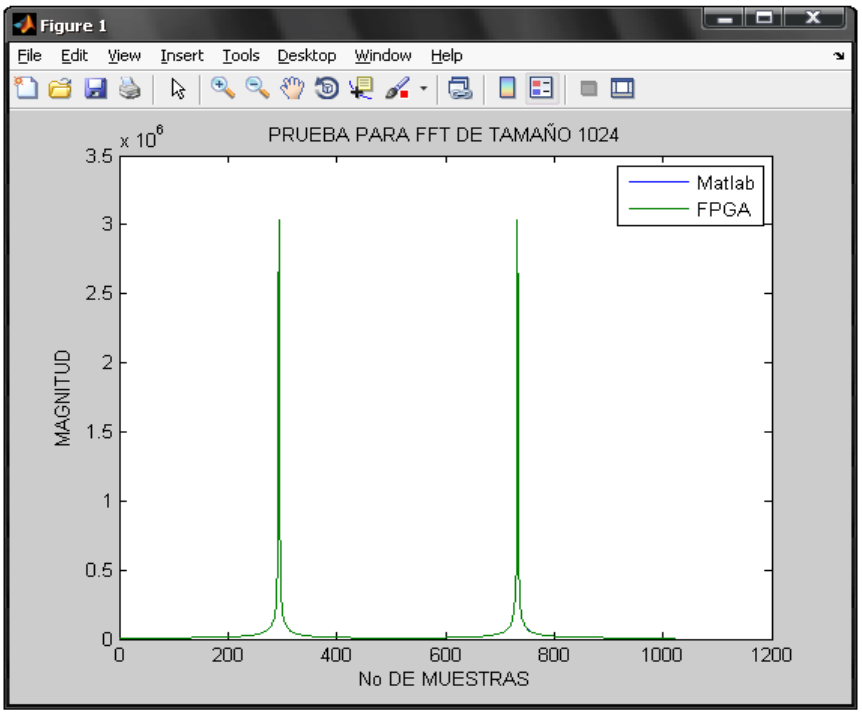

Figura 3.39. Comparación gráficas MATLAB y FPGA para FFT de 1024 muestras.

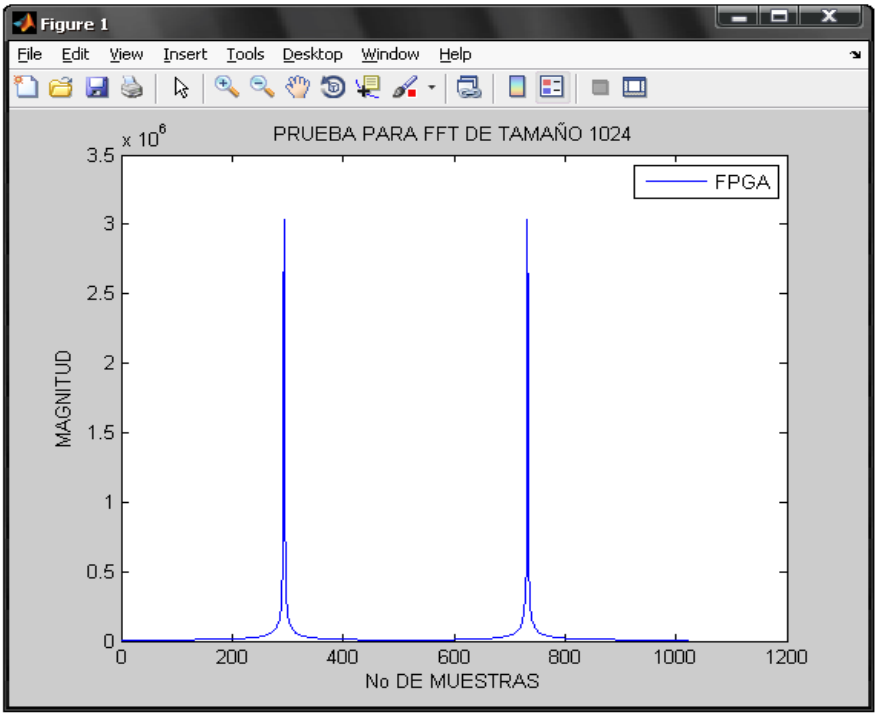

Figura 3.40. Magnitud de FFT de 1024 muestras en FPGA.

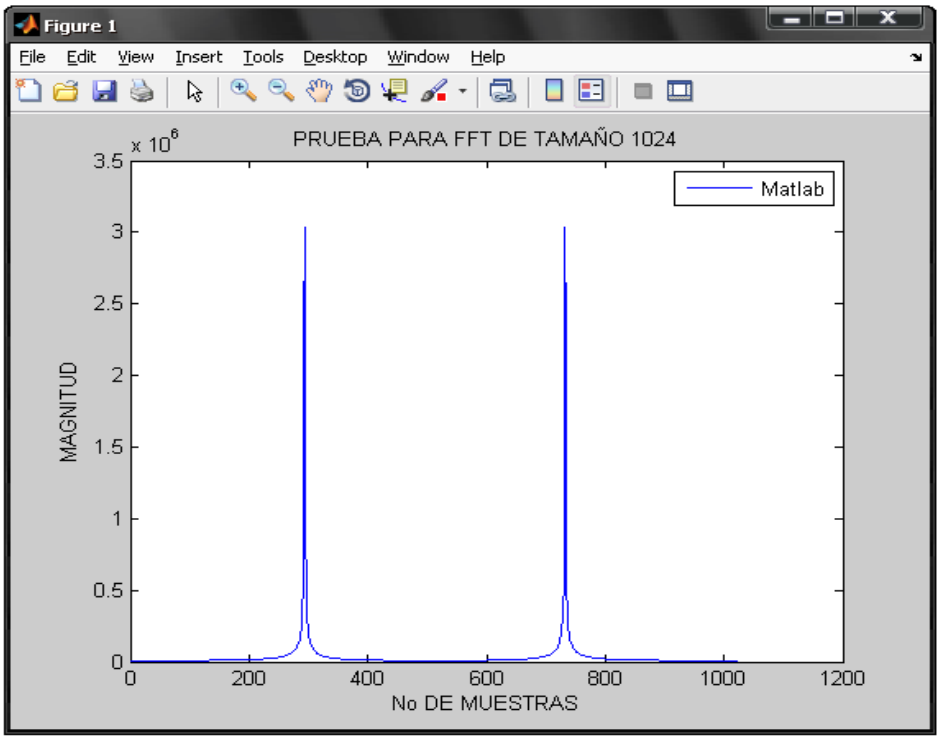

Figura 3.41. Magnitud de FFT de 1024 muestras en MATLAB

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	7,150	9,312	76%
Number of 4 input LUTs	7,964	9,312	85%
Number of occupied Slices	3,283	4,656	70%
Number of Slices containing only related logic	3,283	3,283	100%
Number of Slices containing unrelated logic	0	3,283	0%
Total Number of 4 input LUTs	7,150	9,312	76%
Number used as logic	3,585		
Number used as a route-thru	386		
Number used as Shift registers	379		
Number of bonded IOBs	198	232	85%
Number of RAMB16s	16	20	80%
Number of BUFGMUXs	17	24	70%
Number of MULT18X18SIOs	18	20	90%
Average Fanout of Non-Clock Nets	2.63		

Figura 3.42. Resumen de utilización de recursos para la implementación del sistema de análisis de señales con una FFT de 1024 muestras.

3.2.4 Prueba del sistema de análisis de señales con FFT de 1024 muestras utilizando una señal de audio.

Hasta el momento se ha evaluado el sistema de análisis en la FPGA utilizando una señal sinusoidal simple y la respuesta ha sido satisfactoria hasta llegar a 1024 muestras para la FFT. En este punto se llega casi al límite de uso de recursos de la FPGA así que se fija el tamaño máximo de transformada y por ende del prototipo del sistema de análisis de señales de tipo poliscopio en 1024 muestras.

Ahora es necesario determinar el comportamiento del sistema cuando tiene que procesar una señal mucho más compleja como por ejemplo una señal de audio. Para esto se utiliza la señal Handel de MATLAB, esta señal se carga en una variable y se obtienen 1024 muestras. Es necesario adecuar estas muestras de la señal para ser cargadas en la tarjeta, dividiendo por 1.25 y luego multiplicando por 8192 de acuerdo con la ecuación Ec2.1 del ADC presentada en la sección 2.3 del capítulo 2. Con estas muestras ya adecuadas se crea un archivo .COE que se carga en la FPGA mediante una memoria ROM y se realiza su análisis. El resultado obtenido es comparado con MATLAB.

En la figura 3.43 se muestra una gráfica de la señal de prueba Handel limitada a 1024 muestras. En la figura 3.44 se muestra una superposición de ambos resultados, el obtenido en MATLAB y el generado por la FPGA. En la figura 3.45 se presenta el resultado de la magnitud de la FFT calculada en la FPGA y en la figura 3.46 el resultado obtenido utilizando MATLAB.

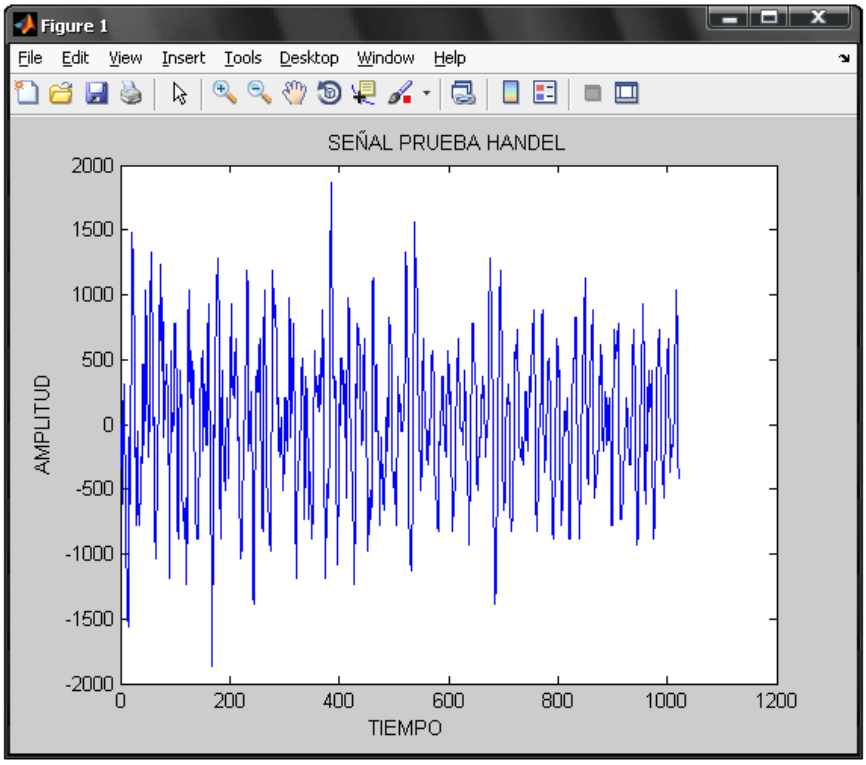

Figura 3.43. Señal de prueba Handel, dominio del tiempo.

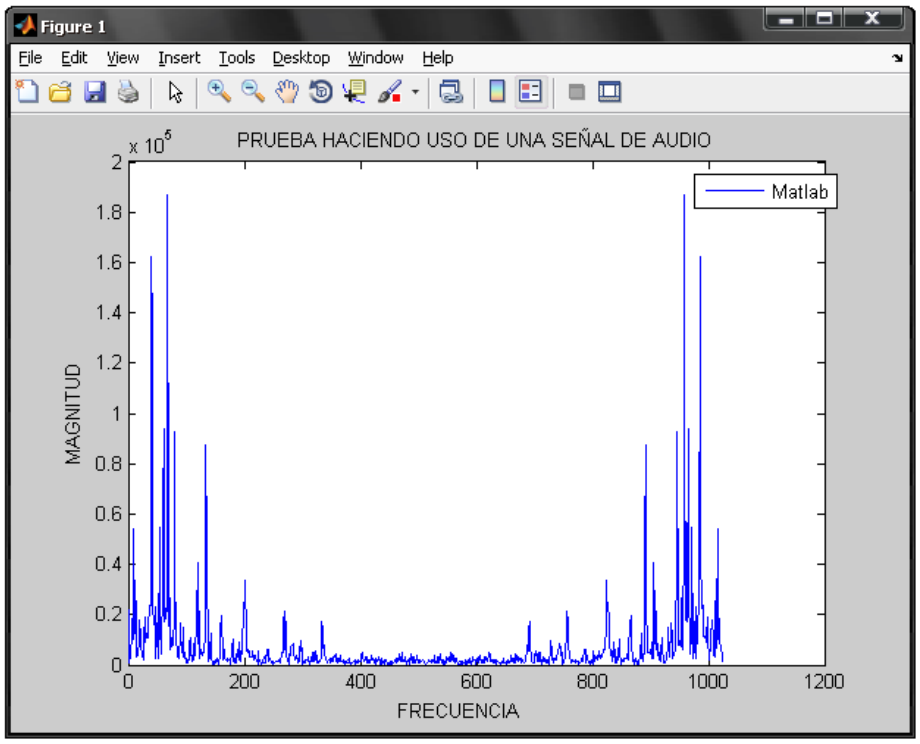

Figura 3.44. Magnitud de la FFT obtenida en la FPGA con 1024 muestras de la señal Handel.

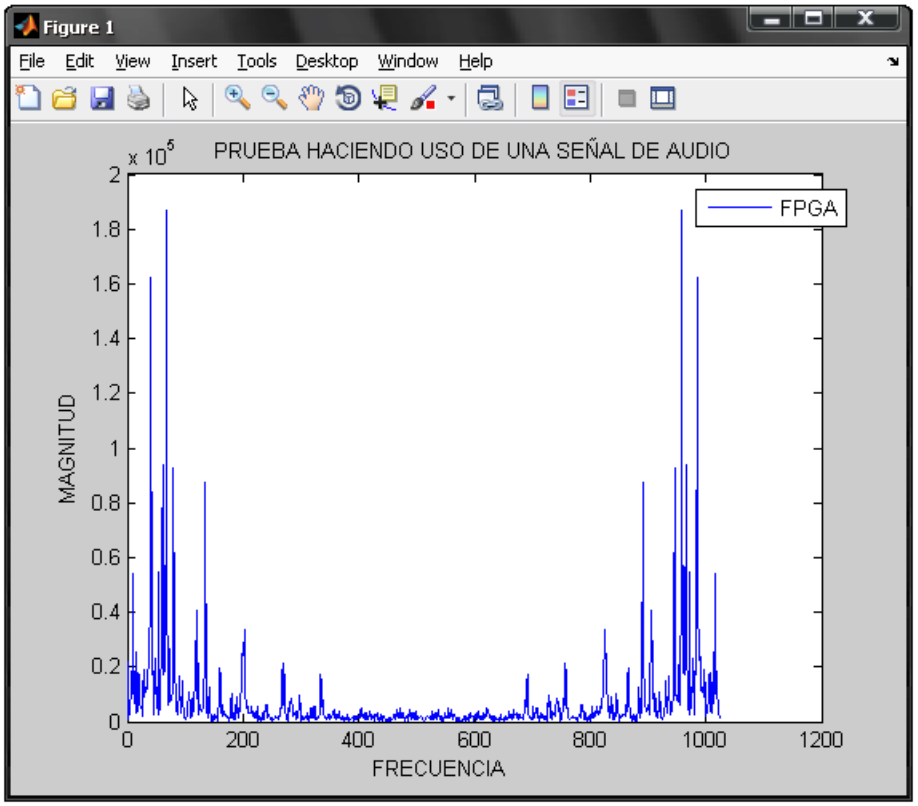

Figura 3.45. Magnitud de la FFT obtenida en MATLAB con 1024 muestras de la señal Handel.

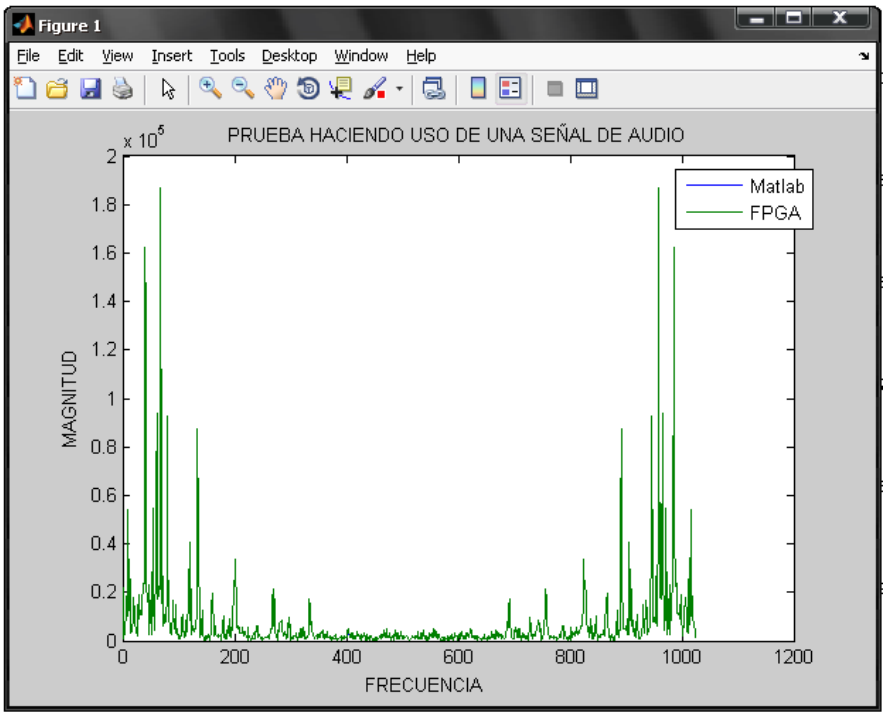

Figura 3.46. Superposición de graficas de ambas FFT, la obtenida en la FPGA y la obtenida mediante MATLAB.

3.2.5 Prueba de análisis de señal mediante una línea de transmisión.

Considerando que el poliscopio realiza análisis de señales sobre la salida de un circuito de prueba que puede ser, desde un filtro hasta una línea de transmisión, se realiza esta prueba, que consiste en obtener una curva de voltaje contra frecuencia de una señal que ha sido sometida a una línea de transmisión coaxial. Se utiliza Simulink para construir un modelo que emule el comportamiento de la línea de transmisión coaxial y de la configuración propuesta mediante bloques de RF debido a que físicamente la prueba no se puede realizar porque el rango de operación del ADC de la Spartan 3A es insuficiente para ello, y porque la generación y adecuación de esta señal para que sea entrada del sistema de análisis de señales de tipo poliscopio utilizando FPGA se sale de los alcances de ese proyecto.

El cable coaxial elegido fue el RG-58c. Las características físicas y eléctricas de este cable son las siguientes:

- Impedancia de cable 50Ω
- Dieléctrico en polietileno sólido, el cual posee una velocidad de propagación de 54.9% de la velocidad de la luz.
- El radio exterior del conductor es de 5 milímetros y el radio interior es de 1.411 milímetros.

En la figura 3.47 se muestra una gráfica de la disposición de los bloques de Simulink para esta prueba.

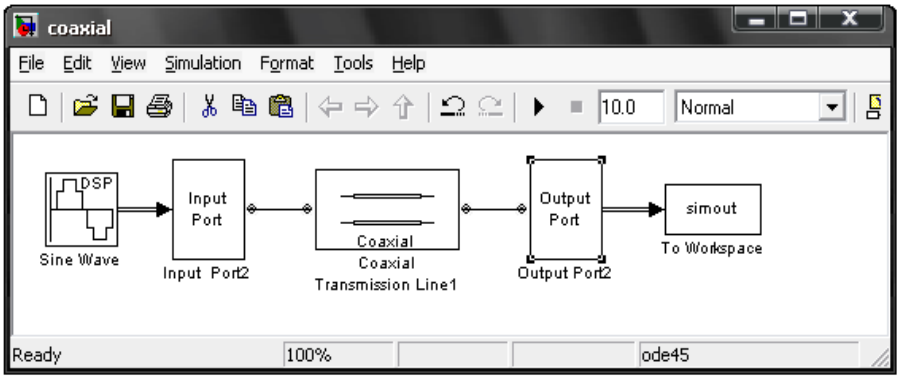

Figura 3.47.Diagrama en Simulink para prueba con línea de transmisión.

La señal de entrada es una señal seno en forma discreta que se configura a una frecuencia de 50 MHz con una amplitud de 1 voltio y la frecuencia de muestreo se fija en 1 GHz. La longitud de la línea coaxial se fija en 10 metros. Los bloques de Input Port y Output Port son bloques de adaptación de la señal para poder ser conectada con los bloques de la librería RF de simulink como el bloque Coaxial Transmisión Line.

El bloque Coaxial Transmisión Line permite configurar los parámetros de una línea de transmisión coaxial introduciendo los valores de impedancia, permitividad y permeabilidad, longitud de la línea etc.

El bloque Simout permite exportar la señal de salida del diagrama de Simulink al workspace de MATLAB, para ser utilizado como una variable.

La metodología de la prueba consiste en obtener mediante el sistema de análisis de señales de poliscopio implementado en la FPGA una curva de voltaje contra frecuencia de la señal de entrada y de la señal de salida del sistema mostrado en la figura 3.47.

También, en esta prueba se quiere mostrar que la impedancia característica de la línea de transmisión elegida es de 50 ohmios. Esto se hace cambiando la impedancia de carga a una diferente a la impedancia característica del cable para obtener una medida de voltaje mucho menor que la obtenida cuando se tiene un valor de impedancia característica en la carga que permite máxima transferencia de potencia.

Para pasar los valores de Simulink a la FPGA se utiliza el bloque Simout que pasa una salida de Simulink a una variable en workspace de MATLAB, a partir de esta variable se obtienen las muestras para que se guardan en un archivo .COE para que sean cargadas en la FPGA.

En la figura 3.48 se muestra la disposición de los bloques de Simulink para obtener la señal de entrada.

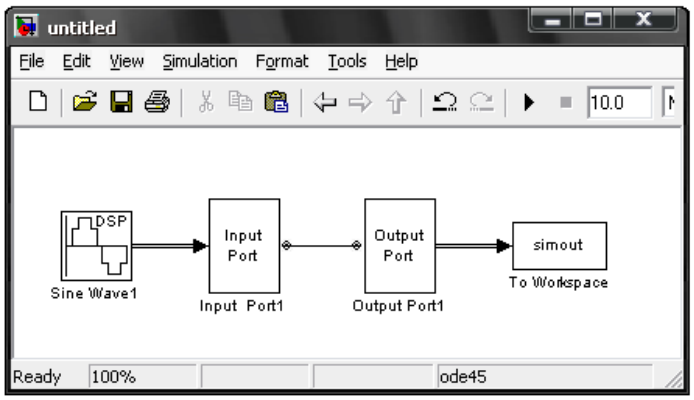

Figura 3.48.Diagrama en Simulink para obtener la señal de entrada para la prueba con línea coaxial.

La grafica de voltaje contra frecuencia obtenida de la señal de entrada se muestra en la figura 3.49. El voltaje de la señal es de 0.9994 centrado en 50MHz.

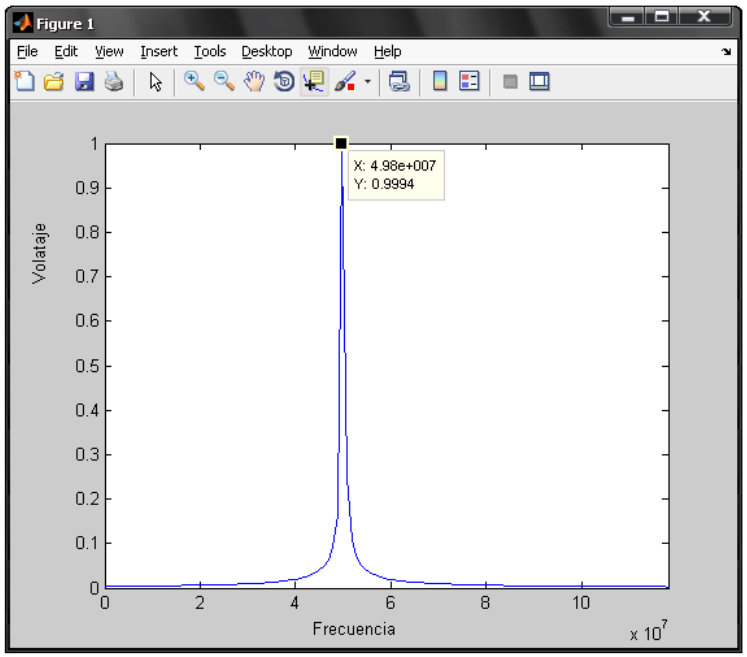

Figura 3.49. Grafica magnitud FFT para la señal de entrada del sistema de la prueba.

La gráfica para la señal de salida, es decir la señal que ha pasado por la línea, se muestra en la figura 3.50. Aquí se observa la disminución del nivel de voltaje de la señal debido a la atenuación producida por la línea.

El voltaje de la señal es de 0.8635 V, centrado a una frecuencia de 50MHz.

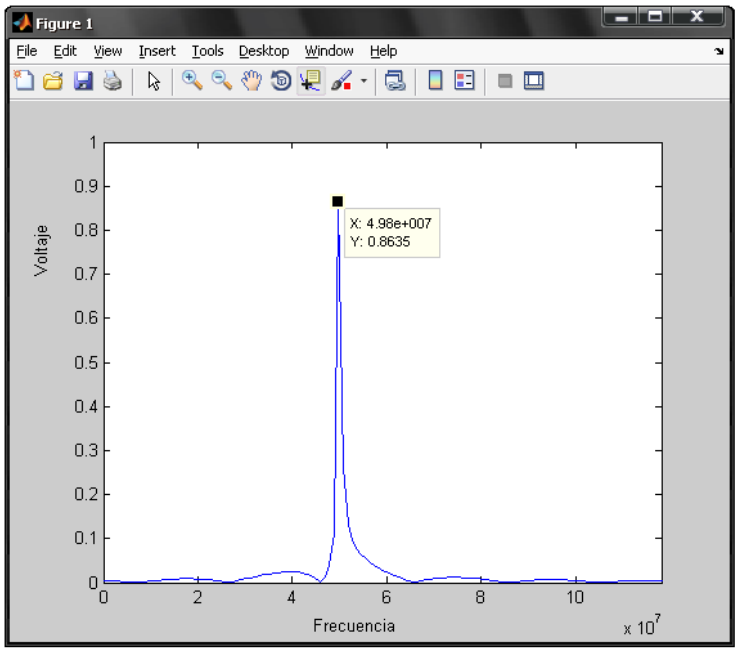

Figura 3.50. Curva Voltaje frecuencia para la señal de salida de la línea.

En la figura 3.51 se muestra la gráfica de la salida de la línea con misma configuración pero con una impedancia de carga de 75 ohmios. Aquí se observa una leve disminución en el nivel de voltaje respecto al obtenido cuando la carga es igual a la impedancia característica. Esto se debe a que cuando se da que la carga es diferente a la impedancia característica, no se produce máxima transferencia de potencia.

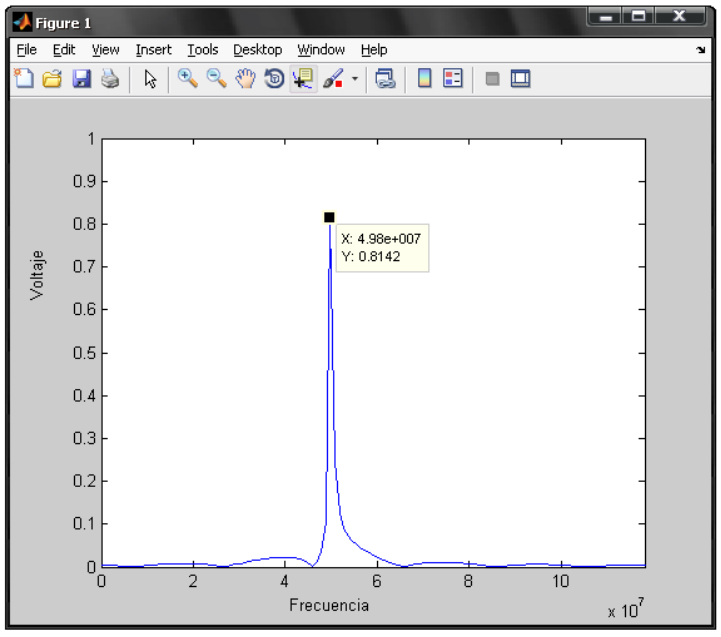

Figura 3.51. Curva voltaje frecuencia para la señal de salida de la línea cuando la carga es diferente a la impedancia característica de la línea.

Capítulo IV Trabajos a futuro y conclusiones.

4.1 Trabajos a Futuro.

- Incorporar un ADC de mayores capacidades que permita digitalizar señales de mayor frecuencia, debido a que la mayor limitación del sistema implementado es el bajo valor de frecuencia de muestreo del ADC LTC1407A con el que cuenta la tarjeta disponible.

- Realizar la aplicación de visualización de la gráfica voltaje contra frecuencia directamente en la FPGA mediante el módulo VGA, con el fin de no tener que utilizar un computador para visualizar los resultados.

- Para la aplicación desarrollada mediante MATLAB se puede buscar una opción bajo licencia GNU para evitar los costos de Licenciamiento

- La aplicación desarrollada puede integrar más funciones como por ejemplo integrar señales de otros sistemas o de simulaciones de Simulink para que las gráficas generadas sean comparadas con las obtenidas mediante la FPGA

- Incorporar características de conectividad mediante la interfaz RJ45 con la que cuenta la FPGA Spartan 3A Starter Kit de tal manera que sea necesaria únicamente una tarjeta y los miembros de una red puedan acceder a sus prestaciones.

- Implementar el sistema en una FPGA de mayores capacidades con un tamaño de FFT mayor a 1024 muestras con el fin de obtener una mejor resolución.

- Incorporar aplicaciones de filtrado sobre la señal graficada.

4.2 Conclusiones.

- El poliscopio es un instrumento muy útil para efectuar medidas sobre líneas de transmisión, filtros y demás circuitos de prueba, permitiendo mediante la curva desplegada observar y determinar características de estos circuitos en un amplio rango de frecuencia. Es un dispositivo complejo ya que integra diversos componentes para realizar su función.

- Los actuales sistemas de medida y de análisis de señal para curvas de voltaje frecuencia no se basan en el funcionamiento del poliscopio tipo SWOB, no presentan como tal una curva voltaje contra frecuencia como lo hace el poliscopio sino que son aplicaciones de procesamiento digital que grafican muchos parámetros incluidas la magnitud y la fase de la transformada de Fourier que ejecutan sobre una señal de entrada.

- Cuando se requiere implementar sistemas complejos, una arquitectura modular basada en componentes facilita esta implementación puesto que el problema principal se divide en subsistemas funcionales más simples que son más fáciles de implementar. En el caso del sistema de análisis de señales de tipo poliscopio se tiene la estructura jerárquica de tipo TOP-DOWN, donde se parte del sistema más complejo y se realiza una abstracción con el fin de obtener componentes más simples de implementar, hizo que fuera posible el total desarrollo del proyecto que de otra manera no hubiera podido implementarse.

- La FFT es una herramienta invaluable a la hora de realizar análisis espectral sobre señales de tipo discreto, su alta eficiencia y la disminución de carga computacional hace posible que pueda ser implementada sobre dispositivos de lógica programable. Sin embargo limitaciones como el tamaño de N hacen que escalar un sistema basado en FFT requiera un amplio uso de recursos, debido a que solo se permiten potencias de dos, si se quiere aumentar el número de muestras se debe duplicar el valor de N lo que implica un doble uso de recursos.

- Una representación en punto flotante facilita la implementación, disminuye los recursos utilizados y agrega una mayor exactitud al sistema. En el caso del sistema de análisis de señales tipo poliscopio, el cambio de una representación en punto fijo a una representación en punto flotante permitió realizar un manejo más adecuado de las operaciones para obtención de magnitud sin que esto representara un excesivo uso de recursos.

- El software ISE™ Design Suite 12.3 (Integrated Software Environment) de Xilinx Inc. es una herramienta de automatización de diseño electrónico o EDA[23] que facilita solución de problemas complejos en dispositivos reconfigurables como la FPGA. Este entorno fue de gran ayuda en la implementación del sistema de

[23] EDA: Electronic Desing Automation - Automatización del Diseño electrónico.

análisis de señales tipo poliscopio pues incorpora en una interfaz gráfica de usuario las herramientas necesarias para la programación, síntesis, depuración, y simulación de un proyecto de descripción de hardware.

- El uso de CORES IP facilito en una gran medida el proceso de implementación del sistema de análisis de señales de tipo poliscopio, porque disminuye el tiempo de desarrollo ya que se cuenta con bloques que han sido probados y depurados para realizar una función específica. Generar un CORE es un proceso sencillo gracias al ISE CORE Generator. Sin embargo, es necesario tener en cuenta la cantidad de recursos empleados y que la configuración con la que se genera este CORE sea la adecuada para el proyecto. Además se requiere adecuar los componentes con los que interactúa el CORE para que puedan integrarse de manera correcta con él.

- Los métodos de simulación provistos por la Suite ISE de Xilinx Inc. fueron fundamentales para realizar el proceso de validación de los componentes del prototipo del sistema de análisis de señales de tipo poliscopio, puesto que, no solo permite visualizar formas de onda, como lo hacen los otros entornos de desarrollo de HDL, sino que también es posible definir archivos test bench donde se especifican las señales que sirven de estímulo para el componente evaluado y la duración de estos estímulos, lo cual hace posible un mejor análisis del comportamiento del bloque que se está simulando.

- Cuando se diseña un proyecto de descripción de hardware sobre lógica programable es necesario considerar los tiempos de inicialización de los componentes implementados, así como también los tiempos de respuesta de los componentes. En el sistema de análisis de señales tipo poliscopio, se obtenía una respuesta inconsistente hasta que se cambió la implementación para que tuviera en cuenta estos tiempos.

- La tarjeta Spartan 3A Starter kit de Xilinx es una FPGA de gama media – baja y fue una gran limitante para el desarrollo del proyecto, principalmente debido a que cuenta con una cantidad reducida de multiplicadores y acumuladores para implementaciones de procesamiento digital de señales de cierta complejidad, además el ADC que incorpora, es de tipo serial, lo que dificulta su manejo; es un ADC de baja frecuencia de muestreo, esto implica que sea imposible que el sistema de análisis de señales tipo poliscopio realice análisis sobre señales de alta frecuencia. Otra característica que limita la Spartan 3A, aparte del bajo reloj que posee, es que incorpora módulos adicionales a la FPGA que comparten recursos que no pueden ser utilizados en funciones propias de una implementación.

- En una implementación de procesamiento digital de señales sobre FPGA el número de multiplicadores que posea el dispositivo programable es un factor crítico para el desarrollo del proyecto.

- Cuando se tiene una limitación de Hardware en un proyecto sobre lógica programable, es recomendable realizar un escalamiento del sistema e ir validando la implementación cada vez que se realiza este escalamiento, con el fin de buscar el mejor balance entre capacidad y rendimiento.

- Matlab y Simulink como entorno integrado de desarrollo y de simulación, fue una herramienta de gran ayuda para la resolución del proyecto, ya que gracias a ella se pudo validar el prototipo, comparando ambos resultados. Además es en esta herramienta en la que se desarrolló la aplicación software que permite realizar tareas de poliscopio.

- El poliscopio es un equipo invaluable debido a su antigüedad, calidad de sus componentes, sus capacidades y aplicaciones. Empresas como RadioMuseum restauran este tipo de equipos para conservarlos en funcionamiento, por lo cual sería importante para la facultad contar con un equipo como el poliscopio funcionando completamente.

- La implementación de un sistema de análisis de señales tipo poliscopio sobre FPGA es un proyecto totalmente viable una vez superadas las limitaciones de tipo hardware, es decir, si se pudiera contar con una mejor FPGA que incorpore capacidades dedicadas al procesamiento digital de señales como una unidad DSP y un módulo ADC de alta frecuencia. Las ventajas de una implementación sobre FPGA radican en su bajo costo y la flexibilidad en su desarrollo. Xilinx cuenta con FPGA como la Genesys Virtex®-5 la cual está alrededor de 449 dólares para entornos académicos, o la Atlys Spartan®-6 cuyo costo para entorno académico es 199 dolares. (fuente: http://www.digilentinc.com).

BIBLIOGRAFIA.

[1] ROHDE & SCHWARZ, INSTRUCTION BOOK OF POLYSKOP TYPE SWOB 1942.

[2] GAVIRIA WILLIAM, MEJIA HUGO. POLISCOPIO DE AUDIOFRECUENCIA. Universidad del Cauca. Popayan-1990

[3] http://www.radiomuseum.org/r/rohde_polyskop_i_swob.html. [Consultado: Agosto 12, 2010].

[4] CLIVE MAXFIELD. THE DESIGN WARRIOR'S GUIDE TO FPGAS: DEVICES, TOOLS AND FLOWS. Elsevier–Newnes. Primera edición 2004. Página 2.

[5] http://www.wikipedia.com/FPGA . [Consultado: Agosto 10, 2010]

[6] HAUCK SCOTT, DEHON ANDRÉ. RECONFIGURABLE COMPUTING: THE THEORY AND PRACTICE OF FPGA-BASED COMPUTATION. Elsevier. 2008. Página 7.

[7] CLIVE MAXFIELD. FPGAS: WORLD CLASS DESIGNS. Elseiver 2009.

[8] UWE MEYER BAESE. DIGITAL SIGNAL PROCESSING WITH FIELD PROGRAMMABLE GATE ARRAYS. Springer. Tercera edición. 2007

[9] OPPENHEIM A. WILLSKY A. SEÑALES Y SISTEMAS. Segunda Edición. Prentice hall. 1998.

[10] http://www.xilinx.com/publications/prod_mktg/TDP_DSP_Product_Brief.pdf. [Consultado: Agosto 14, 2010]

[11] J. VILLASENSOR, B. HUTCHINGS "THE FLEXIBILITY OF CONFIGURABLE COMPUTING," IEEE SIGNAL PROCESSING MAGAZINE. Página 67-84. 1998

[12] SPARTAN-3A FPGA FAMILY: DATA SHEET, Xilinx Inc. Disponible en http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf. [Consultado: Agosto 15, 2010].

[13] SPARTAN-3A FPGA STARTER KIT BOARD USER GUIDE, Xilinx Inc. Disponible en http://www.xilinx.com/support/documentation/boards_and_kits/ug330.pdf [Consultado: Agosto 15, 2010].

[14] GLYN JAMES, DAVID BURLEY. MATEMÁTICAS AVANZADAS PARA INGENIERÍA. Segunda Edición. Pearson. 2002. Página 361.

[15] ALBERT H. KAISER. DIGITAL SIGNAL PROCESSING USING THE FAST FOURIER TRANSFORM (FFT). Grin Verlag. 1997.

[16] PETER J. BROCKWELL, RICHARD A. DAVIS. TIME SERIES: THEORY AND METHODS. Segunda Edición. Springer 2006. Página 373

[17] KUO S. LEE B. REAL TIME DIGITAL SIGNAL PROCESSING-2001

[18] E. O. BRIGHAM. THE FAST FOURIER TRANSFORM AND ITS APPLICATIONS, Prentice Hall Signal Processing Series, Englewood Cliffs, NJ 1988.

[19] SMITH STEVEN W. THE SCIENTIST AND ENGINEER'S GUIDE TO DIGITAL SIGNAL PROCESSING. California Technical Pub. 1997.

[20] FAST FOURIER TRANSFORM IP CORE V7.1, Xilinx Inc. Disponible en http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf. [Consultado: Agosto 15, 2010].

[21] PETER J. ASHENDEN. THE DESIGNER'S GUIDE TO VHDL. 2nd Edition. Editorial Morgan Kaufman, 2002

Anexo A. Entorno de Desarrollo ISE Design Suite 12.3, creación de un proyecto, generación de CORES y programación del dispositivo.

Entorno de Desarrollo ISE Design Suite 12.3

El software ISE™ Design Suite 12.3 (Integrated Software Environment).Es una herramienta de automatización de diseño electrónico o EDA[24] que facilita solución de problemas complejos en dispositivos reconfigurables como la FPGA. El ISE integra el *Project Navigator* que es la interfaz gráfica de usuario que permite el acceso a todos los componentes del proyecto. Los diseños de usuario se pueden introducir mediante diferentes formatos: los esquemáticos, los grafos de estados y las descripciones hardware en VHDL o Verilog. Una vez compilados los diseños se puede simular su comportamiento a nivel funcional o a nivel temporal. A nivel funcional no tiene en cuenta los retardos provocados por el hardware y a nivel temporal se simula el diseño teniendo en cuenta cómo se va a configurar el hardware.

Para realizar un proyecto en el ISE, una vez se tiene un diseño definido, se especifica este diseño, es decir, se crea o agrega componentes al proyecto. Luego de esta etapa se realiza la síntesis mediante el módulo XST[25] que compila el diseño para transformar las fuentes HDL en una arquitectura específica de diseño lista de red o netlist.

Después de esta etapa viene la parte de simulación y validación de la implementación aunque en cualquier momento se puede verificar la funcionalidad del diseño utilizando la herramienta de simulación. Esta herramienta integrada con el ISE se denomina *ISim*. Sin embargo, es posible integrar otros simuladores como *ModelSim*.

Luego de la síntesis se realiza la implementación del diseño, etapa en la cual se traduce la descripción de hardware en rutas y conexiones de circuitos lógicos y se realiza el mapeo de estas rutas.

Después se procede a generar el archivo de programación para la FPGA, este archivo puede ser descargado en un dispositivo seleccionado.

Usando el Proyect Navigator se puede modificar las propiedades de los procesos para controlar la implementación y así optimizar el diseño. Para tratar de cumplir los objetivos de diseño más rápido, se puede utilizar *SmartXplorer* para automatizar múltiples implementaciones corriendo con diferentes propiedades en los procesos.

[24] EDA: Electronic Desing Automation - Automatización del Diseño electrónico.
[25] XST-Xilinx Synthesis Technology : La Tecnología de Síntesis Xilinx es una herramienta de Xilinx, que sintetiza diseños HDL para crear archivos netlist específicos de Xilinx llamados NGC. El archivo NGC es una lista de red que contiene los datos de diseño lógico y restricciones.

Después de la implementación se muestra la utilización de los recursos del dispositivo, rendimiento temporal y estado de la implementación y se guardan los resultados en archivos .log de informes.

Creación de un proyecto

A continuación se presenta una guía de cómo generar y compilar un proyecto, así como el procedimiento para programar el dispositivo lógico.

1. Se abre el ISE Project Navigator siguiendo la ruta donde se ha instalado o mediante inicio->archivos de programa->Xilinx ISE Design Suite 12.3

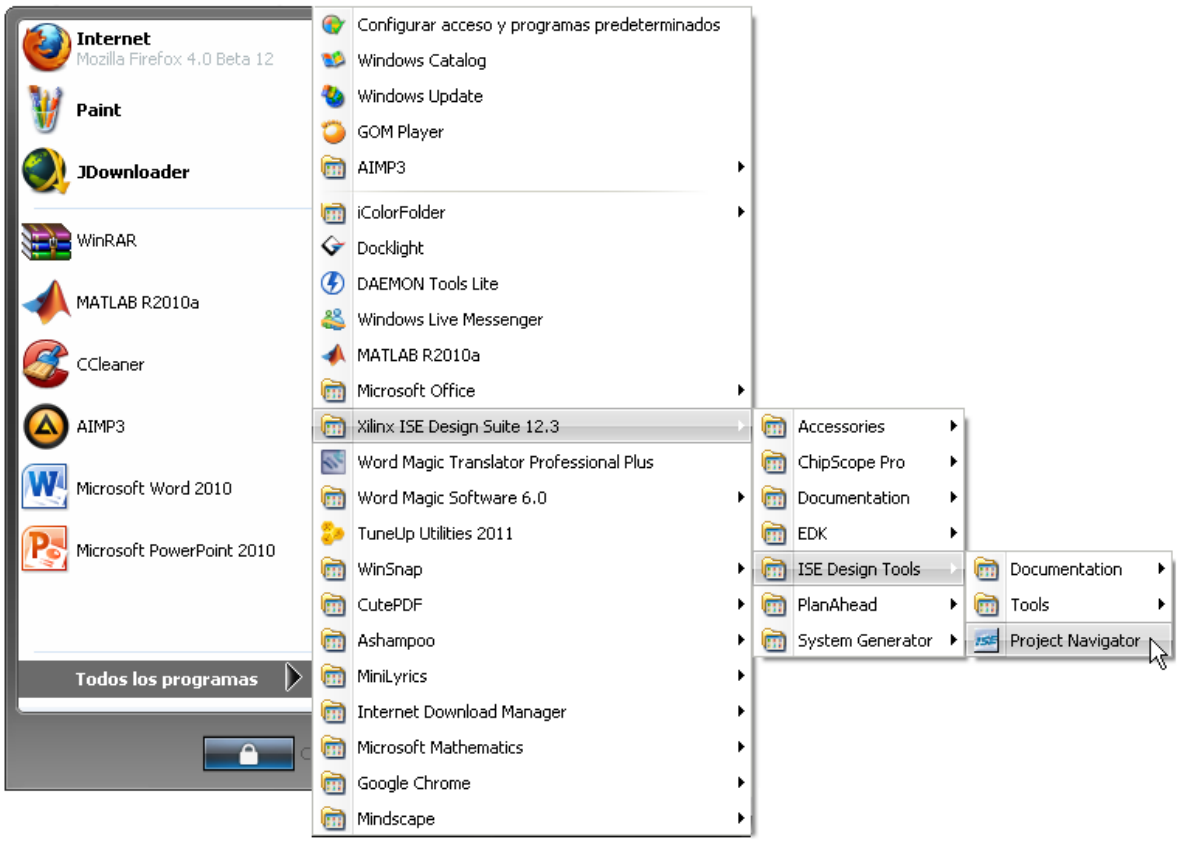

Figura A.1. Abrir ISE Project Navigator.

2. Se crea un nuevo proyecto en File-> New Project

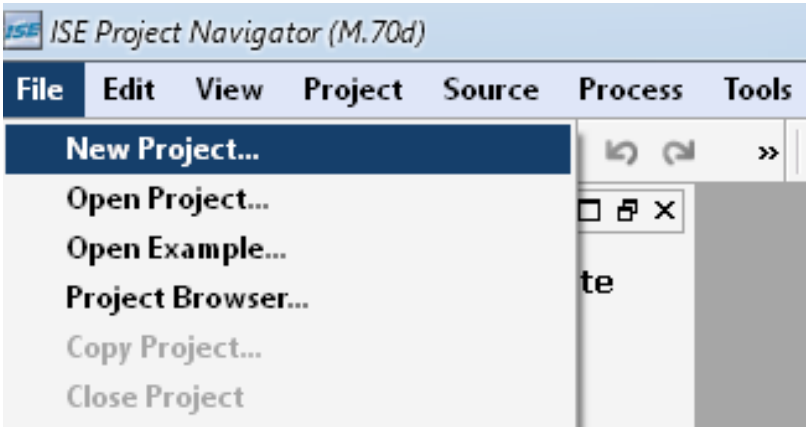

Figura A.2. Nuevo proyecto en el ISE Project Navigator

3. En la ventana que aparece, se especifican el nombre del proyecto y la localización de los archivos fuente. También se define el tipo de archivo fuente para la entidad de alto nivel. Se completan los campos y se da click en next.

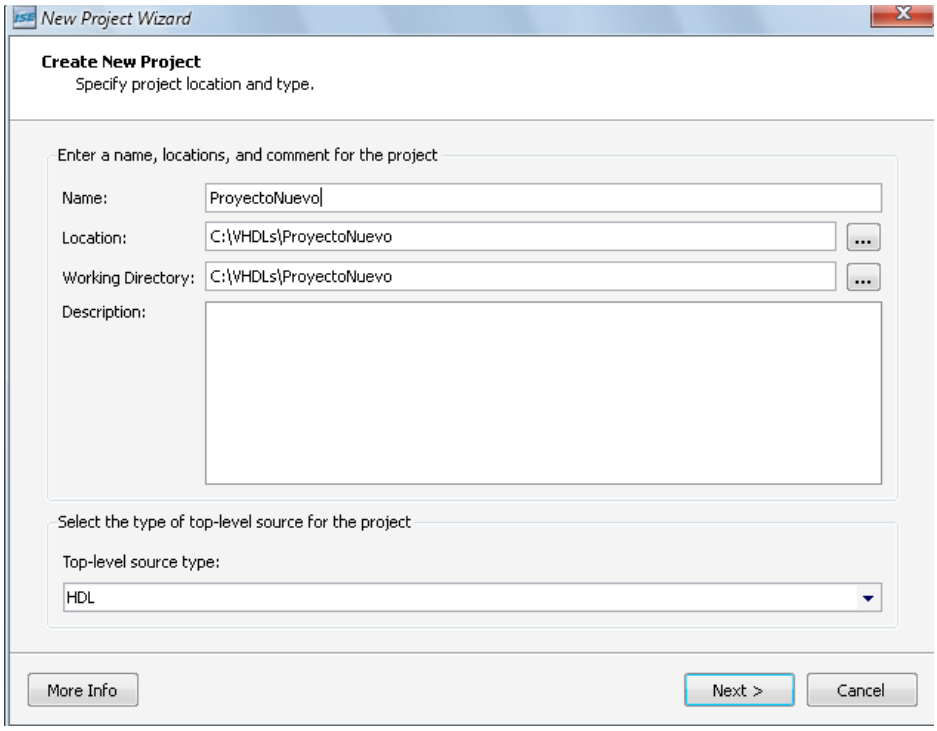

Figura A.3. Definir ubicación del Nuevo proyecto.

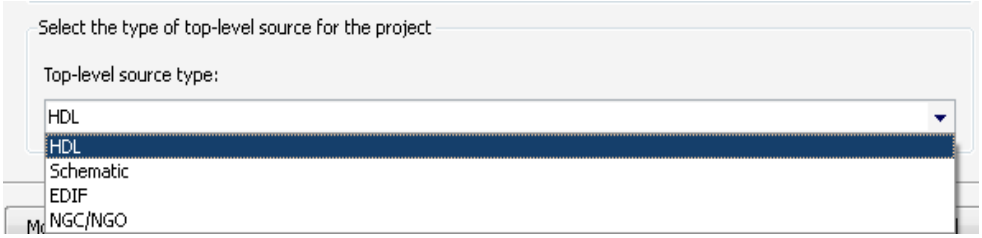

Figura A.4. Definir tipo de fuente para la entidad Top-Level.

4. En la ventana que aparece se definen los datos del dispositivo que se va a utilizar, especificando la familia, dispositivo FPGA y referencia de paquete. Llenamos los datos y damos click en next. En la figura A.5 se muestra la configuración utilizada para la Spartan 3A Starter Kit.

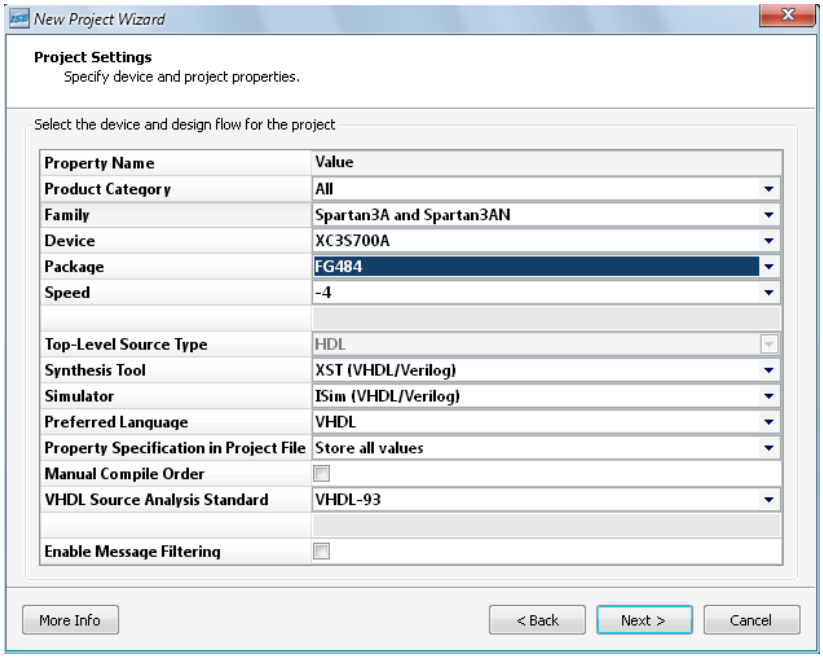

Figura A.5. Definir tipo datos del dispositivo programable.

5. En la ventana que aparece a continuación se muestra el resumen de la configuración del nuevo proyecto. En la figura A.6 se muestra el resumen para el caso de la Spartan 3A Starter Kit. Se da click en finalizar y con esto termina la creación de un nuevo proyecto.

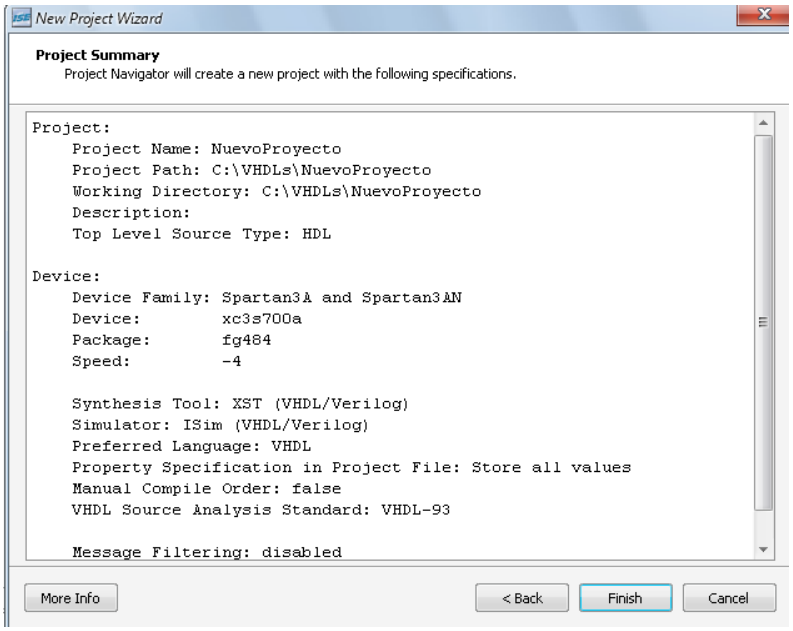

Figura A.6. Resumen de configuración de nuevo proyecto.

6. En este punto es necesario agregar fuentes al proyecto. Para esto se va a la ventana hierarchy o jerarquía y se da click derecho en el nombre del dispositivo elegido en la configuración. Aparecerá un menú en el cual se pueden agregar nuevos archivos al proyecto o archivos ya existentes. En este caso agregaremos un archivo nuevo.

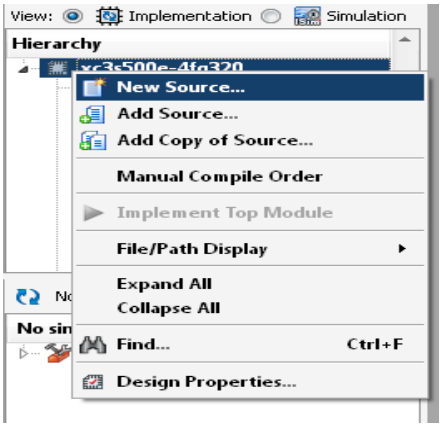

Figura A.7. Añadir nuevo archivo al proyecto

7. Se selecciona el tipo de archivo que queremos añadir. En este punto se puede añadir un módulo VHDL, un esquemático o un IPCORE, que son los CORE de propiedad intelectual de Xilinx Inc. En este caso queremos añadir un componente así que elegimos VHDL Module. Se selecciona el nombre del archivo y se da click en next.

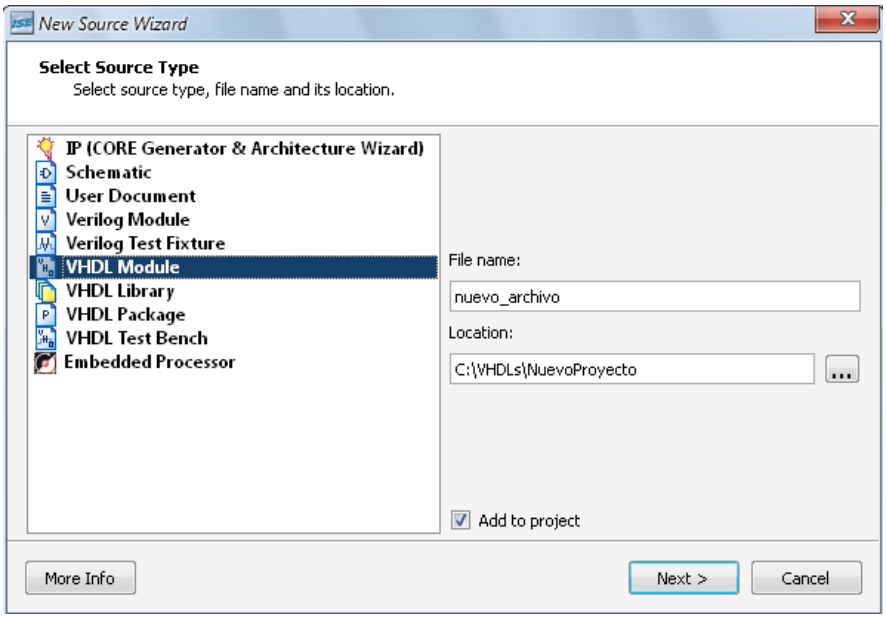

Figura A.8. Nombre y Tipo del nuevo archivo

8. En la ventana que aparece a continuación se elige el nombre de la entidad, el nombre de la arquitectura, los nombres de los puertos del componente, la dirección de esos puertos (in/out), si se trata de un bus se marca la casilla de bus y se especifica su tamaño. En la figura A.9 se muestra una configuración de puertos para el archivo nuevo. Una vez configurado lo anterior se da click en siguiente.

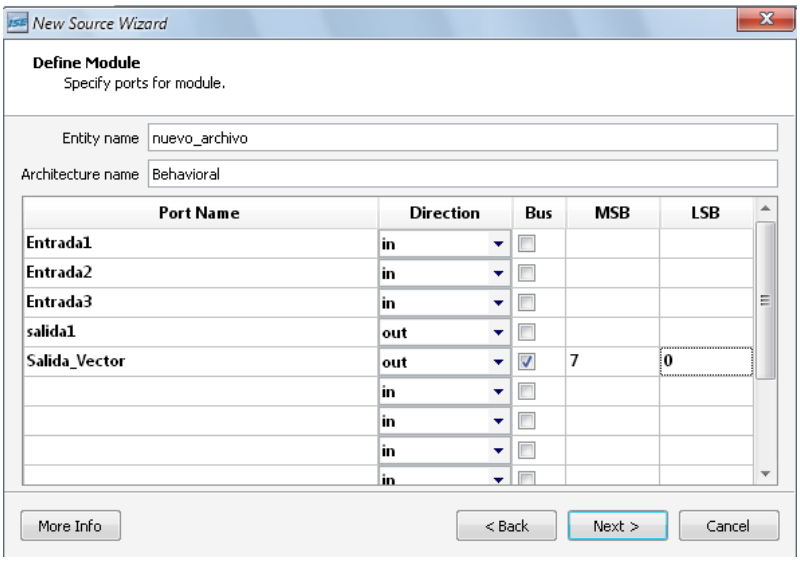

Figura A.9. Nombre de la entidad, Arquitectura y Puertos

9. En la ventana siguiente aparece un resumen, se da click en finish y ya está, un nuevo módulo de VHDL con la configuración de puertos se añade a nuestro proyecto.

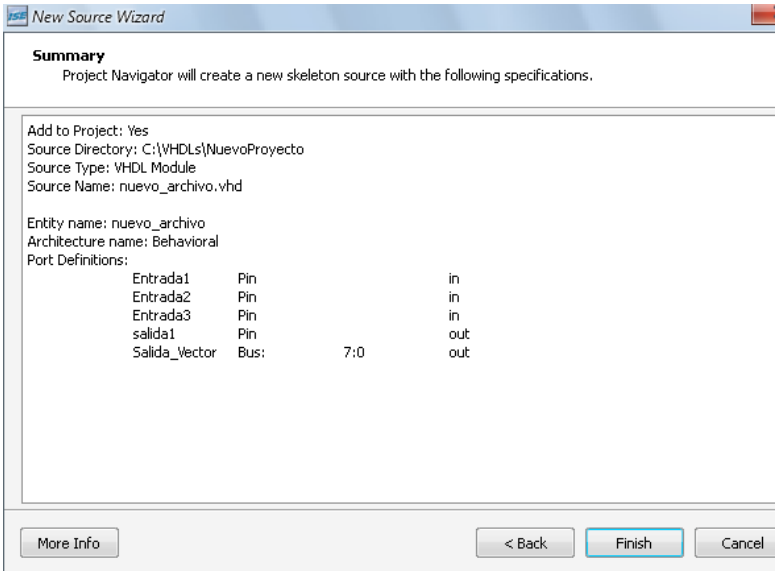

Figura A.10. Resumen, añadir nuevo archivo.

10. Podemos observar en la figura A.11 que el archivo generado contiene la configuración de puertos que se introdujo para el nuevo archivo. Sin embargo esta configuración se puede cambiar en cualquier Momento.

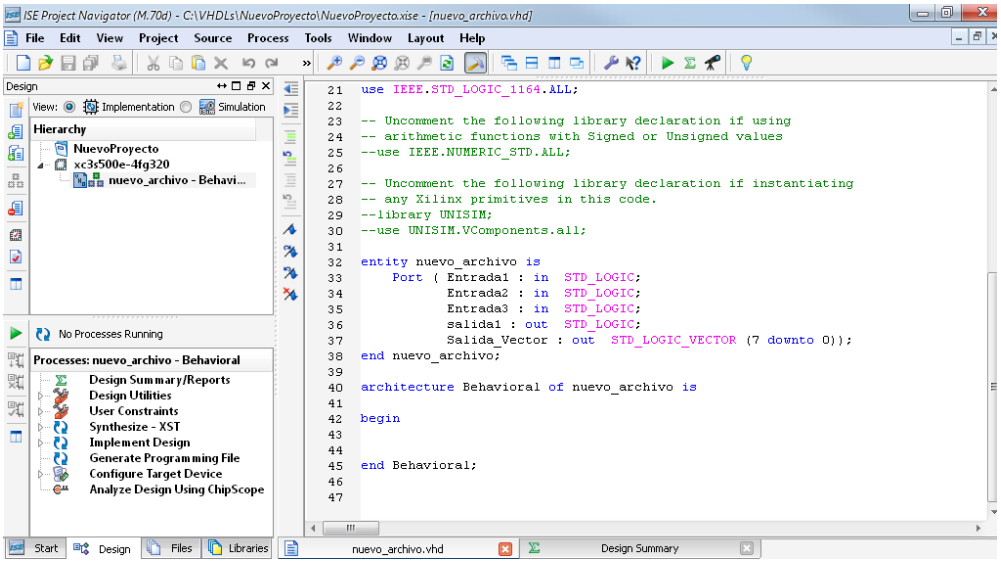

Figura A.11. Añadido nuevo archivo.

Generación de Cores

A lo largo de este documento se menciona que algunos bloques funcionales de la implementación del sistema de análisis de señales de tipo poliscopio se realizaron adecuando CORES suministrados por la Suite ISE de Xilinx Inc. cuya propiedad intelectual pertenece a Xilinx.

El procedimiento para generar estos CORES utilizando el software CORE GENERATOR de Xilinx, el cual forma parte la suite ISE de Xilinx.

Para generar un CORE se procede de la siguiente manera:

1. En el ISE projec Navigator, en el menú Tools de la barra de Menú, se selecciona la opción CORE Generator. Esto se muestra en la figura A.11.

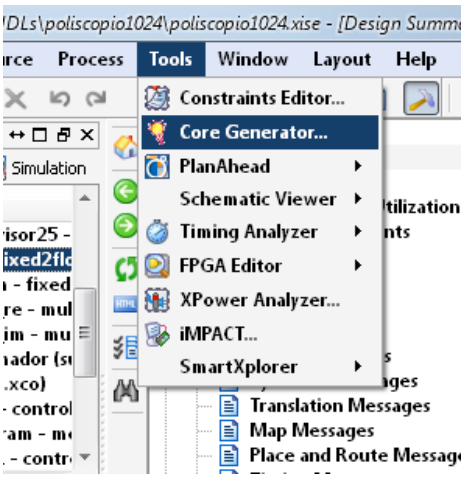

Figura A.11. Generando un CORE, paso 1.

2. En la ventana que se muestra, de la lista que aparece a la izquierda, se selecciona el CORE que se quiere generar. En este caso se selecciona la FFT. nótese que los CORES que no son soportados por el dispositivo aparecen en color gris. En la figura A.12 se muestra la ventana de elección de CORE con la selección hecha.

Figura A.12. Generando un CORE, paso 2

3. Una vez seleccionado el CORE se da doble click y se espera a que cargue la interfaz de configuración para ese CORE. Se realiza la respectiva configuración, se sigue el asistente que se muestra en la figura A.13. y se da click en Generate una vez se han configurado todas las opciones. Es importante señalar que para más información sobre el CORE se debe consultar su datasheet haciendo click en el menú Documents y en el submenú Datasheets.

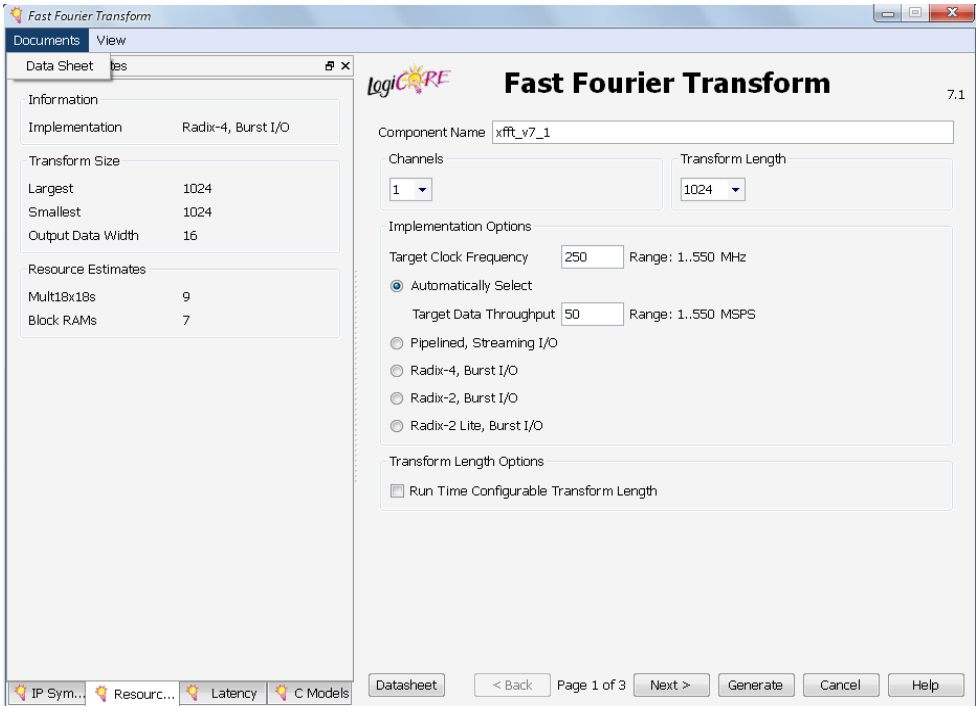

Figura A.13. Generando un CORE, paso 3

Programación del dispositivo

Para programar el dispositivo es necesario generar el archivo de programación a partir de la síntesis del código VHDL. El proceso de programación se realiza una vez se ha llevado a cabo la síntesis, se ha implementado el proyecto, es decir, traducido, mapeado y trazado rutas sobre los bloques lógicos.

1. El primer paso consiste en generar el archivo de programación mediante la opción Generate Programing File, que se ubica en la sección Design del ISE. En la figura A.14 se muestra una captura de pantalla para ilustrar este proceso.

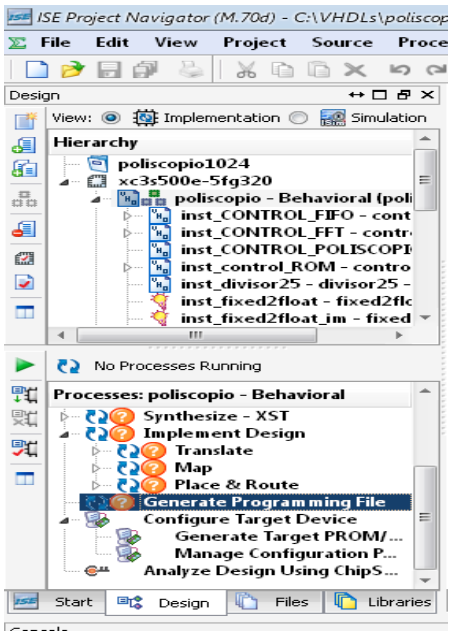

Figura A.14. Programación del Dispositivo. Generando el archivo de programación.

2. Se da click derecho en la opción Generate Programing File y luego se da click en Run. se espera a que el proceso haya concluido satisfactoriamente. Esto se indica con un símbolo de aprobación verde que remplaza al símbolo de pregunta, cuando el archivo contiene warnings el símbolo es un signo de admiración amarillo pero de todas formas el archivo de programación ha sido generado. . En la figura A.15 se muestra un gráfico para ilustrar este paso.

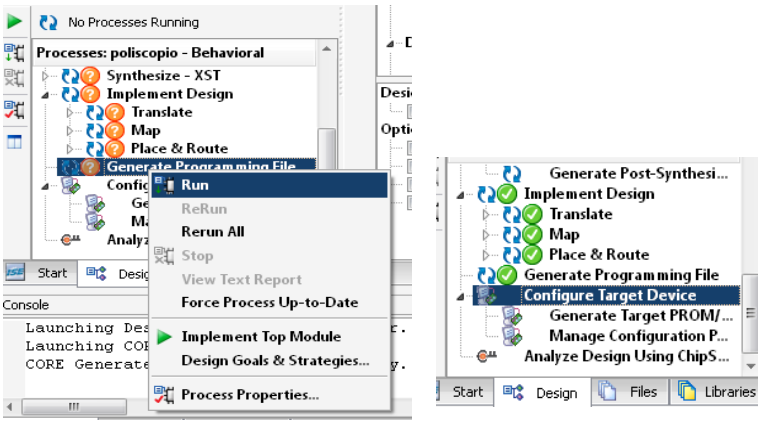

Figura A.15. Programación del Dispositivo. Archivo de programación Generado

3. Una vez el archivo de programación ha sido generado, se procede a configurar el dispositivo utilizando la opción Configure Target Device. En la figura A15. Se muestra donde se ubica esta opción. Cuando se da doble click aquí, se abre la herramienta ISE IMPACT que sirve para programar el

dispositivo. En este punto es necesario que el dispositivo esté encendido y conectado al computador.

4. En el ISE IMPACT se selecciona la opción Boundary scan para que el programa cargue un escenario de archivo nuevo donde con un click en un menú contextual el programa encuentra automáticamente el dispositivo programable. En las figuras siguientes se muestra este proceso.

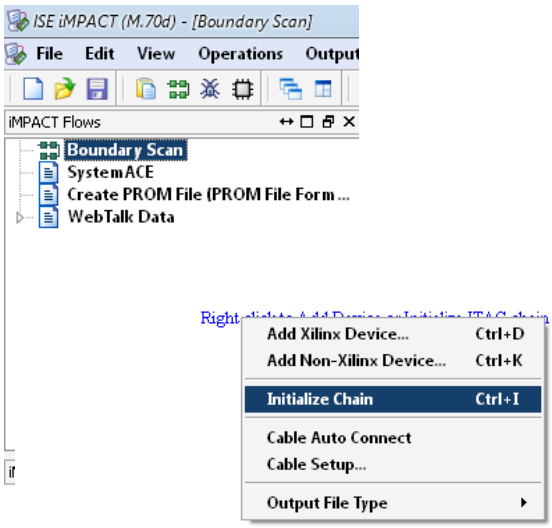

Figura A.16. Programación del Dispositivo. IMPACT, detectar dispositivo

5. En este punto, en la pantalla que aparece, es posible seleccionar que se va a programar la FPGA y cargar el archivo de programación que es de extensión .bit y apretar el boton de program. Después de unos segundos el dispositivo queda programado y se muestra un mensaje que así lo indica. En la figura A.17 y A.18 se puede ver este proceso.

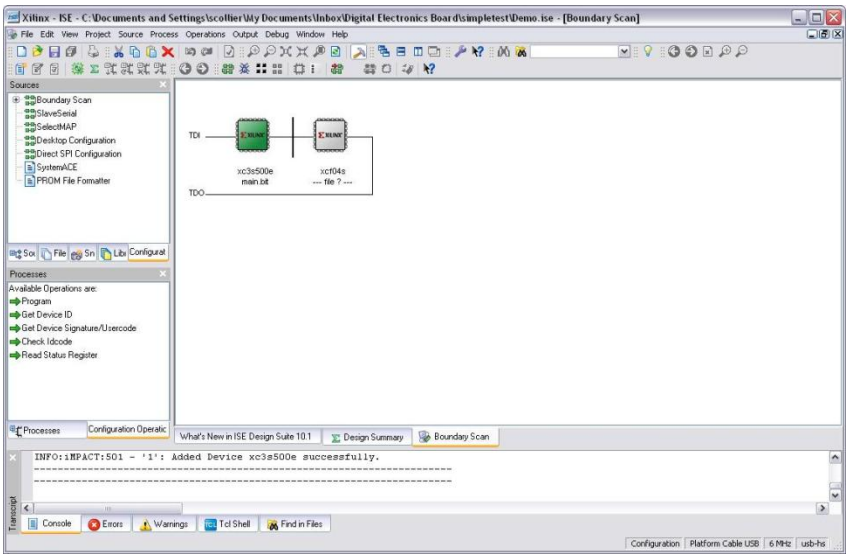

Figura A.17. IMPACT, programar dispositivo.

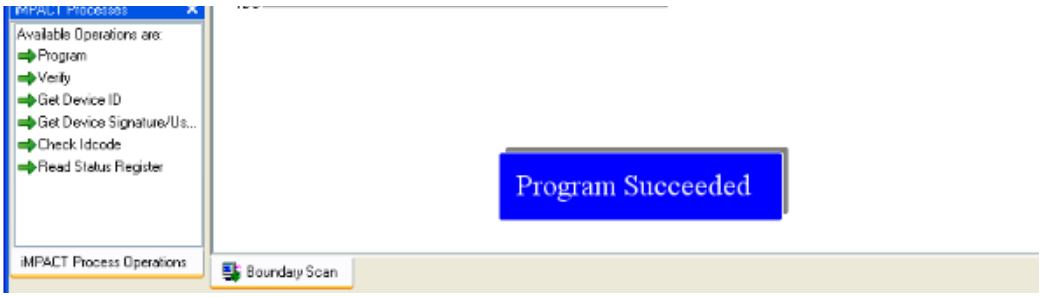

Figura A.18. IMPACT, Dispositivo programado.