UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INFORMÁTICA

"Metodología para la Implementación de Métodos Numéricos en Hardware"

Tesis

Que como parte de los requisitos para obtener el Título de

INGENIERO EN COMPUTACIÓN

PRESENTA

ALEJANDRA MIGUEL VARGAS MANDUJANO

DIRECTOR DE TESIS:

Dr. JUAN MANUEL RAMOS ARREGUÍN

Santiago de Querétaro, Qro. , Febrero de 2013

A mi madre,

por enseñarme a nunca darme por vencida.

A mi novio,

por toda la paciencia y apoyo en esta etapa.

A mi director de tesis,

por confiar en mi, tenerme paciencia y guiarme en este trabajo.

AGRADECIMIENTOS

Muchas veces crees que hay tantos obstáculos en tu vida que llegas a decir “creo que ya no puedo más”, entonces es cuando llegan personas que te alientan y te dicen “Tú puedes, no te rindas tan fácil, cree en ti”, es así como durante este trabajo estuvieron a mi lado personas muy importantes que hicieron que nunca me diera por vencida, que estuvieron en los buenos y malos momentos y quienes siempre creyeron en mí.

Mami es para ti mi más grande agradecimiento por permitirme llegar a esta etapa tan importante de mi vida, siempre fuiste un digno ejemplo a seguir por ser una mujer valiente y guerrera, que a pesar de todos los obstáculos que se te presentaron en la vida seguiste adelante, me enseñaste a nunca darme por vencida, a luchar y dar lo mejor de mí siempre. Gracias mami por todo el apoyo, y todo el amor que me has dado.

A ti Toño por estar a mi lado, porque a pesar de que muchas veces me desespere tu siempre me apoyaste y creiste en mi, por tenerme paciencia y por ayudarme durante todo este trabajo, gracias por todo tu cariño, y por convertirte en una parte muy importante de esto.

A mis amigos que me dejaron entrar en su vida, que compartieron conmigo muchísimas cosas, por escucharme y darme su apoyo incondicional, gracias a cada uno por siempre alentarme a seguir.

A mi director de tesis que creyó en mí y tuvo la confianza de trabajar conmigo, gracias por el apoyo que me dio, por todo su conocimiento y tiempo que me aporto durante el trabajo realizado.

Gracias a todos.

I N D I C E

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE DE FIGURAS

INDICE DE FIGURAS

CAPÍTULO I

Introducción

1.1 Estado del Arte

Al momento de aplicar las Matemáticas a situaciones del mundo real, se encuentra a menudo con problemas que no pueden ser resueltos analíticamente o de manera exacta y cuya solución debe ser abordada con ayuda de algún procedimiento numérico. Por ejemplo, se sabe que no todo sistema de ecuaciones puede resolverse fácilmente, al menos desde el punto de vista analítico, pero se han desarrollado herramientas de cómputo para resolver este tipo de problemas. Al conjunto de herramientas de cómputo desarrolladas para resolver problemas numéricos, se les llama Métodos Numéricos, y al ser resueltos por un sistema digital como una computadora, un microprocesador, microcontrolador, etc., la solución se puede obtener mucho más rápido que si se resolviéra de manera manual.

Los Métodos Numéricos son un conjunto de técnicas que se utilizan para la solución de problemas de la vida real, con solo usar un número finito de operaciones aritméticas, las cuales se encargan de aplicar algoritmos para obtener una solución aproximada de los mismos.

El gran poder del cómputo que se tiene en estos días es una gran ventaja para poder resolver problemas más complejos, los cuales pueden ayudarnos a tener una mejora en la calidad de vida del hombre. Así, encontramos muchas aplicaciones de los Métodos Numéricos en el trascurrir de la vida diaria, desde los tecnológicos como es la ingeniería estructural o la aerodinámica de aviones, hasta aplicaciones más sofisticadas como lo son ingeniería en alimentos, ingeniería medica, diseño de fármacos, biología, entre otros (Rionda, 2004).

Gracias a la evolución que han tenido en la actualidad los métodos numéricos, así como el desarrollo de los equipos de cómputo, es posible modelar el choque de un vehículo o hacer el análisis aerodinámico estructural de un avión,

resolviendo cada sistema algebraico de ecuaciones, con varios cientos de miles de incógnitas.

Por ejemplo, hablando de la aplicación de los métodos numéricos en lo que son diversos problemas de la ingeniería podríamos decir que se encuentran en:

- Mecánica de solidos, fluidos.
- Medios de transporte (concepción y producción, ya sea avión, automóvil o barco).
- Procesamiento digital de señales (voz, sonido, imágenes, video, datos).

Por lo tanto, los Métodos Numéricos y sus aplicaciones computacionales, nos ayudan a resolver diversos problemas físicos de una manera muy eficiente y gracias al uso de los mismos cada día los resultados son más cercanos a la realidad.

Existen también aplicaciones de los Métodos Numéricos en la resolución de problemas matemáticos, los cuales se plantean a partir de la modelización matemática de fenómenos o procesos fisicoquímicos; específicamente se aplican a la simulación de procesos químicos, que permite realizar de forma automática cálculos sobre balances de materia, balances de energía y propiedades físicas y químicas (Cruz, 2011).

En los modelos de sistemas continuos que contienen gran número de grados de libertad y al ser discretizados generan sistemas de ecuaciones lineales que llegan a ser de gran tamaño, lo que da lugar a que el tratamiento de sistemas requieran de recursos computacionales más avanzados, como lo es la ayuda de los Métodos Numéricos en particular el método de descomposición de dominio de subestructuración precondicionado, para la resolución de ecuaciones diferenciales parciales elípticas en equipos paralelos con memoria distribuida y mostrar las

ventajas de lo como se puede encontrar una solución rápida y optima de estos modelos (S. Botello, 2007).

Dentro de la ingeniería mecánica y el diseño mecánico, podemos encontrar una aplicación importante de los métodos numéricos en lo que es el campo de las vibraciones. El estudio de las vibraciones se refiere a lo que son los movimientos oscilatorios de los cuerpos y a las fuerzas asociadas a ellos (Guerrero, 2008).

Existe también la aplicación de algunos métodos numéricos en cuanto a los elementos ópticos difractivos en longitudes de ondas ópticas y poder obtener su optimización computacional, para realizar con ellos los análisis electromagnéticos en el dominio temporal (Francés, 2011).

Otra aplicación de los métodos numéricos, esta en la aproximación de una función por un polinomio y estimar el error de truncamiento, esto se realiza con ayuda de la aplicación del teorema de Taylor (Tischer, 2009).

Se tiene también la ayuda de la Implementación de algoritmos como lo son la Transformada Rápida de Fourier (FFT), el cual nos ayuda a realizar un proceso donde se pueden representar fielmente la composición frecuencial de las señales de comunicaciones, por lo cual optimiza la operatoria necesaria para la evaluación numérica cuya ventaja principal radica en la rapidez de conseguirla (Vidal D., 2007).

Este trabajo de tesis requiere del estudio de dos áreas de investigación las cuales son: Métodos numéricos que hemos mencionado con anterioridad y Cómputo Reconfigurable.

El Computo Reconfigurable se basa especialmente en los dispositivos lógicos reprogramables, especialmente en los FPGA's (Field Programmable Gate Array), los cuales son dispositivos semiconductores programables que se basan en una matriz de bloques lógicos configurables (CLB) conectados a través de interconexiones programables. A diferencia de los circuitos integrados de aplicación específica (ASIC), que esta hecho a la medida para un uso en particular, en vez de ser para propósito general (XILINX ALL PROGRAMMABLE, 2012).

El principal método de programación de los FPGAs es utilizando lenguajes descriptivos de hardware, como es VHDL (very-high-speed integrated circuits hardware description language), el cual proporciona amplias facilidades para la realización de algoritmos.

En la Tabla 1.1 se muestran algunas de las aplicaciones en las cuales ya se han implementado los métodos numéricos en ciertas áreas donde como antes mencionamos gracias a ellos podemos tener una mejor aproximación de muchos problemas físicos de una manera eficiente y muy cercanos a la realidad.

Como ejemplo, se tiene la aplicación de estos en cuanto al procesamiento de imágenes el cual nos ayuda a tener una mejor calidad de las mismas o también nos da la facilidad de búsqueda de información comparado con el tiempo que se toma en el procesamiento en una computadora personal.

También se pueden aplicar en el procesamiento de señales que no es más que una manipulación matemática de una señal de información ya sea para modificarla o mejorarla en algún aspecto, este se caracteriza por tener la representación en el dominio del tiempo discreto o representación en el dominio de frecuencia discreta.

Tabla 1.1 Aplicaciones de métodos numéricos en FPGA

Aplicación	Descripción (Breve Resumen):
Implementación en FPGA de Ruido Gaussiano para simulaciones en Hardware	El canal del ruido blanco Gaussiano (AWGN) es un modelo de canal universal para analizar los esquemas de modulación. Los métodos numéricos generan muestras de variables aleatorias uniformes y se aplican después a los algoritmos para obtener muestras Gaussianas con media cero y varianza unidad (Micco, 2010).
Hardware accelerated montecarlo financial simulation overlow cost FPGA cluster Parallel& Distributed Processing.	Implementación del método Montecarlo para resolver problemas matemáticos mediante el muestreo aleatorio de variables aleatorias, para acelerar cálculos de problemas gravamen financiero (International, 2009).
Diseño e Implementación en FPGA del Método Recursivo Gram-Schmidt	Implementación de un algoritmo de ortogonalización que utiliza el teorema de GRAM-SCHMIDT para la compensación del desbalanceo que se produce en las transmisiones de la Quadrature amplitude modulation (QAM) debido a las diferentes interferencias provocadas por los componentes del sistemas tales como los convertidores (A/D) o (D/A), interferencia del canal (ruido), etc. (Logroño, 2011).
Implementación de la FFT en hologramas de Fourier generados con FPGAs	Implementación de la Transformada Rápida de Fourier para la construcción de hologramas digitales con las FPGAs y reconstrucción de los hologramas de Fourier. Simulación del proceso y programación en VHDL para la síntesis del FPGA (Castillo, 2006)
Algoritmos de procesamiento de imágenes en FPGA	Implementación de varios algoritmos en un FPGA para realizar el procesamiento de imágenes y reconocimiento de objetos dentro de dichas imágenes (Quintero, 2006).
Integración Numérica con Redes Neuronales	Aplicación de Algoritmos basados en Redes Neuronales Artificiales para la solución de integrales definidas, con una mejor aproximación y precisión (Christhofer, 2010).
Direct Digital Frequency Synthesizer with CORDIC Algorithm and Taylor Series Approximation for Digital Receivers	Se presenta un nuevo enfoque para el diseño de una técnica de sintetizador de frecuencia digital directa (DDFS) para la demodulación compleja utilizado en receptores digitales (Maher, 2009).
Prototipado en FPGAs para inyección de fallas. Aplicación de sistemas distribuidos sobre bus CAN.	La inyección de fallas es una herramienta útil para depurar los mecanismos de tolerancia a fallas durante la fase de desarrollo, se utiliza para analizar el comportamiento de un modelo o un prototipado de algún sistema en especial circuitos electrónicos frente a fallas provocadas artificialmente y aplicando a las FPGAs para acelerarla inyección de fallas en sistemas electrónicos digitales (P.Julio, 2005).
Implementación de Funciones Elementales en Dispositivos FPGA.	Diseño eficiente de funciones elementales para su uso en sistemas de comunicaciones, para la implementación en dispositivos FPGA de las funciones elementales log_2y atan($^{y}/_{x}$) (Mazón, 2011).

Tabla 1.1 (Continuación)

Aplicación	Descripción (Breve Resumen):
Procesamiento de Imágenes de Mastografía con Implementación en Hardware.	Implementación del procesamiento digital de señales en el hardware reconfigurable (FPGA) para aumentar la velocidad de procesamiento de una imagen de mastografía comparado con el tiempo que se toma en el procesamiento en una computadora personal (Domínguez, 2009).
Implementación de Arquitecturas para el cálculo de funciones trascendentales empleando el algoritmo CORDIC en un FPGA.	CORDIC (COordinate Rotation Digital Computer),se basa en rotaciones y desplazamientos, operaciones que presenta el algoritmo que implementa eficientemente las funciones trigonométricas necesarias para la navegación aérea en tiempo real y empleando pocos recursos en hardware (FPGA), (Ríos, 2006).
Three Dimensional Reconstruction Strategies Using a Profilometrical Approach base on Fourier Transform.	El resultado del procesamiento se utiliza para varios tareas como son en: el radar de apertura sintética (SAR), Imágenes de resonancia magnética (MRI), en la navegación de robot en 3D, etc. (Pedraza et al., 2011).

1.2 Objetivos

El objetivo principal de esta Tesis es desarrollar un método para la implementación de algunos algoritmos basados en métodos numéricos que nos ayuden a resolver procedimientos matemáticos, poder encontrar una solución más exacta y cercana a la realidad. Poder aplicar en hardware, utilizando tecnología de Dispositivos Lógicos Programables, así como tarjetas de desarrollo basados en FPGA, para obtener sistemas digitales capaces de resolver un algoritmo en menos tiempo que una PC convencional.

1.3 Problemática

La manera tradicional como se han implementado los métodos numéricos es en una PC, lo cual implica que difícilmente se va a poder realizar un análisis de una señal en tiempo real, debido al tiempo que maneja la computadora, por lo cual tenemos que encontrar una manera de como poder realizar este análisis en el menor tiempo posible y obtener un buen resultado.

Por tanto podemos decir que es mejor usar dispositivos lógicos programables, en este caso el uso de un FPGA el cual permite la implementación en paralelo de uno o más algoritmos matemáticos, con tiempos de cómputo que van desde un ciclo de reloj hasta algunos ciclos de reloj y gracias a esto la ventaja es que se tiene un resultado a menor tiempo que una PC convencional, de tal manera que es posible el procesamiento de señales en tiempo cuasi-real y con mejores resultados que los obtenidos en una PC.

Para comprobar esto se tendrá la ayuda del software Matlab, el cual será una de las herramientas principales para poder comprobar los resultados obtenidos y con esto poder desarrollar herramientas que nos ayuden en el análisis y procesamiento de señales y mostrar las ventajas de implementar los algoritmos matemáticos en hardware a diferencia del software.

1.4 Síntesis de la tesis.

En el capitulo 2, se habla de teoría que ayudaran a comprender algunos de los conceptos mencionados a lo largo de la Tesis como lo son: el lenguaje descriptivo de Hardware (VHDL), que es un FPGA el cual será una de las partes mas importantes del trabajo ya que dentro de este dispositivo se implementara cada uno de los métodos numéricos, se definirá que son los métodos numéricos, que son las Series de Taylor las cuales son de mucha importancia en el estudio de los métodos numéricos, que es la Transformada de Fourier y dentro de esta se mencionara la Transformada Discreta de Fourier y la Transformada Rápida de Fourier, así como también se hablara de la herramienta de software MatLab.

En el capitulo 3, se realiza el desarrollo de las Series de Taylor, una vez analizadas se pasara al estudio de la Transformada de Fourier basándose principalmente en la Transformada Rápida de Fourier (FFT) la cual tiene gran importancia en una amplia variedad de aplicaciones, desde la resolución de ecuaciones diferenciales, tratamiento digital de señales, tratamiento digital de

imágenes, reducción de ruido en señales, entre otros. En este capitulo se muestra como es que se realizó la implementación de los métodos numéricos ya mencionados anteriormente y la implementación en MatLab que ayudo en la comprensión y comprobación de como es que trabajan estos métodos y de como se realizó de un manera satisfactoria la implementación de los mismos de software a hardware.

En el capitulo 4, se muestra cada una de las pruebas y resultados de las implementaciones en hardware de la Serie de Taylor que es específicamente la implementación de la función Seno, la Transformada Rápida de Fourier (FFT), y los resultados en Matlab que nos ayudaran a comprobar que se realice correctamente la implementación de hardware.

El capitulo 5 muestra las conclusiones y el posible trabajo a futuro en el cual se menciona porque es mejor implementar algoritmos en hardware a diferencia de implementarlos en software.

CAPÍTULO II

Marco Teórico

2.1 Lenguajes de Descripción de Hardware

Los lenguajes de descripción de hardware (HDLs) son utilizados para describir la arquitectura y comportamiento de un sistema electrónico los cuales fueron desarrollados para trabajar con diseños complejos.

Comparando un HDL con los lenguajes para el desarrollo de software vemos que en un lenguaje de este tipo un programa que se encuentra en un lenguaje de alto nivel (VHDL) necesita ser ensamblado a código máquina (compuertas y conexiones) para poder ser interpretado por el procesador. De igual manera, el objetivo de un HDL es describir un circuito mediante un conjunto de instrucciones de alto nivel de abstracción para que el programa de síntesis genere (ensamble) un circuito que pueda ser implementado físicamente (Torres, 2008).

Las ventajas que se tiene al utilizar un HDL:

- Con la ayuda de HDL podemos verificar con facilidad el funcionamiento de nuestro sistema dentro del procesamiento de diseño sin necesidad de implementar el circuito.
- Simularlo y tener una respuesta acerca del funcionamiento del sistema, antes de que sea implementado y poder realizar cualquier tipo de cambios si es que se requiere y de esta manera tener un resultado optimo del diseño en cuestión.
- Con este tipo de metodología, se puede reducir el tiempo de diseño a comparación del diseño antiguo de compuertas y así mismo reduce la cantidad de errores que se pueden producir en el armado de los circuitos.

2.1.1 ¿Qué es VHDL?

En la actualidad VHDL (very-high-speed integrated circuits hardware description language), es un estándar de la industria para la descripción, modelado y síntesis de circuitos digitales de gran generalidad derivado del lenguaje de alto nivel ADA. Dispone de tipos abstractos para definir formato y valores de señales, variables, constantes, etc. Y proporciona amplias facilidades para la realización de algoritmos (Torres, 2008).

VHDL es un lenguaje estructurado, jerárquico y concurrente, donde una descripción contiene elementos principales los cuales son:

- Librerías: Permiten acceder a elementos adicionales a los definidos por el lenguaje básico, almacenan distintos componentes y elementos a utilizar.

- Entidad: Es la descripción, en una caja negra de bloques funcionales, donde se indican las terminales externas del mismo. Tiene la misión de modelar la interfaz de un circuito o sistema con el exterior a través de unas entradas y salidas, según el nivel de descripción.

- Arquitectura: Contiene la vista interna, declaración de variables intermedias, componentes externos y la descripción de las tareas a realizar de nuestro circuito.

- Banco de Pruebas: Son de gran utilidad en cuanto a la simulación del diseño y pueden escribirse en el propio lenguaje y se usado para comprobar diversos modelos. Estos puede o no estar incluidos en la descripción.

Algunas de las ventajas de utilizar el lenguaje VHDL son:

- Permite el paralelismo , lo cual quiere decir que todo lo descrito se ejecuta al mismo tiempo.
- En cuanto a los Test bench (Banco de pruebas) se pueden describir en el mismo lenguaje y usarse para comprobar distintos modelos.
- Mejora la calidad y reduce el tiempo del diseño.

2.2 ¿Qué es un FPGA?

Un FPGA (Field Programmable Gate Array), son dispositivos semiconductores programables que se basan en una matriz de bloques lógicos configurables (CLB) conectados a través de interconexiones programables. A diferencia de los circuitos integrados de aplicación específica (ASIC), donde está personalizado para el diseño particular del dispositivo FPGA puede ser programado para la aplicación deseada o los requisitos de funcionalidad (XILINX, 2012). Los FPGAs son completamente reconfigurables y al instante cambian su configuración cuando se compila una diferente configuración de circuitos.

Los FPGAs llevan a cabo diferentes operaciones de manera paralela, por lo que éstas no necesitan competir por los mismos recursos. Cada tarea de procesos independientes se asigna a una sección dedicada del chip, y puede ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica.

A diferencia de generaciones anteriores de FPGAs, utilizando bloques I/O con lógica programable e interconexiones, hoy en día consisten en varias combinaciones de SRAM embebidas, Transmisores-Receptores de alta velocidad, I/O de alta velocidad. En la mayoría de las FPGAs, los bloques de lógica incluyen elementos de memoria, que pueden ser simples flipflops o bloques más completos de memoria (Corporation, 1995-2012).

En comparación con ASIC o ASSP, las FPGAs ofrecen muchas ventajas de diseño, incluyendo:

- Prototipado rápido
- Menor tiempo de comercialización
- La capacidad de re-programa en el campo de la depuración
- Reducción de los costos
- Largo ciclo de vida del producto para mitigar el riesgo de obsolescencia

Algunos de los beneficios que se obtienen al utilizar tecnología FPGA son:

- Aprovechar el paralelismo del hardware, así los FPGAs exceden la potencia de cómputo de los procesadores digitales de señales (DSP) rompiendo el paradigma de ejecución secuencial y logrando más en cada ciclo de reloj.
- Poder controlar entradas y salidas (E/S) a nivel de hardware ya que esto ofrece tiempos de respuesta más veloces y funcionalidad especializada que coincidan con los requerimientos de una aplicación.

Cada chip FPGA está compuesto de un número finito de recursos predefinidos con interconexiones programables para implementar un circuito digital reconfigurable. En la Figura 2.1 se muestra una tarjeta de desarrollo Nexys 2.

En las Figuras 2.2a, 2.2b y 2.3, se muestra cada la arquitectura de un FPGA, en la Figuras 2.2a y 2.2b se muestra donde se encuentran y como están organizadas las columnas de bloques de memorias RAM, así como los arreglos de los bloques de lógica programables y los multiplicadores.

La Figura 2.3 muestra lo donde se encuentran nuevamente los bloques de memoria, los bloques lógicos, a diferencia de la Figuras 2.2a y 2. 2b, esta muestra donde se encuentran los bloques de Entrada y Salida así como también donde se encuentras las conexiones programables de la FPGA.

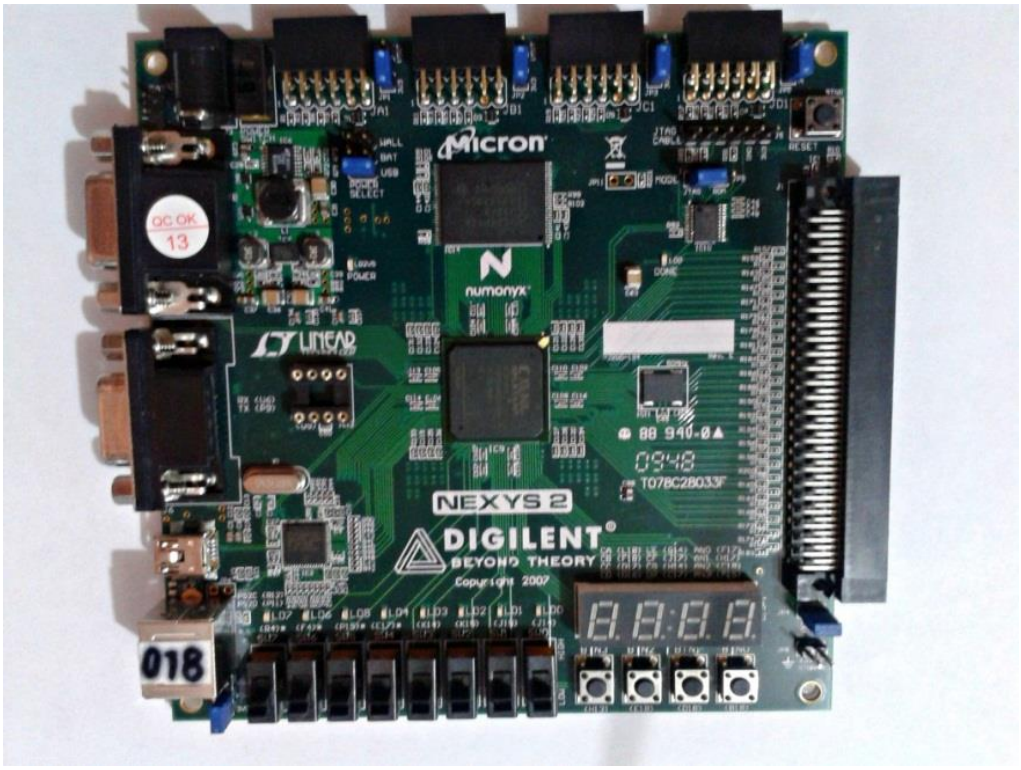

Figura 2.1 Nexys 2 FPGA.

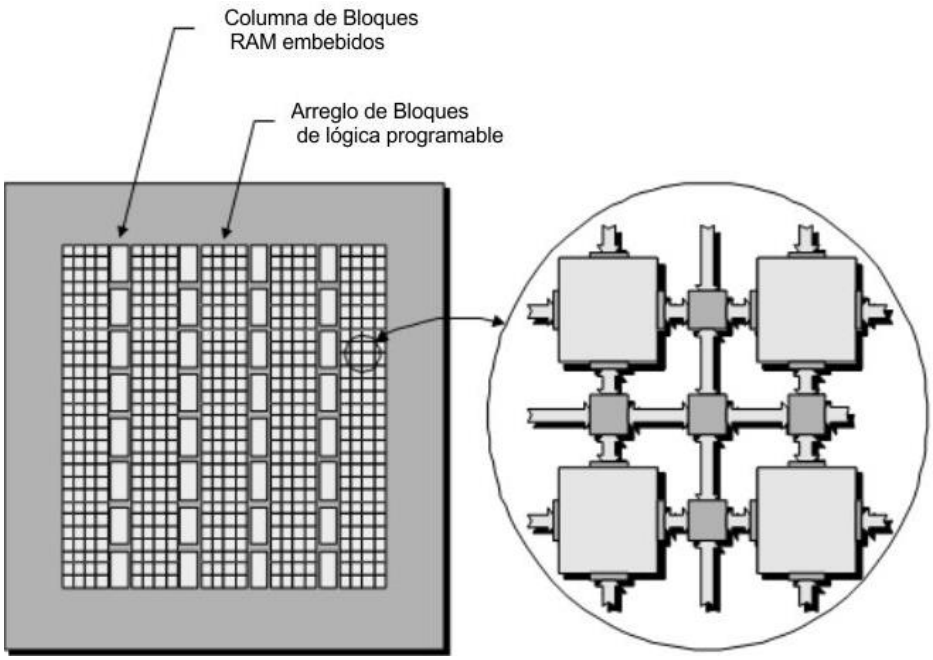

Figura 2.2a Estructura de un FPGA

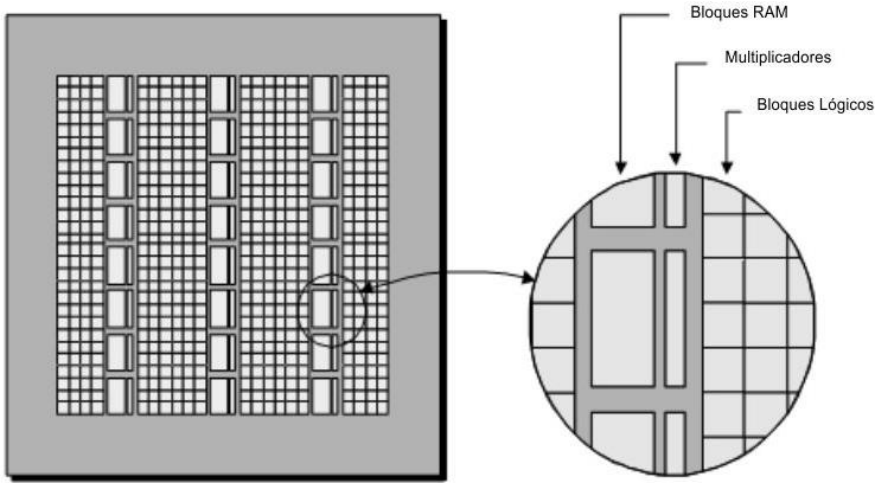

Figura 2.2b Estructura de un FPGA

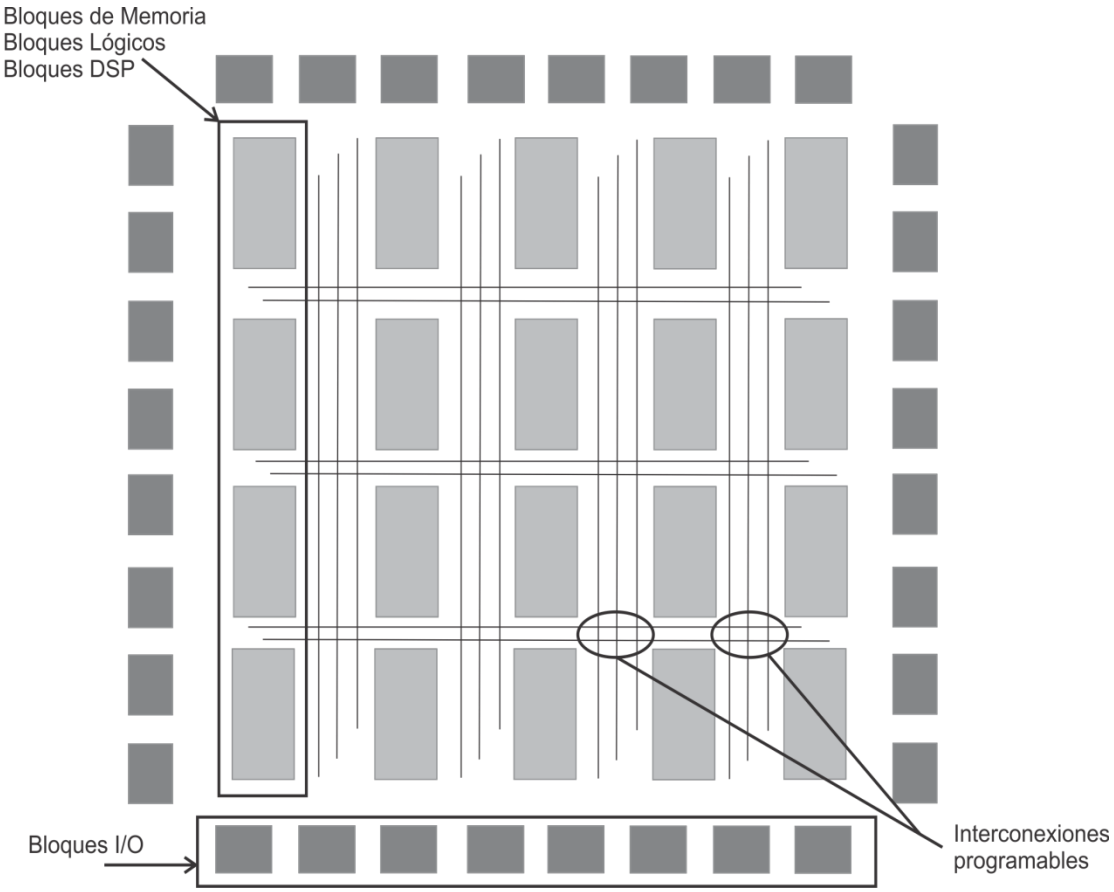

Figura 2.3 Estructura Básica de un FPGA

Los tres elementos básicos en una FPGA son el bloque configurable (CLB, Configurable Logic Block), las interconexiones y los bloques de entrada/salida (E/S). Los bloques CLB de una FPGA son menos complejos que sus homónimos en un CPLD, pero suele haber muchos más de ellos. La matriz distribuida de interconexiones programables permite interconectar los bloques CLB entre sí y conectarlos a las entradas y a las salidas. Los bloques de E/S situados alrededor del perímetro de las estructura proporcionan un acceso de entrada/salida o bidireccional, individualmente seleccionable, hacia el mundo exterior.

Cada bloque lógico de la FPGA está formado por múltiples módulos lógicos más pequeños (que son los componentes básicos) y por una serie de interconexiones programables locales que se emplean para conectar entre sí los módulos lógicos que componen el CLB, la Figura 2.4 muestra la arquitectura de un CLB.

Un módulo lógico puede configurarse para implementar lógica combinacional, lógica registrada o una combinación de ambas. Se emplea un flip-flop que forma parte de la lógica asociada para implementar lógica registrada. La Figura 2.5 muestra un diagrama de bloques de un módulo lógico típico basado en LUT (Look-Up Table, tipo de memoria programable que se utiliza para generar funciones booleanas) (Alonso, 2010).

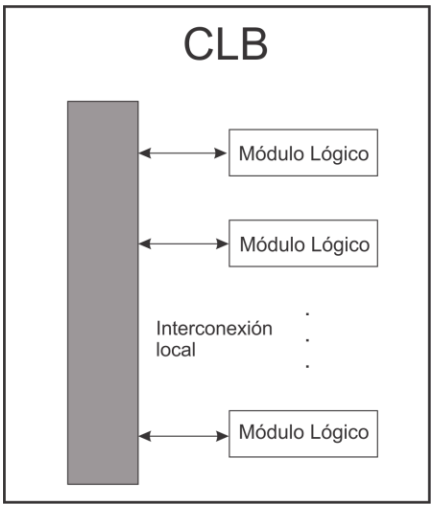

Figura 2. 4 Arquitectura de un CLB.

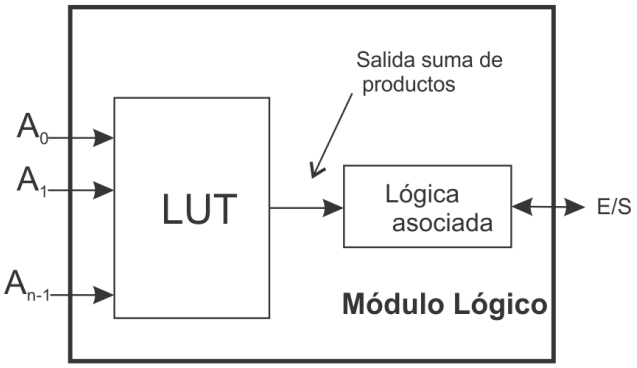

Figura 2.5 Diagrama de Bloques de un módulo lógico.

Otros de los elementos funcionales programables de las FPGA es el Bloque RAM proporciona almacenamiento de datos en forma de 18-kbit en bloques dual- port, Bloques de Multiplicadores que aceptan dos números de 18 bits binarios como entradas y permiten calcular el producto, Bloques Digital Clock Manager (DCM) que proporciona auto-calibración, soluciones totalmente digitales en la difusión, retraso, multiplicar, dividir y cambio de fase de la señal de reloj.

Los bloques de RAM tienen una estructura de doble puerto. Dos puertos idénticos llamados A y B permiten acceso independiente al mismo rango de memoria, que tiene una capacidad máxima de 18.432 bits – o 16.384 cuando no se usan las líneas de paridad. Cada puerto tiene su propio set de líneas de control, de datos y de reloj para las operaciones síncronas de lectura y escritura. Estas operaciones tienen lugar de manera totalmente independiente en cada uno de los puertos.

Hay cuatro rutas de datos básicos, como se muestra en la Figura 2. 6.

1. Escribir y leer desde el puerto A
2. Escribir y leer desde el puerto B

3. La transferencia de datos desde el puerto A al puerto B
4. La transferencia de datos desde el puerto B al puerto A

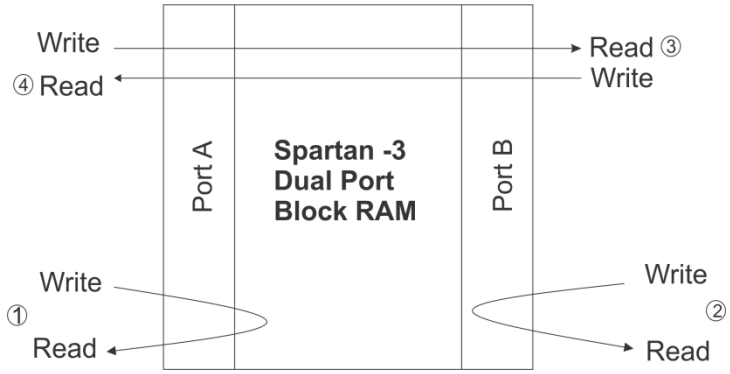

Figura 2.6 Partes de los Bloques RAM.

En cuanto a los multiplicadores cada multiplicador realiza la operación P = A × B, donde 'A' y 'B' son de 18 bits de las palabras en forma de complemento a dos, y 'P' es la completa precisión de 36-bit de producto, también en forma de complemento a dos. Las entradas de 18 bits representan valores que van desde -131,07210 a +131,07110 con un producto resultante que van desde -17,179,738,11210 a +17,179,869,18410 (XILINX, Data Sheet., 2009).

2.3 ¿Qué son los Métodos Numéricos?

Desde finales de la década de los cuarenta, la amplia disponibilidad de las computadoras digitales ha llevado a una verdadera explosión en el uso y desarrollo de los métodos numéricos. Al principio, este crecimiento estaba limitado por el costo de procesamiento de las grandes computadoras (mainframes), por lo que muchos ingenieros seguían usando simples procesamientos analíticos en una buena parte de su trabajo.

La evolución de computadoras personales de bajo costo ha permitido el acceso, de mucha gente, a las poderosas capacidades de cómputo. Por lo cual podemos mencionar la importancia del estudio de los métodos numéricos (Chapra, 2007).

1. Son herramientas muy poderosas para la solución de problemas y capaces de manipular sistemas de ecuaciones grandes, manejar no linealidades y resolver geometrías complicadas, las cuales son comunes en la práctica de la ingeniería y a menudo imposibles de resolver en forma analítica. Por tanto al estudiarlos aumenta la habilidad de resolver problemas.
2. Debido a que la mayoría de los métodos numéricos están diseñados para usarlo en las computadoras, son especialmente adecuados para ilustrar el poder y las limitaciones de las computadoras.

Por tanto ahora en las Ciencias y la Ingeniería son indispensables en la solución de problemas considerados como una herramienta matemática.

Los métodos numéricos son procedimientos para calcular las incógnitas o variables que resultan de aplicar las teorías de la ingeniería y las matemáticas a casos prácticos y que en ocasiones no pueden obtenerse por métodos basados en técnicas analíticas exclusivamente. Con ellos podemos formular problemas matemáticos de tal forma que puedan resolverse usando algunas operaciones aritméticas, ya que con estos tratamos de diseñar métodos para “aproximar” de manera eficiente las soluciones de problemas expresados matemáticamente, utilizando sólo operaciones más simples de la aritmética. Estos pueden ser aplicador, por ejemplo, en:

- Cálculo de derivadas
- Integrales
- Ecuaciones diferenciales
- Operaciones con matrices

- Interpolaciones
- Ajuste de curvas
- Polinomios

Ya que los métodos numéricos son un conjunto de técnicas que se utilizan para la solución de problemas de ingeniería, esto gracias a que tan solo su usa un número finito de operaciones aritméticas, con las cuales se encargan de diseñar diversos algoritmos para obtener las soluciones más aproximadas a los problemas del mundo real. Por tanto tenemos algunas de las áreas donde se pueden aplicar los métodos numéricos las cuales son en: Ingeniería Industrial, Ingeniería Química, Ingeniería Civil, Ingeniería Mecánica, Ingeniería Computacional, Ingeniería Eléctrica, entre otras (Guerra, 2006).

En el proceso de solución de problemas por medio de computadoras y con la ayuda de los métodos numéricos se requieren los pasos siguientes:

- Especificación del problema: Con esto se indica que se debe identificar perfectamente el problema y sus limitaciones, las variables que intervienen y los resultados deseados.
- Análisis: Es la formulación de la solución del problema denominada también algoritmo, de manera que se tenga una serie de pasos que resuelvan el problema y que sean susceptibles de ejecutarse en la computadora.
- Programación: Este paso consiste en traducir el método de análisis o algoritmo de solución expresándole como una serie detallada de operaciones.
- Verificación: Es la prueba exhaustiva del programa para eliminar todos los errores que tenga de manera que efectúe lo que desea los resultados de prueba se comparan con soluciones conocidas de problemas ya resueltos.

2.4 Series de Taylor

La serie de Taylor es, sin duda, el fundamento matemático más importante para comprender, manejar y formular métodos numéricos que se basan en la aproximación de funciones por medio de polinomios. Aunque a veces no sea muy evidente, la mayoría de los métodos numéricos se basan en la aproximación de funciones por medio de polinomios (Rocha, 2005).

Proporciona una buena forma de aproximar el valor de una función en un punto en términos del valor de la función y sus derivadas en otro punto. Las series de Taylor se basan en ir haciendo operaciones según una ecuación general y mientras más operaciones tenga la serie, más exacto será el resultado que se esta buscando, como se muestra en la Ecuación (1).

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a)^1 + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^n(a)}{n!}(x-a)^n \qquad (1)$$

La Ecuación (1) también se expresa de una forma más compacta como se presenta en la Ecuación (1.1).

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \qquad (1.1)$$

Donde n! es el factorial de n y $f^{(n)}(a)$ denota la n-ésima derivada de la f en el punto a; como se puede observaren la ecuación, hay una parte en la cual hay que desarrollar un binomio$(x-a)^n$ por lo que para simplificar se igualara a “a” siempre a 0. La derivada cero de f es definida como la propia f y $(x-a)^0$ y 0! Son ambos definidos como uno.

Algunas de las funciones en la Serie de Taylor que con más frecuencia se utilizan (Bastidas, 2002) se muestran en las Ecuaciones (2) a (8).

a) $\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + \cdots + x^n + \cdots$, si $-1 < x < 1$. (2)

b) $\log(1+x) = \sum_{n=0}^{\infty} \frac{(-1)^{n-1}}{n} x^n = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \cdots$, si $-1 < x \leq 1$. (3)

c) $e^x = \sum_{n=0}^{\infty} \frac{1}{n!} x^n = 1 + x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \cdots$, para todo $x \in \mathbb{R}$. (4)

d) $\text{sen } x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \cdots$, para todo $x \in \mathbb{R}$. (5)

e) $\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \cdots$, para todo $x \in \mathbb{R}$. (6)

f) $\text{arc sen } x = \sum_{n=0}^{\infty} \frac{1*3*5\,(2n-1)}{2*4*6(2n)} * \frac{x^{2n+1}}{2n+1} = x + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \cdots$, si $-1 \leq x \leq 1$. (7)

g) $\text{arc tg } x = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1} = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 + \frac{1}{7}x^7 + \cdots$, si $-1 \leq x \leq 1$. (8)

2.4.1 Aproximación Polinomial

El principal objetivo de las aproximaciones polinomiales es obtener una función polinomial que represente la distribución gráfica de una serie de puntos en un sistema coordenado. Una vez obtenida la función de aproximación, puede emplearse para realizar los cálculos cuando la diferencia entre el valor real de la función y la aproximación polinomial es suficientemente pequeña.

Varios métodos pueden emplearse para aproximar una función dada, mediante polinomios, uno de los más ampliamente utilizados hace uso de las fórmulas de Taylor. Las aproximaciones polinomiales, secesiones y series infinitas, forman parte importante dentro del cálculo diferencial e integral.

El polinomio de la Ecuación (9) permite predecir el valor de la función en un punto cualquiera x, en términos de la propia función y de sus derivadas en el punto x_i.

$$P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n + \cdots \quad (9)$$

El polinomio P(x) de la ecuación (10) se hace coincidir con la función f(x) ¸ y las primeras n derivadas del polinomio, ecuaciones (10.1) y (10.2), se hacen coincidir con las n primeras derivadas de la función en el punto x_i (Rocha, 2005).

$$P(x_i) = f(x_i) \tag{10}$$

$$P'(x_i) = f'(x_i) \tag{10.1}$$

$$P^{(n)} = f^{(n)}(x_i) \tag{10.2}$$

2.4.2 Teorema de Taylor

Recibe su nombre del matemático británico Brook Taylor, quién lo enuncio con mayor generalidad en 1712, aunque previamente James Gregory lo había descubierto en 1671. Este teorema permite obtener aproximaciones polinómicas de una función en un entorno de cierto punto en que la función es diferenciable (Rocha, 2005).

Sea $f(z)$ una función analítica en todo punto del disco C_0, con centro en z_0 y radio r_0. Entonces, en cada punto de z del disco C_0, $f(z)$ se expresa en la ecuación (11).

$$f(z) = \sum_{n=0}^{\infty} \frac{f^{(n)}(z_0)}{n!}(z - z_0)^n \tag{11}$$

Es decir, la serie de potencia de la ecuación (11) converge a $f(z)$ cuando $|z - z_0| < r_0$.

La serie se denomina desarrollo en serie de Taylor o, simplemente, desarrollo de Taylor de $f(z)$ alrededor del punto z_0. Si $z_0 = 0$, el desarrollo de Taylor adquiere la forma de la ecuación (11.1).

$$f(z) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} z^n \qquad (11.1)$$

Y se denomina desarrollo de Maclaurin de $f(z)$.

2.4.3 Ejemplo de la Serie de Taylor.

A continuación realizaremos un ejemplo de las series de Taylor aplicando la ecuación (5):

Se desea encontrar una serie de Taylor para la función $f(x) = sen(35°)$, donde primero tenemos que conocer a *f* y sus derivadas en el punto x_0 el cual se muestra en la ecuacuión (12).

$$f(x) = sen\,(x) \qquad (12)$$
$$x_0 = \frac{\pi}{6}$$

Evaluando las derivadas de la ecuación (12), tenemos las ecuaciones (12.1) a la (12.4).

$$f(x) = sen(x) \quad => \quad f\left(\frac{\pi}{6}\right) = 0.5 \qquad (12.1)$$

$$f'(x) = cos(x) \quad => \quad f'\left(\frac{\pi}{6}\right) = 0.86602 \qquad (12.2)$$

$$f''(x) = -sen(x) \quad => \quad f''\left(\frac{\pi}{6}\right) = 0.5 \qquad (12.3)$$

$$f'''(x) = -cos(x) \quad => \quad f'''\left(\frac{\pi}{6}\right) = -0.86602 \qquad (12.4)$$

Una vez obtenidas cada una de las derivadas, se aplica la Ecuación (1) a la Ecuación (13).

$$f(x) = f\left(\frac{\pi}{6}\right) + \frac{f'\left(\frac{\pi}{6}\right)}{1!}[x - \left(\frac{\pi}{6}\right)]^1 + \frac{f''\left(\frac{\pi}{6}\right)}{2!}[x - \left(\frac{\pi}{6}\right)]^2 + \frac{f'''\left(\frac{\pi}{6}\right)}{3!}[x - \left(\frac{\pi}{6}\right)]^3 \quad (13)$$

Sustituyendo en la Ecuación (13) tenemos la Ecuación (13.1).

$$sen(x) = 0.5 + 0.86602\left[x - \left(\frac{\pi}{6}\right)\right] - 0.25\left[x - \left(\frac{\pi}{6}\right)\right]^2 - 0.1443366\left[x - \left(\frac{\pi}{6}\right)\right]^3 \quad (13.1)$$

De esta manera se observo que esta expresión sirve para estimar valores de *sen(x)* para *x* cercanos a $\frac{\pi}{6}$, sustituyendo en la Ecuación (13.1) y evaluando la Ecuación (13.2), resulta la Ecuación (13.3).

$$sen(35°) = 0.5 + 0.86602\left[\frac{5\pi}{180}\right] - 0.25\left[\frac{5\pi}{180}\right]^2 - 0.1443366\left[\frac{5\pi}{180}\right]^3 \quad (13.2)$$

$$sen(35°) = 0.5 + 0.0755749 - 0.001903858 - 0.000095922 = \mathbf{0.57357512} \quad (13.3)$$

Para comprobar el resultado de $sen\,(35°)$ se realizó la operación en calculadora donde se obtuvo el resultado de $\mathbf{0.5735764}$, que si comparamos con el resultado obtenido en la Ecuación (13.3) del ejercicio anteriormente realizado tenemos una gran similitud, con una diferencia minima en la parte decimal.

2.5 Series de Fourier y Transformada de Fourier

Una función en el dominio temporal indica cómo la amplitud de la señal cambia en el tiempo, su representación en el dominio de la frecuencia permite conocer cuan a menudo la frecuencia se puede visualizar considerando que la señal en estudio está compuesta por la suma de ondas sinusoidales simples de amplitud y fase adecuadas o de exponenciales complejas relacionadas armónicamente.

La herramienta matemática que permite el pase del dominio tiempo al dominio de la frecuencia es la Serie de Fourier para las señales periódicas, y de la Transformada de Fourier para las señales de energía finita (Cordoba, 2008).

La Trasformada de Fourier nos permite analizar las señales desde un punto de vista distinto, facilitando la compresión de muchos procedimientos utilizados en telecomunicaciones, tales como la modulación y la representación digital de las señales (Bonafonte, 2009).

2.5.1 Series de Fourier

Podemos describir cualquier señal periódica compleja en forma de una suma de muchos fasores (senos o cosenos). Un método para describir una señal de esta forma, son las series de Fourier.Las series de Fourier nos permiten expresar algunas funciones periódicas *f(t)* de periodo T como se muestra en la Ecuación (14).

$$f\,(\mathrm{t}) \;=\; ½\,\mathrm{a}_0 + \;\mathrm{a}_1\cos(\omega_0\mathrm{t}) + \mathrm{a}_2\cos(2\;\omega_0\mathrm{t}) + \ldots + \mathrm{b}_1\mathrm{sen}(\omega_0\mathrm{t}) + \mathrm{b}_2\mathrm{sen}(2\;\omega_0\mathrm{t}) \qquad (14)$$

Donde $\omega_0 = 2\pi/T$ (15)

Es decir:

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}[a_n\cos(n\omega_0 t) + b_n\,sen(n\omega_0 t)] \qquad (16)$$

- Componentes y armónicos

Una función periódica f(t) se puede escribir como la suma de componentes sinusoidales de diferentes frecuencias $\omega_n = n\omega_0$, a la componente

sinusoidal de la frecuencia $n\omega_0$: $C_n\cos(n\omega_0 t + \theta_n)$, se le llama enésima armónica de f(t).

A la primera armónica (n=1) se le llama la Componente fundamental y su periodo es el mismo que el de f(t) y a la frecuencia $\omega_0 = 2\pi f_0 = 2\pi/T$ se le llama frecuencia angular fundamental. Los coeficientes C_n y los ángulos θ_n son respectivamente las amplitudes y los ángulos de fase de los armónicos.

En la Figura 2.7 se muestra un ejemplo, donde, La función $f(t) = \cos\left(\frac{t}{3}\right) + \cos\left(\frac{t}{4}\right)$, tiene un periodo T= 24π, por lo tanto su frecuencia fundamental es $\omega_0 = \frac{1}{12}\,rad/seg$, la componente fundamental es de la siguiente forma: $0 * \cos\left(\frac{t}{12}\right)$. Como ejemplo tenemos su tercer armónico que es, $\cos\left(\frac{3t}{12}\right) = \cos\left(\frac{t}{4}\right)$ y su cuarto armónico será $\cos\left(\frac{4t}{12}\right) = \cos\left(\frac{t}{3}\right)$.

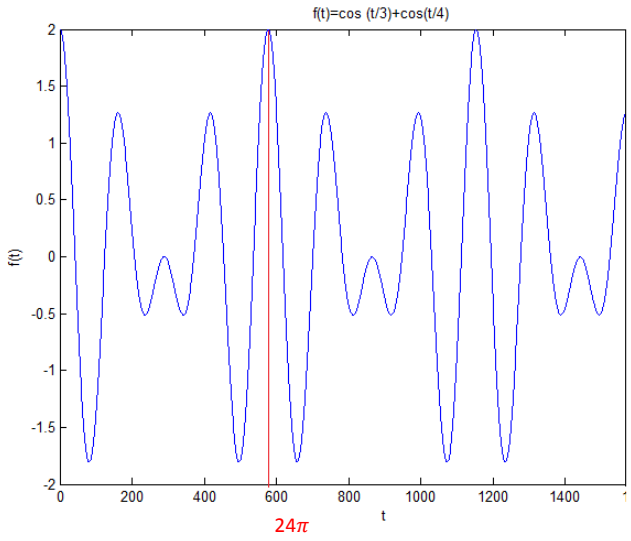

Figura 2. 7 Grafica dela función $f(t) = \cos\left(\frac{t}{3}\right) + \cos\left(\frac{t}{4}\right)$

- Cálculos de los coeficientes de la Serie de Fourier

Para poder obtener la serie de Fourier de una función necesitamos saber cuales son los valores de cada uno de los coeficientes a0, a1, a2,…, b0, b1, b2. Los coeficientes se obtienen multiplicando ambos miembros por $\cos(\mathrm{n}\omega_0\,\mathrm{t})$ e integrando de –T/2 a T/2 y con la Ecuación (17) obtenemos los coeficientes a_n.

$$a_{1,2,3,\ldots} = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos(n\omega_0 t)dt \qquad n = 0,1,2,3\ldots \tag{17}$$

Similarmente, con la Ecuación (18) obtenemos b_n:

$$b_{1,2,3,\ldots} = \frac{2}{T}\int_{-T/2}^{T/2} f(t)sen(n\omega_0 t)dt \qquad n = 1,2,3,\ldots \tag{18}$$

Y con la Ecución (19) se encuentra el coeficiente a_0 , integrando de –T/2 a T/2.

$$a_0 = \frac{2}{T}\int_{-T/2}^{T/2} f(t)dt \tag{19}$$

Para encontrar los coeficientes $\mathrm{a_n}$ utilizamos la Ecuación (20).

$$\mathrm{a_n} = \frac{2}{\mathrm{T}}\int_{-\mathrm{T}/2}^{\mathrm{T}/2} \mathrm{f(t)}\cos(\mathrm{n}\omega_0\mathrm{t})\mathrm{dt} \tag{20}$$

$$\mathrm{a_n} = \frac{2}{\mathrm{T}}\left[\int_{-\mathrm{T}/2}^{0} -\cos(\mathrm{n}\omega_0\mathrm{t})\,\mathrm{dt} + \int_{0}^{\mathrm{T}/2}\cos(\mathrm{n}\omega_0\mathrm{t})\,\mathrm{dt}\right] \tag{20.1}$$

Resolviendo la Ecuación (20.1), da como resultado la Ecuación (21).

$$\mathrm{a_n} = \frac{2}{\mathrm{T}}\left[\frac{-1}{\mathrm{n}\omega_0}\,\mathrm{sen}(\mathrm{n}\omega_0\mathrm{t})\Big|_{-\mathrm{T}/2}^{0} + \frac{1}{\mathrm{n}\omega_0}\,\mathrm{sen}(\mathrm{n}\omega_0\mathrm{t})\Big|_{0}^{\mathrm{T}/2}\right] = 0 \qquad \text{para } \mathrm{n} \neq 0 \tag{21}$$

Evaluando la Ecuación (21), con los valores ω_0 y T dados a continuación:

$$\omega_0 = \frac{1}{12}\frac{\text{rad}}{\text{seg}} \qquad\qquad \text{T} = 24\pi$$

Y sustituyendolos se obtiene la Ecuación (22).

$$\begin{aligned} a_n = \frac{2}{T}\left[\left(\frac{-1}{\frac{1}{12}}\ sen\left[\frac{1}{12}(0)\right]\right)\right] - \left[\frac{-1}{\frac{1}{12}}\ sen\ \left(\frac{1}{12}\left[\frac{-24\pi}{2}\right]\right)\right] \\ + \left[\frac{-1}{\frac{1}{12}}\ sen\left(\frac{1}{12}\left[\frac{24\pi}{2}\right]\right)\right] - \left[\frac{-1}{\frac{1}{12}}\ sen\ \left(\frac{1}{12}[0]\right)\right] \end{aligned} \tag{22}$$

Resolviendo la ecuación (22) da como resultado la ecuación (22.1).

$$a_n = \frac{2}{T}\Big(-12\ \text{sen}(0) - \big(-12\ \text{sen}\ (-\pi)\big) + 12\ \text{sen}(\pi) - 12\ \text{sen}\ (0)\Big) \tag{22.1}$$

Sustituyendo T=24π en la Ecuación (22.1), obtenemos la ecuación (23).

$$a_n = \frac{2}{24\pi}[0+0] = \frac{2}{24\pi}[0] = \frac{1}{12}\pi\ (0) = 0 \tag{23}$$

Con esto se puede comprobar que el resultado de la formula será cero porque *n* no es igual a cero. Lo que compueba lo antes descrito en la Ecuación (21).

Para encontrar los coeficientes a_0 tomamos la ecuación (24).

$$a_0 = \frac{2}{T}\int_{-T/2}^{T/2} f(t)dt \ = \frac{2}{T}\left[\int_{-T/2}^{0} -\text{dt} + \int_{0}^{T/2} \text{dt}\right] = \frac{2}{T}\left[-\text{t}\,|_{-\frac{T}{2}}^{0} + \text{t}\,|_{0}^{\frac{T}{2}}\right] = 0 \tag{24}$$

Resolviendo la Ecuación (24) con T = 24π, se obtiene cero el cual se muestra en la Ecuación (25).

$$a_0 = \left[-(0) - \left(-\frac{24\pi}{2}\right)\right] + \left[-(0) - \left(\frac{24\pi}{2}\right)\right] = 12\pi + (-12\pi) = 0 \qquad (25)$$

De igual manera podemos comprobar, que el resultado de $a_0$será igual a cero.

Finalmente para encontrar los coeficientes b_n tenemos la Ecuación (26), la cual nos ayuda a encontrar la ecuación definitiva para encontrar los cieficientes que se muestra en la Ecuación (28).

$$b_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\,\mathrm{sen}(n\omega_0 t)dt = \frac{2}{T}\left[\int_{-T/2}^{0} -\mathrm{sen}(n\omega_0 t)\,dt + \int_{0}^{T/2} \mathrm{sen}(n\omega_0 t)\,dt\right] \qquad (26)$$

Resolviendo las integrales de la Ecuación (26), obtenemos las Ecuaciones (27) y (27.1).

$$b_n = \frac{2}{T}\left[\frac{1}{n\omega_0}\cos(n\omega_0 t)\Big|_{-\frac{T}{2}}^{0} + \frac{-1}{n\omega_0}\cos(n\omega_0 t)\Big|_{0}^{\frac{T}{2}}\right] \qquad (27)$$

$$b_n = \frac{1}{n\pi}[(1-\cos(n\pi)) - (\cos(n\pi) - 1)] \qquad (27.1)$$

Por lo tanto la Ecuación (28) es la definitiva para encontrar los coeficientes b_n.

$$b_n = \frac{2}{n\pi}[1-(-1)^n] \quad \text{para } n \neq 0 \qquad (28)$$

Si resolvemos la Ecuación (27) para comprobar, dándole un valor a ω_0.

$$\omega_0 = \frac{2\pi}{T}$$

Entonces tendremos las Ecuaciones (29) a la (29.5).

$$b_n = \frac{2}{T}\left(\left[\left(\frac{1}{\frac{2\pi n}{T}}\ \cos\left[n\ \frac{2\pi}{T}\left(-\frac{T}{2}\right)\right]\right)\right]_{-\frac{T}{2}}^{0} - \left[\frac{-1}{\frac{2\pi n}{T}}\ \cos\left(n\left[\frac{2\pi}{T}\right]t\right)\right]_{0}^{\frac{T}{2}}\right) \quad (29)$$

$$b_n = \frac{2}{T}\left[\left(\frac{1}{\frac{2\pi n}{T}} - \frac{1}{\frac{2\pi n}{T}}\cos\left[n\ \frac{2\pi}{T}\left(-\frac{T}{2}\right)\right]\right) - \left(\frac{1}{\frac{2\pi n}{T}}\ \cos n\left[\left(\frac{2\pi}{T}\right)\left(\frac{T}{2}\right)\right] - \frac{1}{\frac{2\pi n}{T}}\right)\right] \quad (29.1)$$

$$b_n = \frac{2}{T}\left[\left(\frac{1}{\frac{2\pi n}{T}} - \frac{1}{\frac{2\pi n}{T}}\cos(n\,\pi)\right) - \left(\frac{1}{\frac{2\pi n}{T}}\ \cos(n\,\pi) - \frac{1}{\frac{2\pi n}{T}}\right)\right] \quad (29.2)$$

$$b_n = \left(\frac{1}{n\pi} - \frac{1}{n\pi}\cos(n\pi)\right) - \left(\frac{1}{n\pi}\cos(n\pi) - \frac{1}{n\pi}\right) \quad (29.3)$$

$$b_n = \frac{1}{n\pi}[(1-\cos(n\pi)) - (\cos(n\pi) - 1)] = \frac{1}{n\pi}[1 - (-1) - \cos(n\pi) - \cos(n\pi)] \quad (29.4)$$

$$b_n = \frac{1}{n\pi}[2 - 2\cos(n\pi)] = \frac{2}{n\pi}[1 - \cos(n\pi)] = \frac{2}{n\pi}[1 - (-1)^n] \quad \text{para } n \neq 0 \quad (29.5)$$

De esta manera podemos comprobar en la Ecuación (29.5) que para toda *n* no igual a cero nuestro resultado para la evaluación de los coeficientes b_n nos quedara como se muestra en la Ecuación (28) (Hidalgo, 2003).

2.5.2 Ejemplo de las Series de Fourier

Teniendo los armónicos 3, 5, 7 y $\omega_0 = \pi$ y T = 2, se desea encontrar el valor de cada uno de sus coeficientes tanto a_n como $b_n$de la serie de Fourier.

Primero se evalua el coeficiente $a_0$con la Ecuación (24), sustituyendo los valores anteriores yse obtiene la Ecuación (30):

$$a_0 = 1[-(-1) + (1)] = \mathbf{2} \quad (30)$$

Ahora evaluamos a_1 con la Ecuación (21) y obtenemos las Ecuaciones (31) y (31.1).

$$a_1 = \frac{2}{T}\left[\begin{array}{c}\left(-\frac{1}{\pi} sen[1(\pi)(0)] - \frac{-1}{\pi} sen\left[1(\pi)\left(\frac{-T}{2}\right)\right]\right) + \\ \left(\frac{1}{\pi} sen\left[n\pi\left(\frac{T}{2}\right)\right] - \frac{1}{n\pi} sen\left[n(\pi)(0)\right]\right)\end{array}\right] \quad (31)$$

$$a_1 = \frac{2}{T}\left[\frac{1}{\pi} sen\left(\pi\left[\frac{-2}{2}\right]\right) + \frac{1}{\pi} sen\left(\pi\left[\frac{2}{2}\right]\right)\right] = \frac{2}{T}[0+0] = \frac{2}{2}[0] = \mathbf{0} \quad (31.1)$$

Ahora empleamos la Ecuación (28) para la los coeficientes b_n, para este ejemplo se evalua hasta b_7, los resultados se muestran en las Ecuaciones (32) a la (38).

$$b_1 = \frac{2}{n\pi}[1-(-1)^1] = \frac{2}{\pi}(2) = \mathbf{\frac{4}{\pi}} \quad (32)$$

$$b_2 = \frac{2}{2\pi}[1-(-1)^2] = \pi(0) = \mathbf{0} \quad (33)$$

$$b_3 = \frac{2}{3\pi}[1-(-1)^3] = \frac{2}{3\pi}(2) = \mathbf{\frac{4}{3\pi}} \quad (34)$$

$$b_4 = \frac{2}{3\pi}[1-(-1)^4] = \frac{2}{4\pi}(2) = \mathbf{0} \quad (35)$$

$$b_5 = \frac{2}{3\pi}[1-(-1)^5] = \frac{2}{5\pi}(2) = \mathbf{\frac{4}{5\pi}} \quad (36)$$

$$b_6 = \frac{2}{6\pi}[1-(-1)^6] = \frac{2}{6\pi}(0) = \mathbf{0} \quad (37)$$

$$b_7 = \frac{2}{7\pi}[1-(-1)^7] = \frac{2}{7\pi}(2) = \mathbf{\frac{4}{7\pi}} \quad (38)$$

Por tanto tenemos como resultados :

$$b_n = \frac{4}{\pi}, \frac{4}{3\pi}, \frac{4}{5\pi}, \frac{4}{7\pi}$$

Entonces una vez teniendo los valores de cada uno de los coeficientes tendríamos la serie de Fourier como se muestra en la Ecuación (39).

$$f(t) = \frac{4}{\pi}\left[\text{sen}\,(\omega_0 t) + \frac{1}{3}\,\text{sen}(3\,\omega_0 t) + \frac{1}{5}\,\text{sen}(5\,\omega_0 t) + \frac{1}{7}\text{sen}(7\,\omega_0 t) + \cdots\right] \quad (39)$$

La Ecuación (39) resulta de la evaluación de los coeficientes de b_n ya que los coeficientes pares dan cero, se evalúan los b_n impares por lo que tenemos los coeficientes $b_{1,3,5,7,9\ldots}$, en la Figura 2.8 se muestran los componente de la Serie de Fourier con respecto al ejemplo resuelto anteriormente, donde se muestra con la línea negra la suma de los componentes.

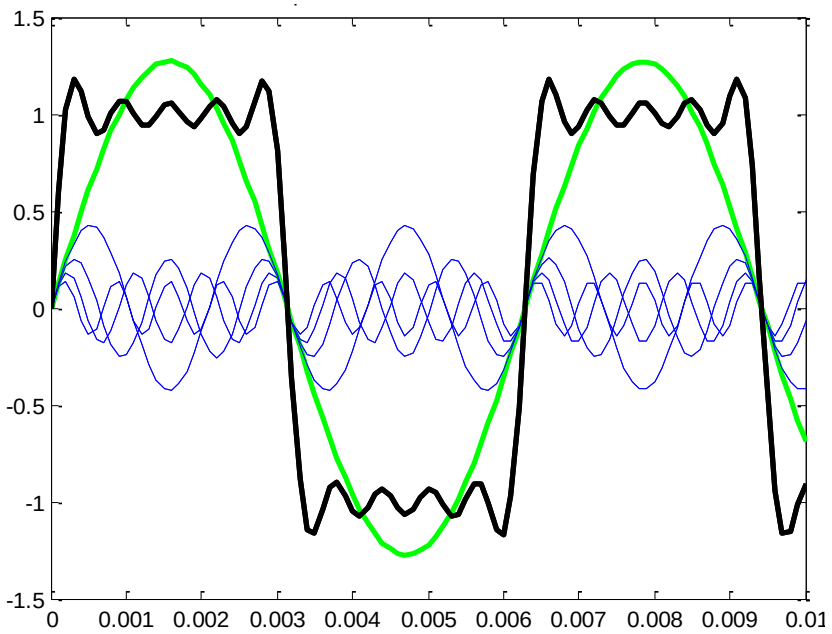

Figura 2.8 Componentes de la Serie de Fourier, 1, 3, 5, 7.

2.5.3 Transformada de Fourier

En aplicaciones reales la mayor parte de las señales no son periódicas, por lo cual necesitamos transformar nuestras series de Fourier para poder analizarlas, para lo cual nos sirve la Transformada de Fourier ya que es básicamente el espectro de frecuencias de una función ya sea un fenómeno ondulatorio, luminoso o electromagnético, con está podemos medir la distribución de amplitudes de cada frecuencia.

Por ejemplo lo que hace el oído humano, este recibe una onda auditiva y la transforma en una descomposición en distintas frecuencias que es lo que finalmente escuchamos. El oído humano va percibiendo distintas frecuencias a medida que pasa el tiempo, sin embargo, la transformada de Fourier contiene todas las frecuencias contenidas en todos los tiempos en que la señal existió; es decir, que en la transformada de Fourier obtiene un solo aspectro de frecuencias para toda la función.

Lo que nos permite la Transformada de Fourier es hacer una descomposición espectral de las formantes de una onda o señal oscilatoria, pero también nos permite que con el espectro generado con nuestro análisis que realizamos, podemos reconstruir la función original mediante la transformada inversa. La Transformada de Fourier contiene todas las frecuencias contenidas en todos los tiempos en que existió la señal, este se refiere a que se obtiene un solo espectro de frecuencia para toda la función (Cáceres, 2007).

Sea x(t) una señal continua. Se define la transformada de Fourier de x, denotada en la Ecuación (40).

$$\mathrm{X(f)} = \int_{-\infty}^{\infty} \mathrm{x(t)e^{-j2\pi ft}\,dt} \qquad (40)$$

Donde:

t : Tiempo
f : Frecuencia en Hz
x (t) : Señal de prueba
$e^{-j2\pi ft}$: Fasor de Sondeo (Kernel Funtion)
X(f) : Espectro de función de la frecuencia f.

2.5.4 Transformada Discreta de Fourier (DFT)

La Transformada Discreta de Fourier es un método muy eficiente para determinar el espectro en frecuencia de una señal. Esta transformada juega un papel muy importante en numerosas aplicaciones de procesamiento de señales digitales. Es una secuencia de x[n] como se muestra en la Ecuación (41):

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk} \qquad \text{para} \qquad k = 0,1,\dots,N-1 \tag{41}$$

Donde el valor de W_n esta dado por:

$$W_n = e^{-\frac{j2\pi}{N}} \tag{42}$$

El W de la Ecuación (42) son constantes conocidas como factores twiddle.

Para realizar una aproximación al cálculo de la Transformada Discreta de Fourier (DFT) esta requerirá la suma compleja de N multiplicaciones complejas, en total N^2 multiplicaciones complejas y N^2 sumas complejas para realizar un DFT de N puntos.

Una de las razones de gran relevancia de la Transformada Discreta es la existencia de algoritmos sumamente eficientes para calcularla. Entre ellos se encuentra el Radix 2, el cual permite hallar de forma rápida la Transformada de Fourier, razón por la que recibe el nombre de FFT (Fast Fourier Transform) (Posadas, 1998).

2.5.5 Transformada Rápida de Fourier (FFT)

Lo que consigue el algoritmo FFT es simplificar enormemente el cálculo del DFT introduciendo “atajos” matemáticos para reducir drásticamente el número de operaciones (5to curso, 1999).

La FFT lo que hace es que por la periodicidad y simetría de los factores twiddle “W” para lo que es el cálculo de la Transformada Discreta de Fourier. La FFT en primera instancia descompone la DFT de N/2 puntos y a su vez cada DFT se vuelve a descomponer en una DFT de N/4 puntos y así sucesivamente. Ya que se termina la descomposición se obtienen (N/2) DFT’s de 2 puntos cada una. Por ejemplo para una FFT de base 2, N debe ser una potencia de 2 y la transformada más pequeña es la DFT de 2 puntos. Para implementar la FFT existen dos procedimientos: diezmado en frecuencia y el diezmado en el tiempo (Cordoba, 2008).

- **Algoritmo para la FFT**

Para poder implementar la FFT primero se tiene que tener en cuenta cual de los procedimientos de diezmado se utilizara ya sea el de frecuencia o el de tiempo. En este caso se utilizó el diezmado en tiempo.

El diezmado en tiempo divide la secuencia de datos de entrada x[n] en dos grupos, uno de índices par y otro en índices impar, con estas sub-secuencias se realiza el DFT de N/2 puntos y sus resultados se combinan para formar el DFT de N puntos (Gómez, 2000). En las Ecuaciones (43) a la (43.2) se muestra el proceso de la DFT que llega a determinar la formula del algoritmo la cual se muestra en la Ecuación (44).

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{(2n+1)k} \quad (43)$$

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{2nk} \quad (43.1)$$

Sustituyendo a $x_1[n] = x[2n]$, $x_2[n] = x[2n+1]$ y $W_N^{2nk} = W_{N/2}^{nk}$ se obtiene la Ecuación (43.2).

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{N/2}^{nk} \quad (43.2)$$

Simplificando se obtiene la Ecuación (44).

$$X[k] = Y[k] + W_N^k Z[k] \qquad k = 0,1,2,\dots,N-1 \quad (44)$$

En la ecuación (44) se muestra que la DFT de N puntos es la suma de dos DFT's de N/2 puntos los cuales son $Y[k]\, y\, Z[k]$ que se realizan con las secuencias originales que se tienen de x[n]. Y cada uno de los términos de $Z[k]$ lo tenemos que multiplicar por un factor twiddle.

Donde el factor twiddle o coeficiente trigonométrico lo obtenemos con la ecuación (45):

$$W_N^n = \cos\left(\frac{2\pi n}{N}\right) - j\, sen\left(\frac{2\pi n}{N}\right) = e^{-\frac{j2\pi n}{N}} \quad (45)$$

Y el coseno y seno son:

$cos\left(\frac{2\pi n}{N}\right)$: Es la parte real del coeficiente.
$-\, sen\left(\frac{2\pi n}{N}\right)$: Es la parte imaginaria del coeficiente.

Entonces N es el número de datos de entrada y *n* toma valores desde 0 hasta N-1.Por ejemplo el diagrama para un diezmado en tiempo de N=8 puntos se muestra la Figura 2.9:

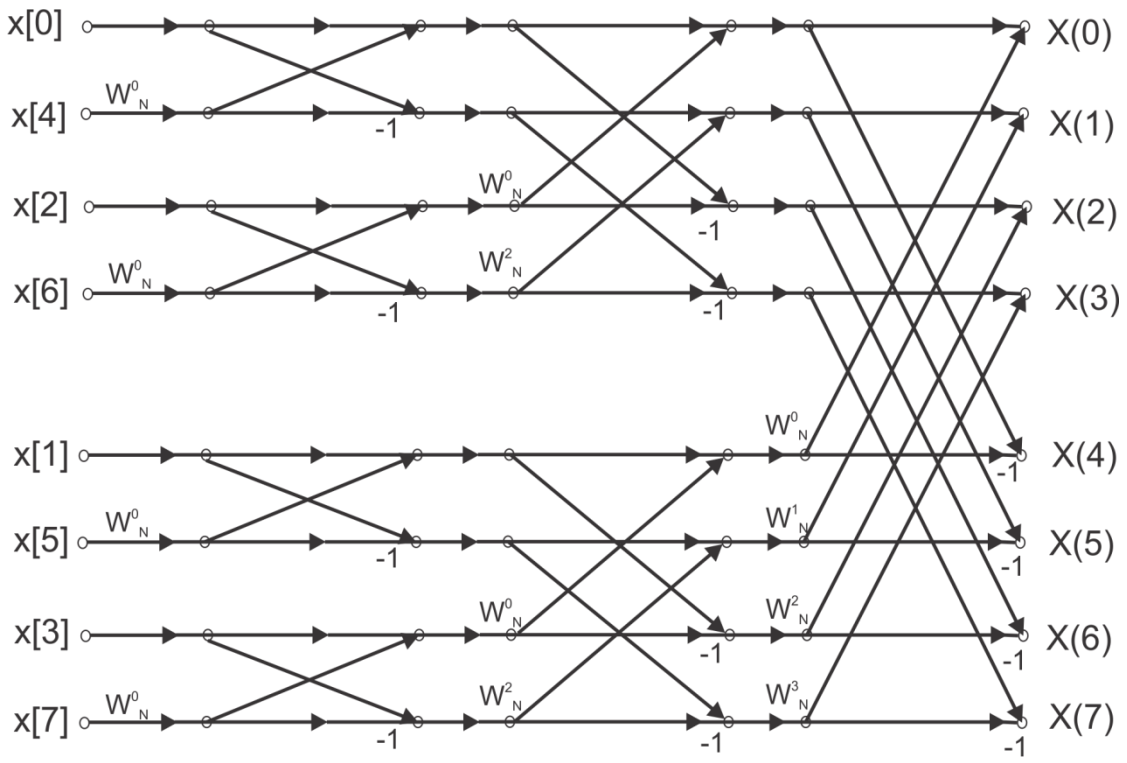

Figura 2.9 Algoritmo FFT para una N=8 puntos en un submuestreo de tiempo.

Como se observa en la Figura 2.10 se realiza la combinación de las DFT's de menos puntos para poder obtener una DFT mayor.

El cálculo de cada etapa consiste en primero tomar dos números digamos que es nuestro par de *(a, b),* después se multiplica la *b* por W_N^n y sumas y restas de *a* el resultado anterior de la multiplicación de b para obtener dos números complejos, digamos que los resultados ahora son (A, B), a esta operacion se le conoce como "Método de Mariposa" y se llama de esta manera debido a la forma del gráfico de las operaciones que tiene la forma de las alas de una mariposa, esta se muestra en la Figura 2.11.

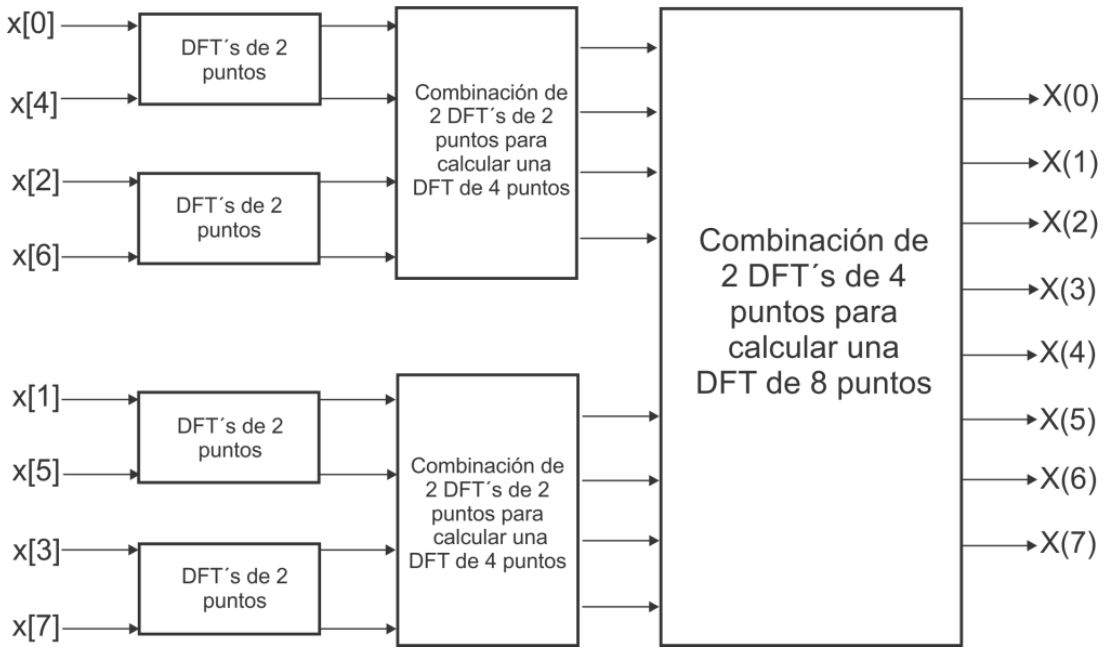

Figura 2.10. Las tres etapas en el cálculo de una DFT de 8 puntos.

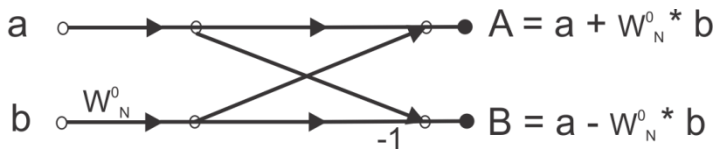

Figura 2.11. Forma básica de una mariposa para una FFT de submuestreo en tiempo.

Algunas cuestiones que se tienen en cuenta para poder entender más el algoritmo de la FFT son (Jones., 2006):

1. En cuanto a la mariposa como se ve en la Figura 2.11, es básicamente una multiplicación compleja y dos sumas complejas. Donde podemos decir que para $N = 2^v$tendremos N/2 mariposas por cada etapa en el caso de una FFT de N=8 puntos tendremos (8/2)= 4 mariposas.

   Y para saber cuantas etapas habrá lo definiremos como $v = log_2 (N)$ que en este caso serán 3 etapas, por lo tanto necesitaremos $\left(\frac{N}{2}\right) log_2(N)$ multiplicaciones complejas, las cuales para nuestro ejemplo de una FFT de 8 serán 12 y $N\ log_2(N)$ sumas complejas que serán 24.

2. Ahora bien después de que realizamos una mariposa sobre el par de números (a, b) y que obtenemos (A, B) no se necesita guardar el par (a, b) ya que estos ya no se utilizaras si no que ahora necesitaremos usar los valores de (A, B) por está razón podemos guardar en el mismo lugar del par (a, b) los nuevos pares (A, B).

3. Algo muy importante también es que en el diezmado de tiempo, el proceso de submuestreo de la secuencia de entrada se realiza mediante un proceso que se le llama orden de binario invertido el cual genera la siguiente secuencia que se muestra en la Tabla 2.2.

4. Para obtener cuantos serán los factores twiddle que necesitaremos lo podemos saber por medio de (N/2) que por ejemplo para un N=8, solo necesitaremos 4 factores twiddle (W^0, W^1, W^2, W^3).

Tabla 2.2 Orden original yorden final al aplicar la inversión de bits de la secuencia de entrada de x[n] para una N=8 puntos.

Orden Original	n		Orden Final	n	
	Decimal	Binario		Decimal	Binario
x(0)	0	000	x(0)	0	000
x(1)	1	001	x(4)	4	100
x(2)	2	010	x(2)	2	010
x(3)	3	011	x(6)	6	110
x(4)	4	100	x(1)	1	001
x(5)	5	101	x(5)	5	101
x(6)	6	110	x(3)	3	011
x(7)	7	111	x(7)	7	111

Tal vez para pequeños valores de N o de puntos, la diferencia que se puede tener es muy pequeña pero para valores más grandes la diferencia entre una DFT y una FFT es enorme. Por ejemplo para una DFT de 1024 puntos, el número de multiplicaciones en una FFT es aproximadamente de 5,000 mientras que par la DFT normal es aproximadamente de 1, 000,000, de aquí la importancia y la gran ventaja de la FFT, ya que reduce mucho el trabajo y realiza de una manera mas rápida y eficiente una DFT normal. Podemos mostrar un poco más estas diferencias en cuanto a la ventaja de usar una FFT contra una DFT en la Tabla 2.3 (Pasaye, 2010).

Tabla 2.3 Ventajas de la FFT contra la DFT.

N	Multiplicaciones usando cálculo directo (N^2) DFT	Multiplicaciones usando FFT [(*N/2*)$log_2 N$]
4	4	4
8	64	12
16	256	32
32	1,024	80
64	4,096	192
128	16,384	448
256	65,536	1,024
512	262,144	2,304
1,024	1,048,576	5,120

2.5.6 Ejemplo Transformada Rápida de Fourier (FFT)

Para realizar una FFT como se menciona anteriormente podemos conocer cuantas etapas tendrá, cuantas operaciones de mariposa, cuantas multiplicaciones complejas, cuantas sumas complejas y cuantos factores twiddle necesitaremos.

Como ejemplo realizaremos una FFT de 8 puntos, que tendrá como datos a evaluar: 5, 10, 15, 20, 25, 30, 35, 40. Primero evaluaremos:

Número de Etapas que se tendrán para N = 8, que determinara la Ecuación (46).

$v = log_2(N) = log_2(8) = 3$ etapas (46)

Número de operaciones Mariposa, que se obtendrán de la Ecuación (47).

N/2 = 8/2 = 4 mariposas (47)

Número de multiplicaciones Complejas, que se obtienen de la Ecuación (48).

$\left(\frac{N}{2}\right) log_2(N) = \left(\frac{8}{2}\right) log_2(8)$ = 12 multiplicaciones (48)

Número de sumas Complejas, que se obtienen de la Ecuación (49).

$N\ log_2(N) = 8\ log_2(8)$ = 24 sumas (49)

Numero de factores Twiddle:

N/2 = 8/2 = 4 twiddle, los cuales serán W^0, W^1, W^2, W^3 (50)

Como se muestra en la ecuación (45), se puede saber cual es el valor de cada uno de los factores los cuales se muestran de la Ecuación (51) a la (54).

$$W_8^0 = \cos\left(\frac{2\pi(0)}{8}\right) - j\ sen\ \left(\frac{2\pi(0)}{8}\right) = \cos(0) - jsen(0) \quad (51)$$

$$\boldsymbol{W_8^0 = 1}$$

$$W_8^1 = \cos\left(\frac{2\pi(1)}{8}\right) - j\ sen\ \left(\frac{2\pi(1)}{8}\right) = cos\left(\frac{\pi}{4}\right) - j\ sen\ \left(\frac{\pi}{4}\right) \quad (52)$$

$$\boldsymbol{W_8^1 = 0.7071 - j0.7071}$$

$$W_8^2 = \cos\left(\frac{2\pi(2)}{8}\right) - j\ sen\ \left(\frac{2\pi(2)}{8}\right) = cos\left(\frac{4\pi}{8}\right) - j\ sen\ \left(\frac{4\pi}{8}\right) \quad (53)$$

$$\boldsymbol{W_8^2 = -j}$$

$$W_8^3 = \cos\left(\frac{2\pi(3)}{8}\right) - j\ sen\ \left(\frac{2\pi(3)}{8}\right) = cos\left(\frac{6\pi}{8}\right) - j\ sen\ \left(\frac{6\pi}{8}\right) \quad (54)$$

$$\boldsymbol{W_8^3 = -0.7071 - j0.7071}$$

Teniendo estos datos, se puede seguir con la resolución de la FFT, no se debe olvidar que para el diezmado de tiempo que es con el que se realiza, su submuestreo lo hace con una secuencia de entrada la cual se muestra en la Tabla 2.2, donde se aplica la inversión de bits. Y el cual para este caso se muestra en la Tabla 2.4.

Tabla 2.4 Inversión de bits de los datos originales de la FFT.

Orden Original	Dato (a + b), donde a= parte real y b= parte imaginaria	Orden Invertido	Dato (a + b), donde a= parte real y b= parte imaginaria
0	5 + 0	0	5 + 0
1	10 + 0	4	25 + 0
2	15 + 0	2	15 + 0
3	20 + 0	6	35 + 0
4	25 + 0	1	10 + 0
5	30 + 0	5	30 + 0
6	35 + 0	3	20 + 0
7	40 + 0	7	40 + 0

Ahora se realizan las operaciones con los datos de la Tabla 2.4 que se muestran en la Figura 2.12, mediante el diagrama mostrado en la Figura 2.9 para cada una de las etapas de la FFT de 8 puntos, tomando en cuenta la forma básica de la mariposa que se muestra en la Figura 2.11 y los factores twiddle resultantes de las Ecuaciones (51) a (54).

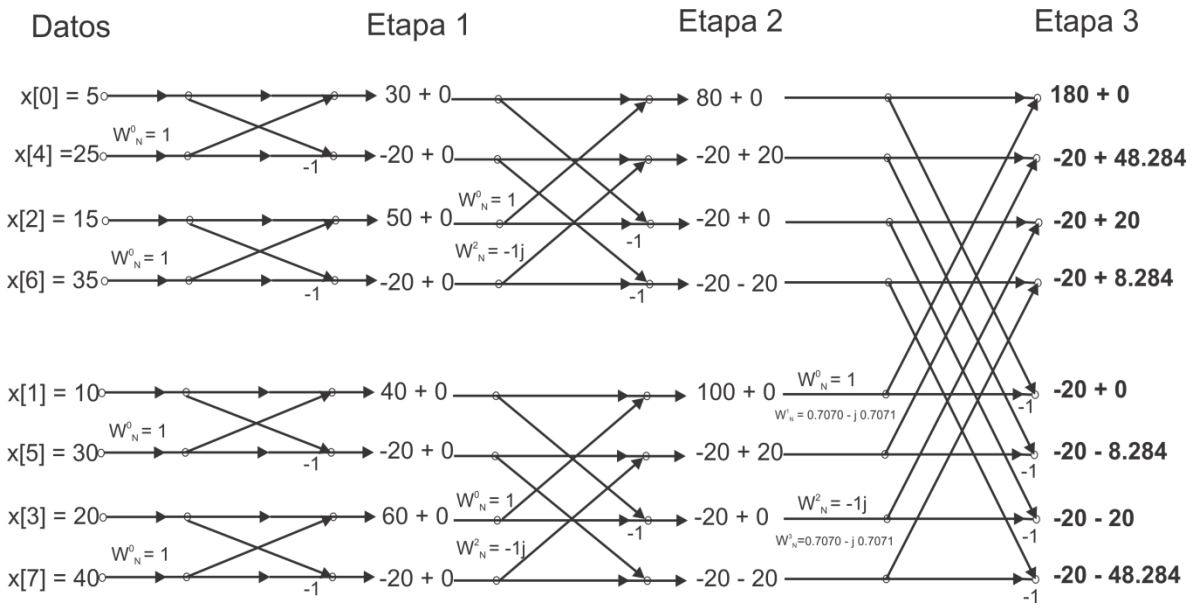

Figura 2.12. Evaluación y Resultados de la FFT de 8 puntos.

Ya que se tiene el resultado de la FFT en la etapa final 3, se puede comparar con los resultados de la FFT que se obtienen en MatLab, los cuales se muestran en la Tabla 2.5 y se comprueba que los resultados son correctos.

Tabla 2.5 Comparación de Resultados para una FFT N = 8 puntos.

Resultado de la evaluación FFT (Etapa 3)	Resultados en MatLab multiplicados por (1.0e+002)
180 + 0	1.8000
-20 + 48.28	-0.2000 + 0.4828i
-20 + 20	-0.2000 + 0.2000i
-20 + 8.28	-0.2000 + 0.0828i
-20 + 0	-0.2000
-20 – 8.28	-0.2000 - 0.0828i
-20 – 20	-0.2000 - 0.2000i
-20 – 48.28	-0.2000 - 0.4828i

Como segundo ejemplo se realiza una FFT de N = 16 puntos, para la cual se tiene:

Número de Etapas resultante que se muestran en Ecuación (55):

$v = log_2(N) = log_2(16) = 4$ etapas (55)

Numero de operaciones Mariposa mostradas en Ecuación (56):

N/2 = 16/2 = 8 mariposas (56)

Numero de multiplicaciones Complejas, obtenidas de la Ecuación (57):

$\left(\frac{N}{2}\right)\log_2(N) = \left(\frac{16}{2}\right)\log_2(16) = 32$ multiplicaciones (57)

Numero de sumas Complejas, obtenidas de la Ecuación (58):

$N\log_2(N) = 16\log_2(16) = 64$ sumas (58)

Numero de factores Twiddle que se obtienen en la Ecuación (59):

N/2 = 16/2 = 8 *twiddle,* los cuales serán W^0, W^1, W^2, W^3 W^4, W^5, W^6, W^7 (59)

Evaluando la ecuación (45), se obtienen los valores de cada uno de los factores twiddle de la Ecuación (59), dando como resultado las Ecuaciones (60) a la (67):

$W_{16}^0 = \cos\left(\frac{2\pi(0)}{16}\right) - j\ sen\ \left(\frac{2\pi(0)}{16}\right) = \cos(0) - jsen(0)$ (60)

$\boldsymbol{W_{16}^0 = 1}$

$$W_8^1 = \cos\left(\frac{2\pi(1)}{16}\right) - j\ sen\ \left(\frac{2\pi(1)}{16}\right) = cos\left(\frac{\pi}{8}\right) - j\ sen\ \left(\frac{\pi}{8}\right) \quad (61)$$

$$\boldsymbol{W_{16}^1 = 0.9238 - j0.3827}$$

$$W_{16}^2 = \cos\left(\frac{2\pi(2)}{16}\right) - j\ sen\ \left(\frac{2\pi(2)}{16}\right) = cos\left(\frac{\pi}{4}\right) - j\ sen\ \left(\frac{\pi}{4}\right) \quad (62)$$

$$\boldsymbol{W_{16}^2 = 0.7071 - j0.7071}$$

$$W_{16}^3 = \cos\left(\frac{2\pi(3)}{16}\right) - j\ sen\ \left(\frac{2\pi(3)}{16}\right) = cos\left(\frac{6\pi}{16}\right) - j\ sen\ \left(\frac{6\pi}{16}\right) \quad (63)$$

$$\boldsymbol{W_{16}^3 = 0.3827 - j0.9239}$$

$$W_{16}^4 = \cos\left(\frac{2\pi(4)}{16}\right) - j\ sen\ \left(\frac{2\pi(4)}{16}\right) = cos\left(\frac{\pi}{2}\right) - j\ sen\ \left(\frac{\pi}{2}\right) \quad (64)$$

$$\boldsymbol{W_{16}^4 = -j}$$

$$W_{16}^5 = \cos\left(\frac{2\pi(5)}{16}\right) - j\ sen\ \left(\frac{2\pi(5)}{16}\right) = cos\left(\frac{10\pi}{16}\right) - j\ sen\ \left(\frac{10\pi}{16}\right) \quad (65)$$

$$\boldsymbol{W_{16}^5 = -0.3827 - j0.9239}$$

$$W_{16}^6 = \cos\left(\frac{2\pi(6)}{16}\right) - j\ sen\ \left(\frac{2\pi(6)}{16}\right) = cos\left(\frac{12\pi}{16}\right) - j\ sen\ \left(\frac{12\pi}{16}\right) \quad (66)$$

$$\boldsymbol{W_{16}^6 = -0.7071 - j0.7071}$$

$$W_{16}^7 = \cos\left(\frac{2\pi(7)}{16}\right) - j\ sen\ \left(\frac{2\pi(7)}{16}\right) = cos\left(\frac{14\pi}{16}\right) - j\ sen\ \left(\frac{14\pi}{16}\right) \quad (67)$$

$$\boldsymbol{W_{16}^7 = -0.9239 - j0.3827}$$

La solución de la FFT de 16 puntos se muestra en la Tabla 2.7, para lo cual se utilizan los factores twiddle resultantes de las ecuaciones (60) a (67) y en la Tabla 2.6 se muestra la inversión de los datos originales.

Tabla 2.6 Inversión de datos originales de la FFT N=16.

Orden Original	Dato (a + b), donde a= parte real y b= parte imaginaria	Orden Invertido	Dato (a + b), donde a= parte real y b= parte imaginaria
0	0 + 0	0	0 + 0
1	8 + 0	8	1 + 0
2	4 + 0	4	2 + 0
3	12 + 0	12	3 + 0
4	2 + 0	2	4 + 0
5	10 + 0	10	5 + 0
6	6 + 0	6	6 + 0
7	14 + 0	14	7 + 0
8	1 + 0	1	8 + 0
9	9 + 0	9	9 + 0
10	5 + 0	5	10 + 0
11	13 + 0	13	11 + 0
12	3 + 0	3	12 + 0
13	11 + 0	11	13 + 0
14	7 + 0	7	14 + 0
15	15 + 0	15	15 + 0

Tabla 2.7 Evaluación y Resultados de la FFT de 16 puntos.

Datos (a + b)	Resultado, Etapa 1	Resultado, Etapa 2	Resultado, Etapa 3	Resultado Final de la FFT, Etapa 4
0 + 0	8+ 0	24 + 0	56 + 0	120 + 0
1 + 0	-8 + 0	-8 + 8	-8 + 19.31	-8 + 40.22
2 + 0	16 + 0	-8 + 0	-8 + 8	-8 + 19.31
3 + 0	-8 + 0	-8 - 8	-8 + 3.31	-8 + 11.97
4 + 0	12 + 0	32 + 0	-8 + 0	-8 + 8
5 + 0	-8 + 0	-8 + 8	-8 – 3.31	-8 + 5.35
6 + 0	20 + 0	-8 + 0	-8 – 8	-8 + 3.31
7 + 0	-8 + 0	-8 - 8	-8 - 19.31	-8 + 1.59
8 + 0	10 + 0	28 + 0	50.31 + 0	-8 + 0
9 + 0	-8 + 0	-8 + 8	-17.68 + 9.63	-8 – 1.59
10 + 0	18 + 0	-8 + 0	-8 – 5.69	-8 - 3.31
11 + 0	-8 + 0	-8 – 8	1.68 – 6.37	-8 – 5.35
12 + 0	14 + 0	22.31 + 0	5.69 + 0	-8 - 8
13 + 0	-8 + 0	-8 – 5.69	1.68 + 6.37	-8 – 11.97
14 + 0	22 + 0	5.69 + 0	-8 + 5.69	-8 – 19.31
15 + 0	-8 + 0	-8 + 5.69	-17.68 – 9.63	-8 – 40.22

Comparando con los obtenidos en MatLab, que se muestran en la Tabla 2.8, vemos que son semejantes.

Tabla 2.8 Comparación de Resultados para una FFT N = 16 puntos.

Resultado de la evaluación FFT (Etapa 3)	Resultados en MatLab multiplicados por (1.0e+002)
120 + 0	1.2000
-8 + 40.22	-0.0800 + 0.4022i
-8 + 19.31	-0.0800 + 0.1931i

Tabla 2.8 (Continuación)

Resultado de la evaluación FFT (Etapa 3)	Resultados en MatLab multiplicados por (1.0e+002)
-8 + 11.97	-0.0800 + 0.1197i
-8 + 8	-0.0800 + 0.0800i
-8 + 5.35	-0.0800 + 0.0535i
-8 + 3.31	-0.0800 + 0.0331i
-8 + 1.59	-0.0800 + 0.0159i
-8 + 0	-0.0800
-8 – 1.59	-0.0800 - 0.0159i
-8 - 3.31	-0.0800 - 0.0331i
-8 – 5.35	-0.0800 - 0.0535i
-8 - 8	-0.0800 - 0.0800i
-8 – 11.97	-0.0800 - 0.1197i
-8 – 19.31	-0.0800 - 0.1931i
-8 – 40.22	-0.0800 - 0.4022i

2.6 Apoyo de diseño con Matlab

MatLab es el principal producto de software de Mathworks, Inc., fundada por los analistas numéricos Cleve Moler y John N., Little. Como su nombre lo indica, este se desarrolló originalmente como un laboratorio para matrices. Hoy, el elemento principal de MatLab sigue siendo la matriz. La manipulación matemática de matrices se ha realizado muy adecuadamente en el ambiente interactivo fácil de utilizar. A esta manipulación matricial agrega también varias funciones numéricas, cálculos simbólicos y herramientas para visualización. MatLab tiene diferentes funciones y operadores que permiten la adecuada realización de los diversos métodos numéricos y cálculos numéricos.

Es útil como herramienta de diseño ya que esta herramienta cuenta con gran flexibilidad y facilidad en la programación, permitiendo la implementación y simulación del código de manera rápida (Chapra, 2007).

Algunas de las facilidades que da MatLab es por ejemplo en este caso haber realizado el estudio de algunos métodos numéricos, como mostrar los armónicos y la suma de los mismos de una serie de Fourier como en la siguiente implementación, en la cual se muestra la Serie de Fourier del primer armónico o frecuencia fundamental, hasta el quinto armónico, donde la frecuencia fundamental para $\omega_0 = \pi$, con el resultado que se muestra en la Figura 2.13.

```
%Universidad Autónoma de Querétaro
%Alejandra Miguel Vargas Mandujano. Exp.136150
% Componentes de la Serie de Fourier
clear all;
clc;
% Primer armónico o frecuencia fundamental de la señal cuadrada en azul
t=0:.0001:.01
wo= 1000
y=4*sin(t*wo)/pi;
plot(t,y)
hold  on
%el segundo armonico en verde
y=(4/pi)*[sin(3*t*wo)/3];
hold on
plot(t,y,'g')
%el tercer armonico en amarillo
y=(4/pi)*[sin(5*t*wo)/5];
hold on
plot(t,y,'y')
%el cuarto armonico en rojo
y=(4/pi)*[sin(7*t*wo)/7];
hold on
plot(t,y,'r')
%el quinto armonico en magenta
y=(4/pi)*[sin(9*t*wo)/9];
hold on
plot(t,y,'m')
%Resultado de la Serie de Fourier, sumando todos los armonicos
y=(4/pi)*[sin(t*wo)+sin(3*t*wo)/3+sin(5*t*wo)/5+sin(7*t*wo)/7+sin(9*t*wo)
/9];
plot(t,y,'k', 'LineWidth',3)
title('Componentes de la Serie de Fourier');
```

Gracias a la flexibilidad que proporciona MatLab con la programación, en este caso realizando con un ciclo *for,* como se muestra en el siguiente código, se ve que mientras mas se agregan armónicos a la serie de Fourier se tiene un resultado más cercano a la señal deseada que es un señal cuadrada. En la Figura 2.14 se muestra la grafica con un total de 10 armónicos, en la Figura 2.15 se muestra con 50 armónicos y por ultimo en la Figura 2.16 se muestra con 300 armónicos en la cual se ve que se acerca más a una señal cuadrada.

Otras de las herramientas que se utilizaron fueron las de crear archivos desde MatLab en formato *.vhd, con código especifico, el cual nos ayudo a poder manejar de una forma genérica los códigos que se implementaron en VHDL, por ejemplo para evaluar algunas FFT de N puntos, al utilizar MatLab, se pudieron realizar cambios rapidos en algunas partes de los códigos implementados en Hardware, como se menciono anteriormente en el caso de los N puntos que se desean evaluar, como ejemplo de una FFT de N = 8 puntos, se pueden cambiar gracias a la versatilidad de MatLab a una implementación para una FFT de N = 256 puntos, estos ejemplos se muestran en el capitulo 3 donde se encuentran algunas implementaciones realizadas desde MatLab que generaron los archivos *.vdh para implementarlos en el lenguaje de programación para Hardware VHDL.

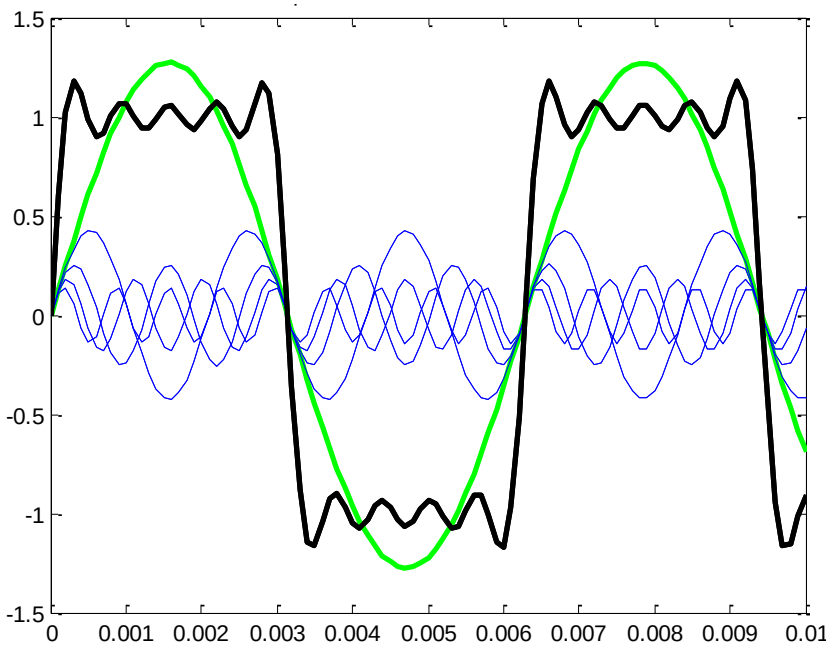

Figura 2.13 Componentes de la Serie de Fourier

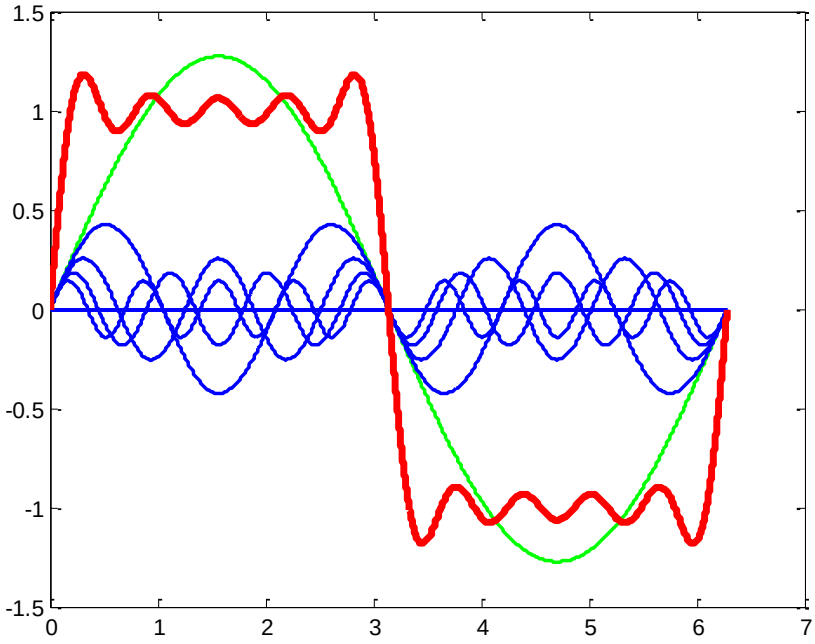

Figura 2.14. Señal con 10 armónicos de la Serie de Fourier

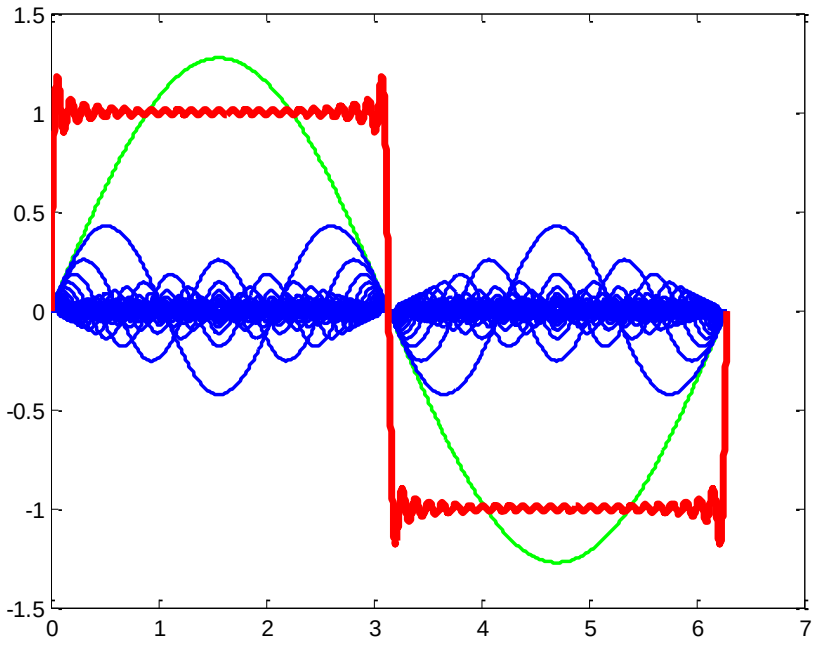

Figura 2.15 Señal con 50 armónicos de la Serie de Fourier

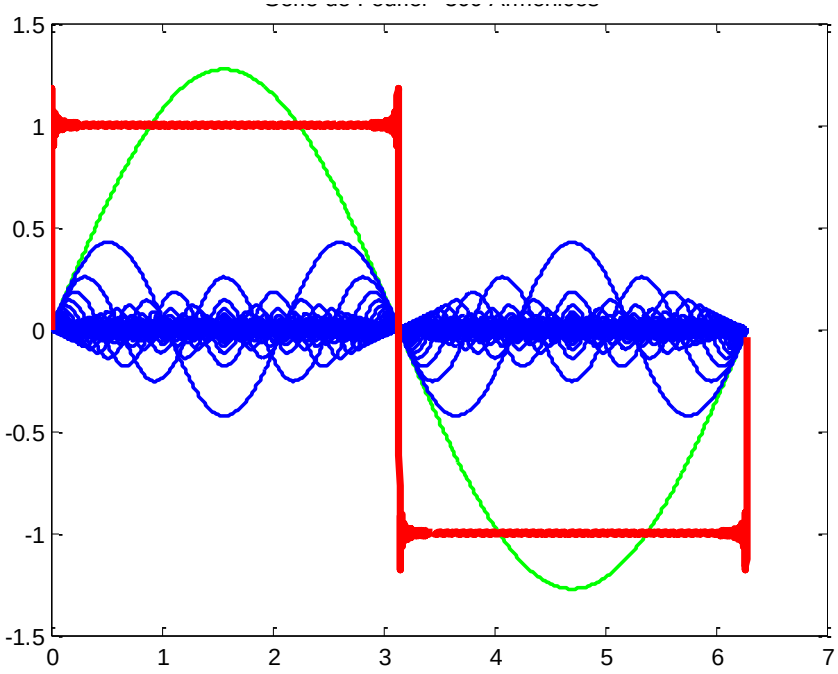

Figura 2.16 Señal con 300 armónicos de la Serie de Fourier

CAPÍTULO III

Desarrollo

3.1 Metodología de diseño del Método en VHDL

Para poder tener una mejor sincronización de cada uno de los procedimientos y técnicas que se requieren para realizar la implementación en Hardware se creó la metodología que se muestra en la Figura 3.17.

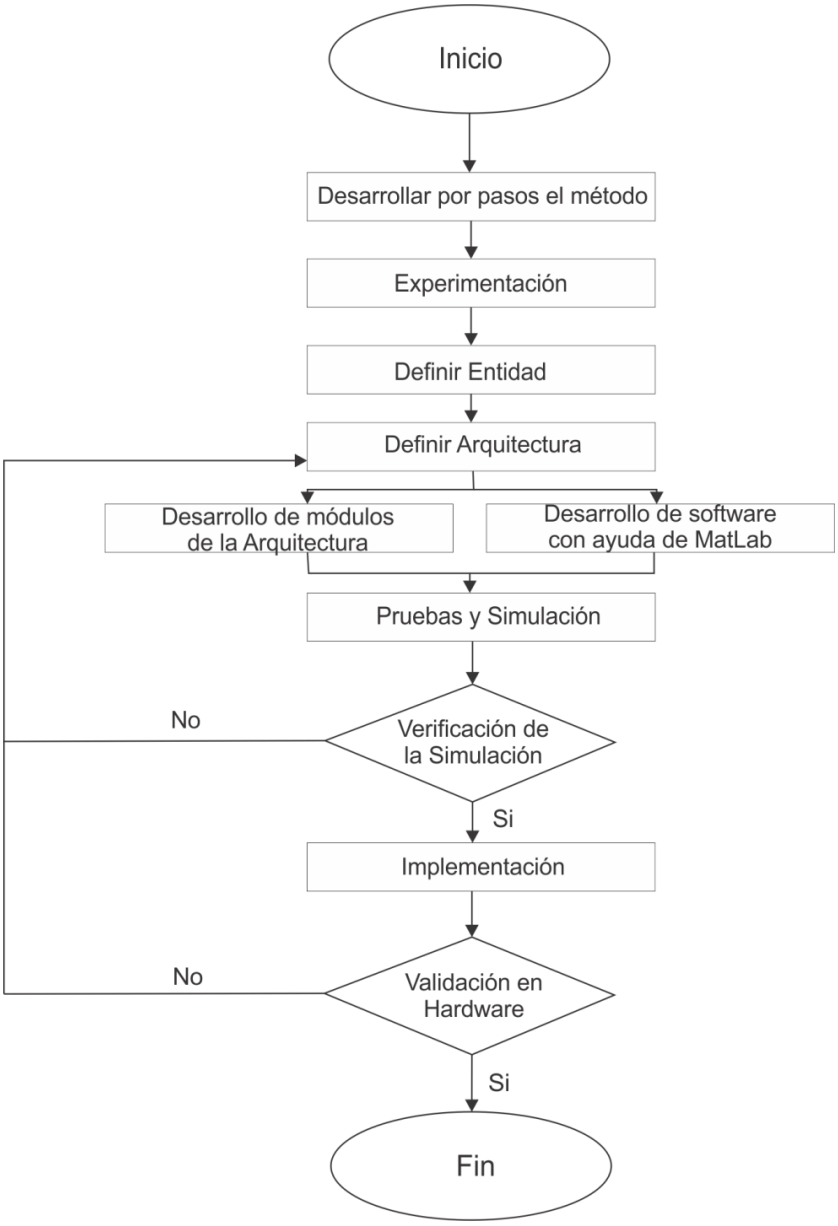

Figura 3.17 Metodología de la Implementación en VHDL.

3.2 Implementación de la Aproximación Polinomial de la Serie de Taylor Función Básica (Seno)

El primer método implementado en Hardware es una Aproximación polinomial de $f(x) = \sin(x)$ en el intervalo cerrado de $0 <= x <= 2\pi$, como se muestra en la Figura 3.18 que es la entidad general de la aproximación, las señales que contiene son el reloj maestro CLK, el RST que sirve para resetear la implementación, STF que indica en que momento debe comenzar la evaluación de la aproximación, el X que es el dato a evaluar con nuestra función seno, Y es la salida de nuestra evaluación y el EOF que indica en que momento todo el proceso termino.

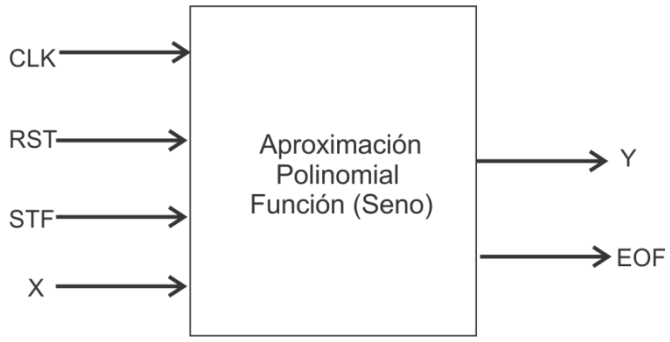

Figura 3.18 Entidad General de la Aproximación Polinomial (Seno).

Para realizar la implementación en Hardware, se toma como referencia la implementación “Rutina Matlab para diseño de una aproximación polinomial del tipo sigmoide” de Rene Romero Troncoso, que se muestra acontinuación, con el cual se puede definir el diagrama de bloques que se utiliza para la aproximación en la implementación en FPGA de la función seno, de igual manera con esta implementación se puede obtener la LUT (lookup table) que contiene cada uno de los coeficientes de la aproximación polinomial en forma de ROM.

En la Figura 3.19 se muestra la arquitectura de la aproximación polinomial, la cual contiene un sumador, un multiplicador, un acumulador, una LUT, un contador, una maquina de estados (FSM) y un registro, los cuales se encuentran dentro de la caja negra que es la entidad general que se muestra en la Figura 3.18.

La implementación cuenta con una especificación muy importante, que es el manejo del punto fijo, para poder tener un resultado mas óptimo; en este caso el punto fijo que se utiliza es de 2.16, se define de esta manera porque se tiene como resultado un entero positivo o un entero negativo, por ejemplo para el resultado del seno se obtuvo un intervalo de 1 a -1, por lo cual los dos primeros bits determinarán si será 1 o -1 donde el primer bit será el bit de signo y los otros 16 serán de la parte decimal del resultado.

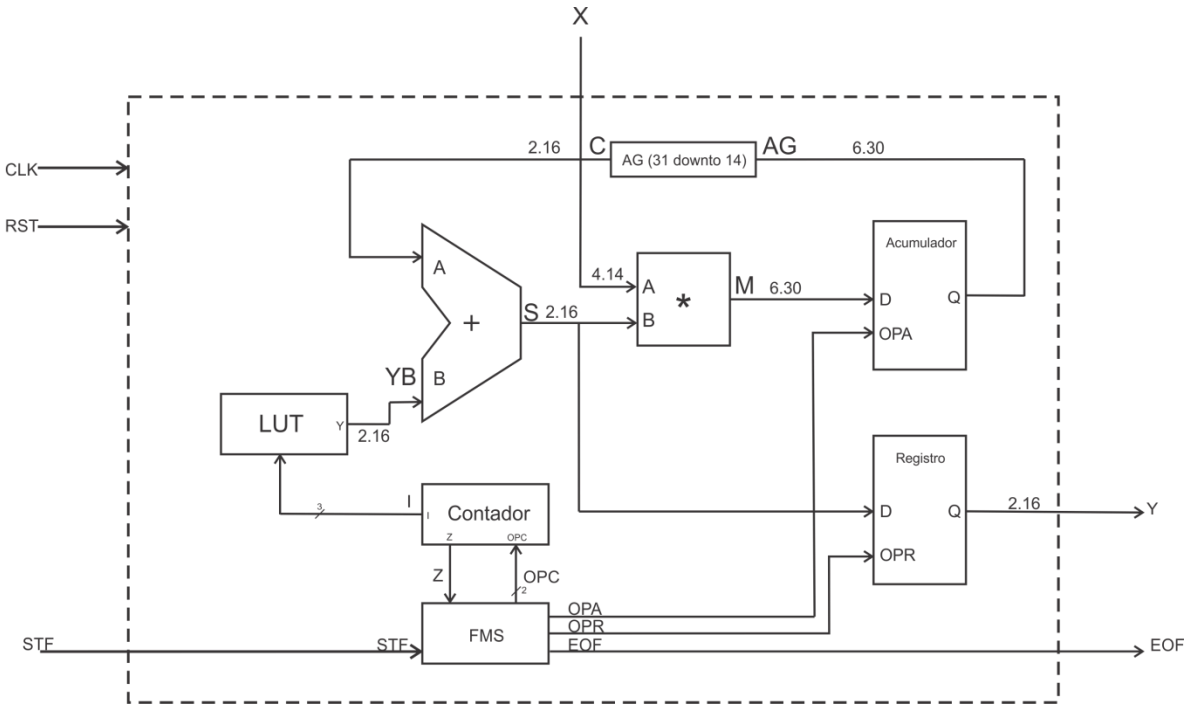

Figura 3.19 Arquitectura de la Aproximación Polinomial.

El bloque LUT de la Figura 3.20, que como antes se menciona se generó con la ayuda de la implementación de Romero Troncoso, contiene cada uno de los

coeficientes que se necesitaran para poder evaluar la aproximación polinomial; el cual será de 3 bits de entrada “I” con la que se indica que coeficiente tendrá que mandar al sumador mediante su salida “A” que será de 18 bits con un formato de punto fijo de 2.16.

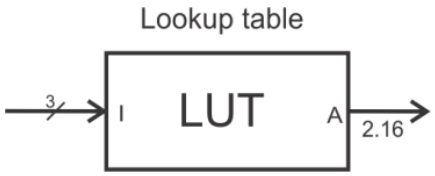

Figura 3.20 LUT, con un punto fijo de 2.16

El bloque Contador que se muestra en la Figura 3.21 controla las salidas de la LUT, el cual tiene como función cargar el coeficiente de la LUT dependiendo de la combinación, las cuales son “0x” mantener, “10” cargar el valor y “11” decrementar el valor de los coeficientes, esto significa que empiece con el coeficiente mas alto de $a_6$hasta llegar al a_0.

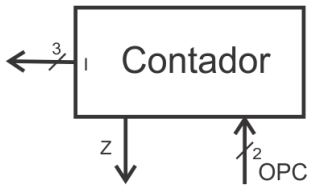

Figura 3.21 Contador Multifuncional.

El bloque FSM (Finite-state machine) de la Figura 3.22 genera cada una de las instrucciones que realiza el sistema para que se hagan las tareas dependiendo de cada estado que se defina.

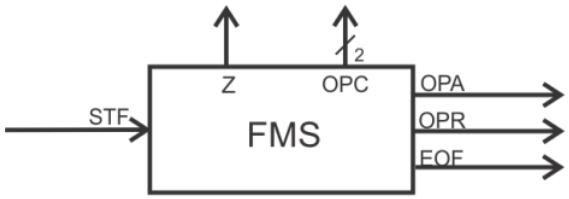

Figura 3.22 Maquina de Control (FSM).

El bloque Acumulador de la Figura 3.23 carga el resultado de la suma y multiplicación de cada uno de los coeficientes por “x”, retiene el dato del resultado dependiendo de la señal OPA la cual indica en que momento y hasta que momento estar actualizando el acumulador, este entrega el resultado de cada una de las evaluaciones que se muestran en la Tabla 3.9.

A la salida del acumulador se le hace un ajuste del punto fijo, ya que hay que mandar esta salida al sumador que es de 18 bits con un punto fijo de 2.16 y la salida del acumulador es de 36 bits con un formato de 6.30, por tanto, antes de mandar el resultado al sumador se tiene que hacer el ajuste de bits para poderhacer la reducción de 6.30 a 2.16 y que de esta manera se pueda procesar en el sumador.

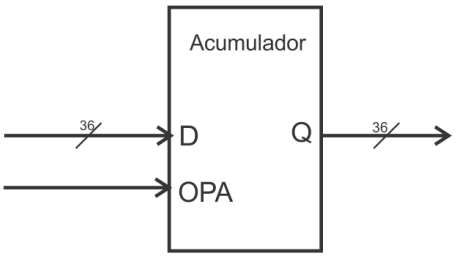

Figura 3.23 Acumulador.

Tabla 3.9 Orden de las Operaciones.

Operación 1	$(a_6 + 0)x$
Operación 2	$(a_6x + a_5)x = a_6x^2 + a_5$
Operación 3	$(a_6x^2 + a_5x + a_4)x = a_6x^3 + a_5x^2 + a_4x$
Operación 4	$(a_6x^3 + a_5x^2 + a_4x + a_3)x = a_6x^4 + a_5x^3 + a_4x^2 + a_3x$
Operación 5	$(a_6x^4 + a_5x^3 + a_4x^2 + a_3x + a_2)x = a_6x^5 + a_5x^4 + a_4x^3 + a_3x^2 + a_2x$
Operación 6	$(a_6x^5 + a_5x^4 + a_4x^3 + a_3x^2 + a_2x + a_1)x = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x$

El bloque sumador de la Figura 3.24 será de 18 bits con el formato de 2.16 que sumara el resultado del acumulador y de cada coeficiente que se esté mandando de la LUT.

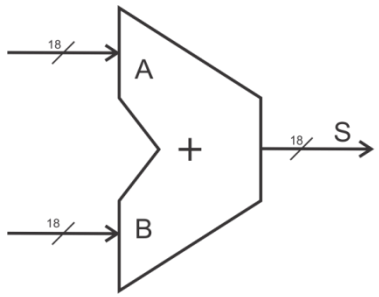

Figura 3.24 Sumador de 18 bits.

El bloque multiplicador de la Figura 3.25 es de 18 bits y este dará el resultado de la multiplicación de “x” por cada uno de los coeficientes en un formato de 6.30 y cual entrara al acumulador con un formato igual a 6.30.

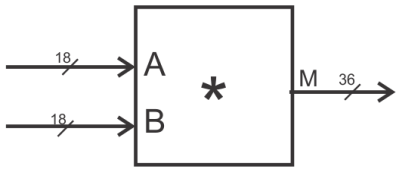

Figura 3.25 Multiplicador 18 bits.

El bloque Registro de la Figura 3.26 ayuda a poder terminar la función, ya que con el acumulador se llega solo hasta la evaluación del polinomio $a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x$, y la aproximación polinomial que nosotros buscamos es:

$$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6$$

Por tanto el resultado de la evaluación se tiene hasta el a_1x y faltaría sumar el coeficiente a_0, el cual se mandara directamente de la salida del sumador hacia el registro de 18 bits con el formato de punto fijo de 2.16, este cargara el dato de la suma de la evaluación, que se tiene del acumulador, y sumando el coeficiente a_0 cuando le llegue la señal OPR de nuestra FSM y de esta manera se obtendrá el resultado de la función seno que saldrá en “Y” que es la salida principal de la entidad general Figura 3.18.

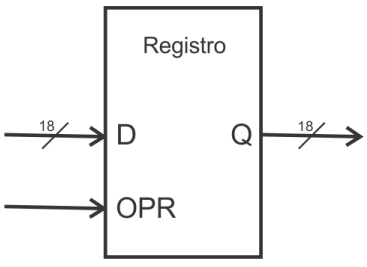

Figura 3.26 Registro 18 bits.

3.3 Implementación de la Transformada Rápida de Fourier (FFT)

3.3.1 Implementación de la FFT en Matlab:

Con la teoría y ejemplos que se realizaron en el capitulo anterior del algoritmo de la FFT, se realizó la implementación del mismo en Matlab, con este se puede ver el comportamiento y funcionamiento aplicado en software, gracias a esto se tuvo una idea mas sintetizada para aplicarlo en hardware.

Una vez que se implemento, se realizaron pruebas con la función ya establecida de Matlab (fft) para comprobar los resultados que se obtuvo de la implementación que se realizó antes , una de las prubas se realizo con una FFT de N= 8 puntos, que se muestra en la Tabla 3.10.

Tabla 3.10 Comparación de Resultados de FFT con N=8.

Datos	Resultados con la Implementación realizada de la FFT (V)	Resultados con la función de MatLab (Y)
1	36	36.00
2	-4 + 9.6569i	-4.00 + 9.6569i
3	-4.00 + 4.00i	-4.00 + 4.00i
4	-4.00 + 1.6569i	-4.00 + 1.6569i
5	-4.00	-4.00
6	-4.00 – 1.6569i	-4.00 – 1.6569i
7	-4.00 – 4.00i	-4.00 – 4.00i
8	-4.00 – 9.6569i	-4.00 – 9.6569i

Se observo que da el mismo resultado tanto en la implementación como con la función de MatLab, también se aplico a una FFT de N = 16 puntos, y los resultados se muestran en la Tabla 3.11, se toma en cuenta que dependiendo del grado N de la FFT cambiaran las mariposas que se realizan, las etapas y el orden de los factores twiddle, por tanto, automáticamente se realiza de forma adecuada

cada una de las operaciones con los factores twiddle correctos. De esta manera se usara la Formula (68):

$$kth = k * \frac{N}{2^i} \tag{68}$$

Donde k es el número de W o factores twiddle de acuerdo a las etapas *i* con un intervalo de 0 a N/2-1 y N es el número de muestras en potencia de 2.

Tabla 3.11 Comparación de Resultados función MatLab e implementación realizada en MatLab, FFT de N=16.

Datos	Resultados con la Implementación realizada de la FFT (V)	Resultados con la función de MatLab (Y)
1	136 + 0i	136 + 0i
2	-8 + 40.22i	-8 + 40.22i
3	-8 + 19.31i	-8 + 19.31i
4	-8 + 11.97i	-8 + 11.97i
5	-8 + 8i	-8 + 8i
6	-8 + 5.35i	-8 + 5.35i
7	-8 + 3.31i	-8 + 3.31i
8	-8 + 1.59i	-8 + 1.59i
9	-8 + 0i	-8 + 0i
10	-8 - 1.59i	-8 - 1.59i
11	-8 - 3.31i	-8 - 3.31i
12	-8 - 5.35i	-8 - 5.35i
13	-8 – 8i	-8 – 8i
14	-8 - 11.97i	-8 - 11.97i
15	-8 - 19.31i	-8 - 19.31i
16	-8 - 40.22i	-8 - 40.22i

Para el algoritmo que se implementa se usa la decimación en tiempo, que como antes ya se mencionó, los datos de entrada son previamente ordenados vía bit-reverso de manera que luego de realizar la FFT se obtiene el orden correcto de cada uno de los datos.

3.3.2 Implementación de la FFT en Hardware (VHDL)

La segunda implementación en Hardware es la FFT (Transformada Rápida de Fourier), que con la implementación antes vista en MATLAB y el estudio previamente realizado en el capitulo anterior, se analiza la manera más adecuada para implementar el algoritmo en Hardware.

Para realizar la FFT y verificar el resultado de esta implementación, se utiliza la comunicación serial la cual envía los datos desde una PC hacia un FPGA, donde al recibir cada uno de los datos , estos se almacenan en un bloque de memoria RAM y se envían al bloque de la FFT para realizar la evaluación de la misma y a su vez al termino de la evaluación los resultados se alamacenan en la memoria RAM y se envián nuevamente vía serial de la memoria que se encuentra implementada en el FPGA a la PC, esto para comprobar los resultados obtenidos de hardware comparados con los obtenidos de la implementación de software.

En la Figura 3.27 se muestra como es que se interconectan la recepción y transmición serial, junto con la memoria RAM, el bloque de la FFT y con un control de acceso hacia la RAM que es el encargado del control en cuanto al control de acceso a las memorias RAM y a la recepción y la transmición.

La entidad general de la FFT de la Figura 3.28, tiene las entradas SOF con la cual se indica en que momento comenzará la operación general de la FFT; R1, R2, I1, I3 que son los números reales e imaginarios a evaluar respectivamente, y tiene las salidas DER, DEI, que son el resultado final de la evaluación de la FFT la parte real e imaginaria respectivamente, la salida AE que envía la dirección de escritura para la memoria RAM, A1 y A2 que indica las

direcciones de lectura de la RAM para poder leer y mandar de nuevo a la FFT los nuevos datos a evaluar, se tiene la salida WE la cual indica en que momento se escribirá la dirección enviada mediante la señal AE, por último se tiene la señal EOF esta indica en que momento termina la evaluación y de esta manera comenzar a enviar la información resultante como antes lo mencionamos de la FPGA a la PC para poder comprobar los resultados obtenidos.

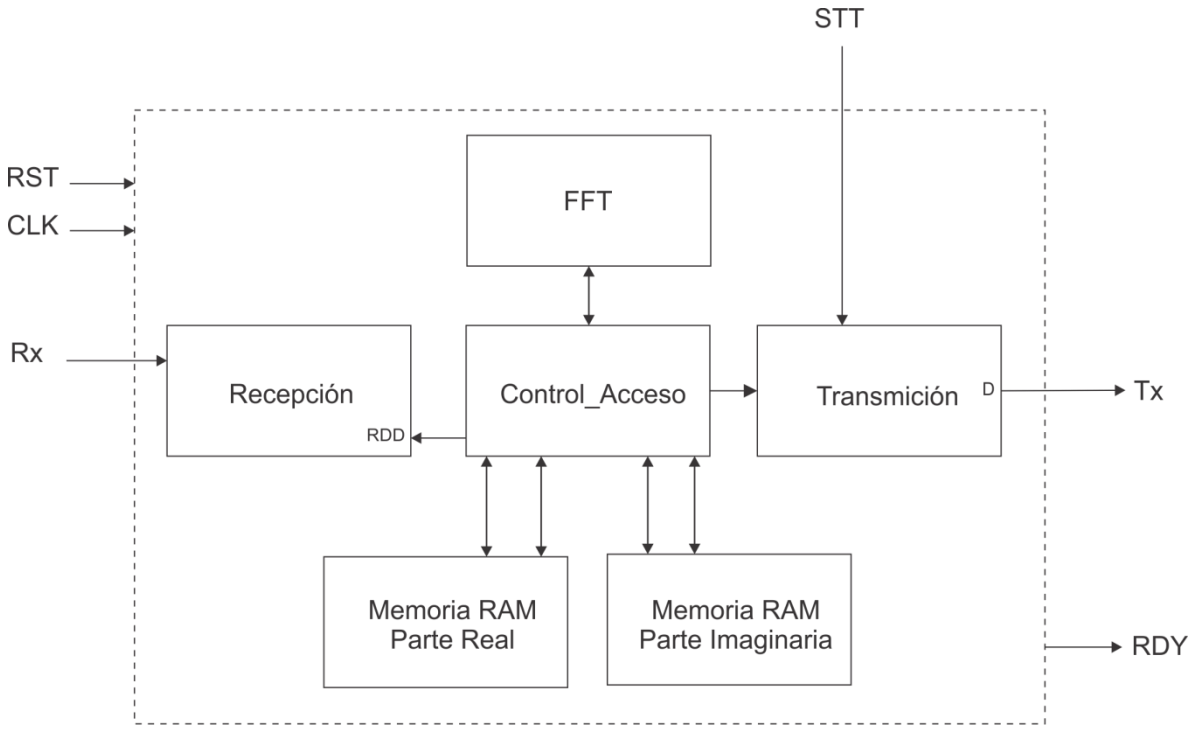

Figura 3.27 Comunicación Serial RS-232

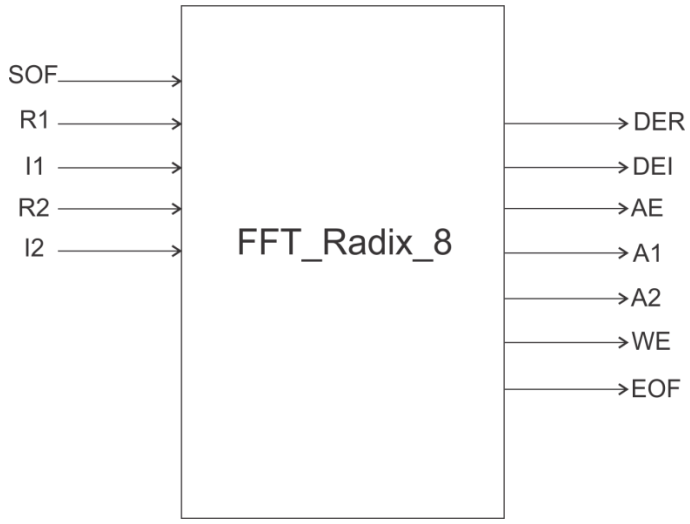

Figura 3.28 Entidad General de la FFT

Cada uno de los bloques que se encuentran en la Figura 3.29, muestran la arquitectura de la FFT de la Figura 3.28, el bloque Mariposa_FFT realiza todas las operaciones necesarias dependiendo de cuantas mariposas necesite una FFT de N puntos, el bloque Generador_Direciones manda las señales A1,A2 y AE a la memoria RAM y la señal que le indica a la LUT_Twiddle que factores utilizará en cada evaluación, el bloque maquina de estados FSM controla internamente toda el proceso de la evaluación.

Como se muestra en la Figura 2.11, la mariposa básica del algoritmo de la FFT consiste en una multiplicación compleja y dos sumas complejas. Se tiene que recordar que cada que se realiza una mariposa sobre un par de números a, b y que obtenemos A,B, no es necesario guardar el par a, b ya que no los volveremos a usar si no que ahora los datos necesarios para el siguiente cálculo son los A, B por esto podemos guardar en el mismo lugar del par a, b el par A, B.

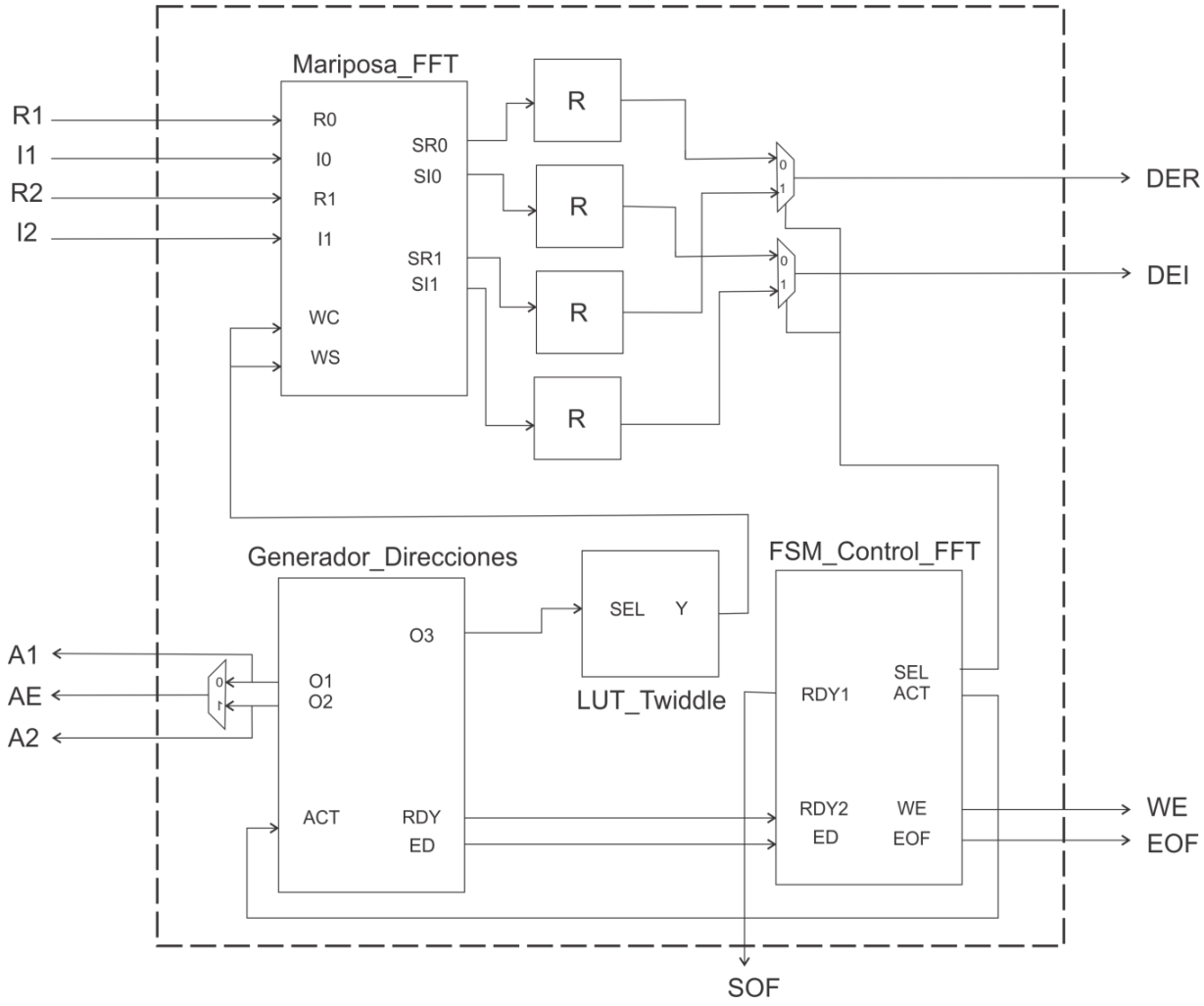

Figura 3.29 Arquitectura de la FFT

El diagrama de la Figura 3.30 muestra la arquitectura del bloque de la Mariposa_FFT de la Figura 3.29, esta requiere de multiplicadores y sumadores, como se muestra cada sumador tiene un ajuste en su salida de 36 bits a 18 bits para Ay B que en este caso son SR0, SI0 y SR1, SI1, esto se realiza para volver a ingresarlos en a y b los cuales se muestran en el diagrama como R0,I0 e R1, I1, respectivamente, de la misma manera se muestra un ajuste de las entradas de a que son R0 e I0, donde se ajusta de 18 bits a 36 bits para poder mandarlo a la entrada del sumador.

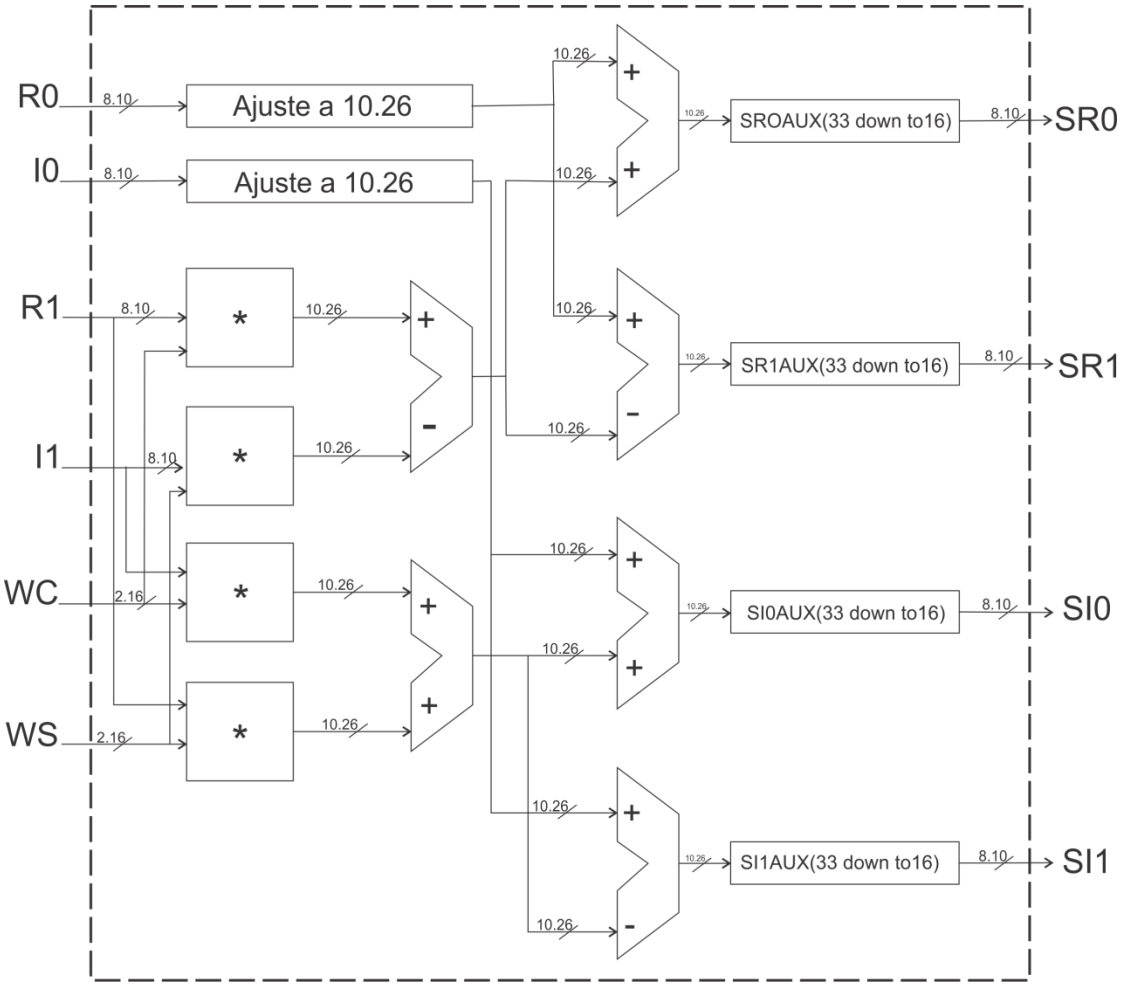

Figura 3.30 Mariposa_FFT

Todo el proceso que realiza el diagrama de la Figura 3.30, es en torno a los coeficientes a, b (R0, I0 y R1, I1) y a los factores twiddle W_N^{nk}(WC, WS). Se tiene:

$$\boldsymbol{a} = a_0 + jb_0 \; ; \quad \boldsymbol{b} = a_1 + jb_1 \; ; \boldsymbol{W_N^{nk}} = c + js \tag{69}$$

Y se efectúan las siguientes operaciones tomando en cuenta los valores de la ecuación (69):

$$\boldsymbol{A} = a_0 + jb_0 + [a_1c + ja_1s + jb_1c - b_1s] \tag{70}$$

$$\boldsymbol{A} = [a_0 + (a_1 c - b_1 s) + j\,[b_0 + (a_1 s + b_1 c)] \quad (71)$$

$$\boldsymbol{B} = a_0 + jb_0 + [a_1 c + ja_1 s + jb_1 c - b_1 s] \quad (72)$$

$$\boldsymbol{B} = [a_0 - (a_1 c - b_1 s) + j\,[b_0 + (a_1 s + b_1 c)] \quad (73)$$

Estas operaciones se realizan con ayuda de cada uno de los bloques expuestos en la Figura 3.30, en donde entran los sumadores, multiplicadores, que son algunos de los bloques utilizados en la implementación de la Aproximación Polinomial de la Serie de Taylor.

Una parte muy importante de esta implementación es el generar las direcciones de los operadores y los valores de los coeficientes twiddle que intervienen en las operaciones de cada una de las mariposas del algoritmo FFT. El motivo radica en que sólo se reserva un único espacio de memoria, donde su capacidad viene dada por los N puntos de la FFT que se desee resolver. El espacio contemplado se deberá usar tanto para almacenar los valores iniciales de cada una de las mariposas así como de guardar sus resultados. Un manejo incorrecto de las direcciones en cualquier proceso producirá una resolución parcial o totalmente errónea del algoritmo, en cualquier caso que se suscite, esto no es deseable.

A manera de ejemplificación, se utilizó una FFT de N = 8 puntos la cual se muestra en la Figura 3.31, para describir el diseño y desarrollo de la metodología para la generación correcta de las direcciones de los operadores y factores twiddle.

La resolución de una FFT de N puntos necesita log2(N) etapas, cada una de ellas con N operaciones en total. En base al bloque Mariposa_FFT de la Figura 3.30, el número de operaciones en las etapas se reduce a N/2 ya que este bloque es capaz de tomar un par de operadores y proporcionar sus resultados a la vez.

Por tanto, para la solución al ejemplo de la Figura 3.31 se necesitan 3 etapas con 4 operaciones cada una de ellas. El orden, las direcciones de los operadores y los factores twiddle de las operaciones se muestra en las Tablas 3.12, 3.13 y 3.14.

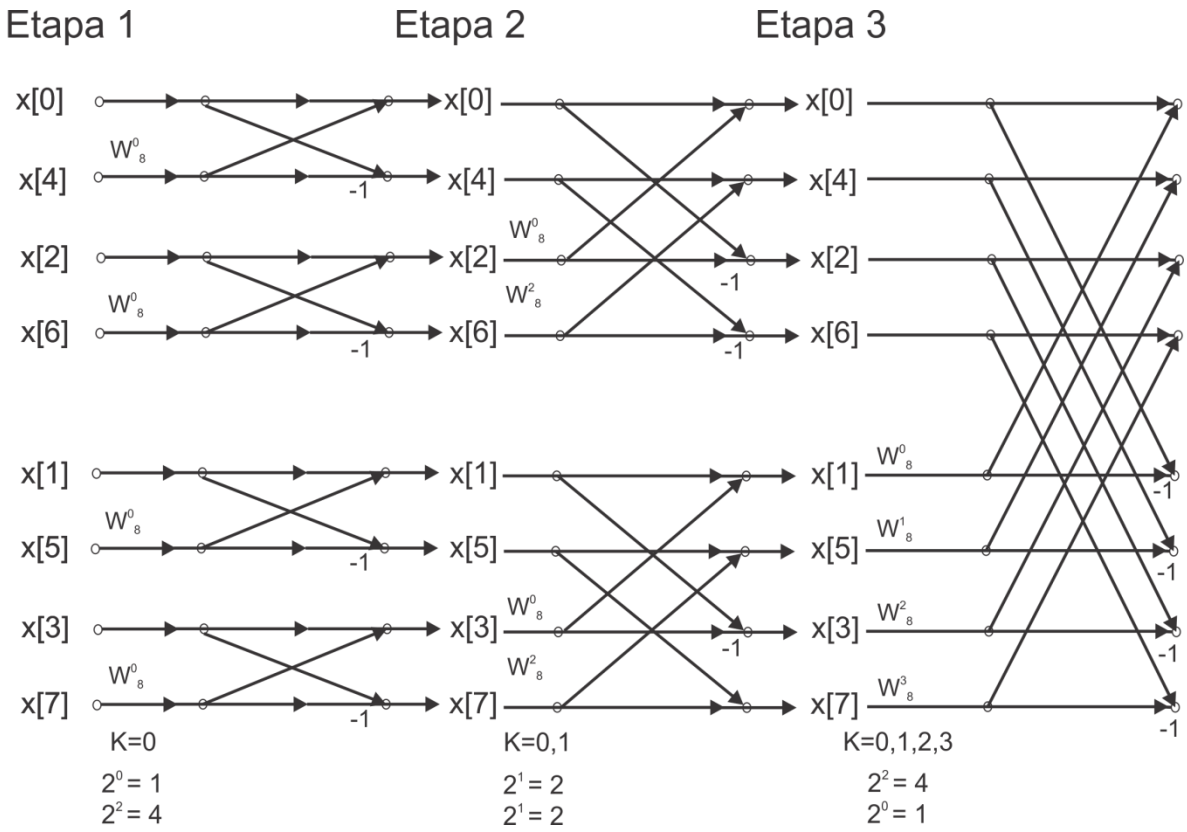

Figura 3.31 FFT de N=8 puntos para diseño de la metodología de la generación de direcciones.

Tabla 3.12 Orden Etapa1de la FFT N=8.

No. Operación	Par Direcciones Operadores	Factor Twiddle
1	0 – 4	W_8^0
2	2 – 6	W_8^0
3	1 – 5	W_8^0
4	3 – 7	W_8^0

Tabla 3.13 Orden Etapa2 de la FFT N=8.

No. Operación	Par Direcciones Operadores	Factor Twiddle
1	0 – 2	W_8^0
2	4 – 6	W_8^2
3	1 – 3	W_8^0
4	5 – 7	W_8^2

Tabla 3.14 Orden Etapa3 de la FFT N=8.

No. Operación	Par Direcciones Operadores	Factor Twiddle
1	0 – 1	W_8^0
2	4 – 5	W_8^1
3	2 – 3	W_8^2
4	6 – 7	W_8^3

A partir del orden de direccionamiento empleado en la etapa 1, se puedo adquirir el direccionamiento necesario para cada etapa posterior. Cada una de las direcciones de la etapa 1 se almacenan como datos (no como direcciones como tal) en un nuevo espacio de memoria fijo con la intención de efectuar un direccionamiento indirecto y así obtener las direcciones de las etapas siguientes. Un direccionamiento indirecto toma un dato de un espacio de memoria para convertirlo en una nueva dirección en el mismo o en cualquier otro medio de almacenamiento. El bloque de memoria para las direcciones de la etapa 1 se muestra en la Tabla 3.15.

Tabla 3.15 Bloque de memoria fijo para la Etapa 1.

Dirección	0	1	2	3	4	5	6	7
Dato	0	4	2	6	1	5	3	7

Se emplea un índice para desplazarse a través del espacio de memoria de la Tabla 3.15. Con la ayuda de éste se obtiene el par de direcciones correspondiente a las Tablas 3.13 y 3.14. La primera dirección se obtiene a partir del dato almacenado en la posición actual del índice. La segunda se adquiere sumando al índice una cantidad específica dependiente de la etapa en que se encuentra, donde ésta es igual a:

$$l = 2^{E-1}, donde\ E\ es\ el\ número\ de\ etapa \tag{74}$$

El índice se posiciona en la dirección 0 al comienzo de cada una de las etapas. Se mantiene un control de la cantidad de pares de direcciones que se han generado. Se obtiene el primer par de direcciones en base a lo establecido anteriormente. Posteriormente, la cantidad de par de direcciones generadas se incrementa en 1, el índice se desplaza una posición y se procede a efectuar la siguiente evaluación:

- Sí la cantidad de pares de direcciones generados es igual a l; el índice se desplaza l posiciones, el número de par de direcciones generadas se actualiza a 0 y se obtienen nuevas direcciones de ser necesario.

- De lo contrario; se obtiene otro nuevo par de direcciones a partir de la nueva posición del índice, la cantidad de par de direcciones generadas se incrementa en 1, el índice se desplaza una posición y se realiza una nueva evaluación.

Con el algoritmo descrito anteriormente, el direccionamiento que se muestra en la Tabla 3.13 se obtiene de la siguiente manera:

1. La posición inicial del índice apunta a la dirección 0 de la tabla 3.13. El primer par de direcciones son los datos en la posición 0 y 2, por tanto, el par de direcciones es 0 – 2. Se desplaza el índice a la dirección 1 y el contador de par de direcciones generadas es igual a 1.
2. La cantidad de par de direcciones generadas no es igual a *l* = 2, por tanto, el segundo par de direcciones son los datos en la posición 1 y 3 de la Tabla 3.15 cuyos datos son 4 – 6. Se desplaza el índice a la dirección 2, el contador de par de direcciones generadas es igual a 2.
3. La cantidad de par de direcciones generadas es igual a *l*=2. El índice se desplaza a la dirección 4 y el contador de par de direcciones generadas se actualiza a 0.
4. Aún son necesarios pares de direcciones, se obtienen los necesarios.
5. El siguiente par de direcciones son los datos en la posición 4 y 6 con valores de 1 – 3. Se desplaza el índice a la dirección 5 y el contador de par de direcciones es igual a 1.
6. La cantidad de par de direcciones generadas no es igual a *l* = 2. El nuevo par de direcciones son los datos en la posición 5 y 7 que son 5 – 7. Se desplaza el índice a la dirección 6, el contador de par de direcciones generadas es igual a 2.
7. La cantidad de par de direcciones generadas es igual a *l*=2. El índice se desplaza a la posición 7 y el contador de direcciones generadas se actualiza a 0.
8. No son necesarios más pares de direcciones. Se inicia un proceso similar para la etapa siguiente.

Cualquier FFT de N puntos necesita l factores twiddle correspondientes a cada una de sus etapas, enumerados k = 0, 1, 2,…, l – 1. El k-ésimo factor twiddle es igual a:

$$\mathrm{m} = \mathrm{k} * 2^{\log_2(\mathrm{N})-\mathrm{E}}, \text{donde E es el número de etapa} \quad (75)$$

Tomando en cuenta el contador de par de direcciones generadas en el algoritmo anterior, los valores de la sucesión k se obtienen a partir de éste.

Los valores que multiplican a k dependiendo de la etapa en que se encuentre siempre serán potencias de 2 estos se muestran en la Tabla 3.16, por tanto, se almacenan los datos pre-calculados en un espacio de memoria fijo.

Tabla 3.16 Datos Pre-calculador en espacio de memoria fijo.

Etapas	Etapa 1	Etapa 2	Etapa 3
Dirección	0	1	2
Dato	4	2	1

De nueva cuenta, se utiliza un direccionamiento indirecto para obtener los valores correspondientes a cada coeficiente twiddle. Estos son pre-calculados y almacenados en un espacio de memoria fijo los cuales se muestran en la Tabla 3.17. El valor de m será la dirección en este espacio de memoria.

Tabla 3.17 Coeficientes twiddle pre-calculados en espacio de memoria fijo.

Dirección	0	1	2	3
Dato	W_4^0	W_4^1	W_4^2	W_4^3

Con lo descrito anteriormente, la Figura 3.32 define la arquitectura interna del bloque Generador_Direcciones.

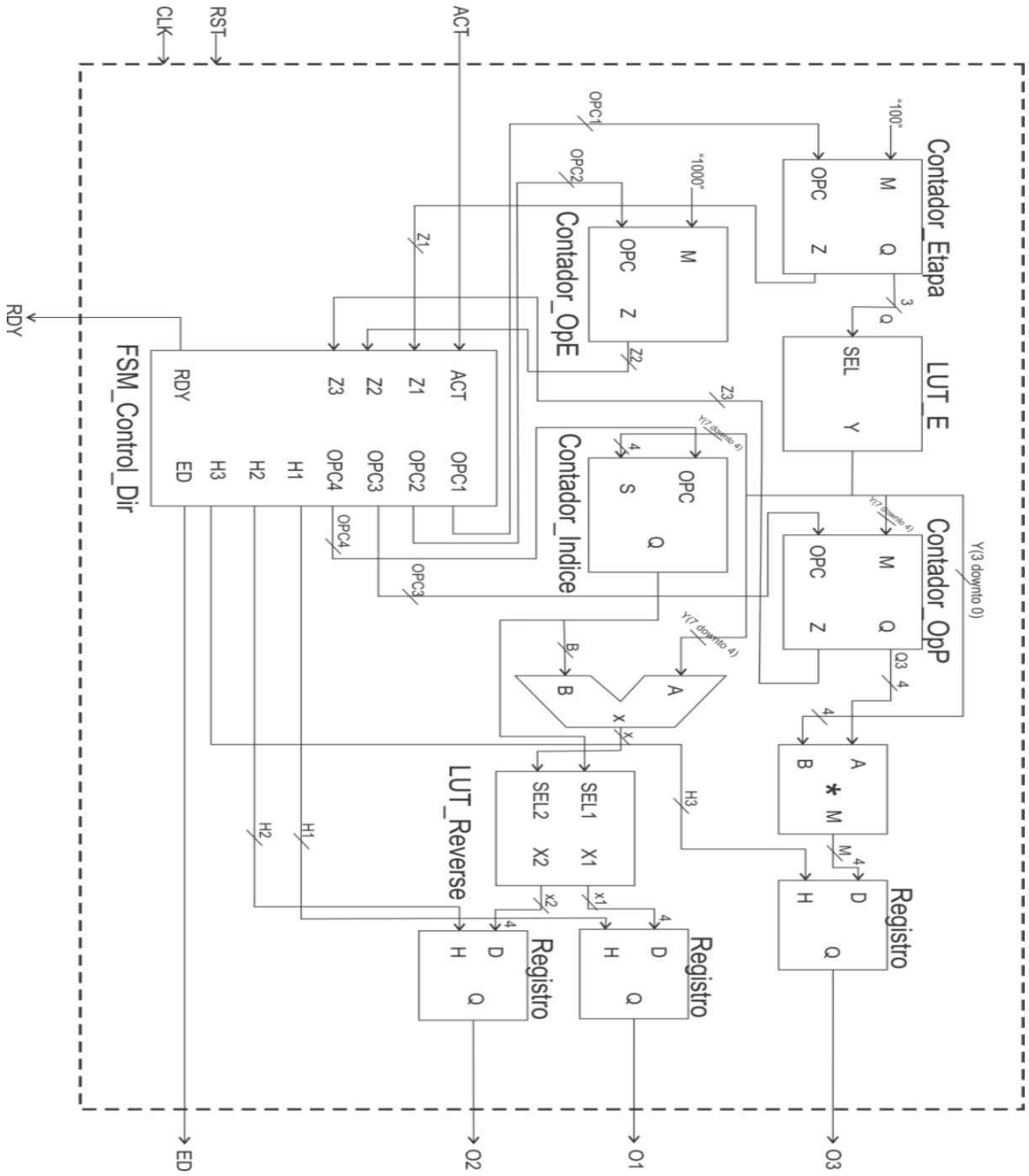

Figura 3.32 Arquitectura del Generador_Direcciones

La arquitectura del generador de direcciones que se muestra en la Figura 3.32, donde tiene como entradas el RST con el que se resetea en cualquier momento el generador, el CLK que es el reloj maestro, ACT que es la señal de activación para que este comience a funcionar, O3 que es la señal que se manda a la LUT_Twiddle para que esta envié a la mariposa los factores correctos y correspondientes, O1 y O2 indican a la memoria cual es el orden en que tiene que enviar los datos a la Mariposa_FFT, ED manda la señal que indica cuando esta lista una dirección y por ultimo RDY que indica cuando todas las direcciones han sido enviadas a la Mariposa_FFT. Este bloque contiene, tablas de consulta (LUTs), contadores, Registros, Sumadores, multiplicadores y una maquina de control que será la encargada de dar un buen funcionamiento interno.

El bloque Contador_OpE es empleado para llevar la cuenta de las operaciones realizadas en cada una de las etapas de la FFT que se desea resolver. Este bloque es necesario para saber en que instante se incrementar la cuenta del bloque Contador _Etapa.

El bloque Contador_Etapa lleva la cuenta y el control de las etapas de la FFT que se desea resolver. Su salida Q representa el número de etapa en la que se opera y será el selector del bloque LUT_E.

El bloque LUT_E es un multiplexor cuya salida es la combinación de los valores de *I* y la Tabla 3.16 correspondientes a cada etapa.

El bloque Contador_Indice es el encargado de llevar el control del desplazamiento y posicionamiento del índice. Su salida en conjunto con la del bloque LUT_E obtuvo el par de direcciones de los operadores de las Tablas 3.12, 3.13, 3.14.

El bloque Contador_OpP lleva la cuenta de los pares de direcciones generados. Su salida en conjunto con la del bloque LUT_E dio la dirección de los coeficientes twiddle de la Tabla 3.17.

El bloque LUT_Reverse es la implementación del espacio de memoria fijo de la Tabla 3.15.

El bloque FSM_Control_Dir es el encargado de controlar el algoritmo que permite generar de manera eficiente cada una de las direcciones necesarias en la resolución de la FFT involucrada.

Cada uno de los bloques Registro son empleados para almacenar las direcciones de los operadores y la direcciones de los factores twiddle durante el tiempo requerido para efectuar las operaciones correspondientes.

El bloque FSM_Control _FFT es la encargada de controlar la generación del direccionamiento requerido y el almacenamiento de los resultados derivados de las operaciones efectuadas.

El bloque LUT_Twiddle contiene las direcciones y los datos de los valores de los coeficientes twiddle de la Tabla 3.17.

Los bloques Registro son empleados para almacenar, de manera temporal, los resultados de las operaciones efectuadas el tiempo necesario para su correcto almacenamiento.

3.4 Desarrollo de la implementación utilizando Matlab

Se genero un programa con el software MatLab el cual genera los archivos necesarios para cualquier FFT de N puntos, automáticamente se crean las tablas de consulta (LUT) que se utilizan en el modulo de la Figura 3.32 la LUT_reverse, LUT_Etapa y la LUT_twiddle de la Figura 3. 29, estas LUTS se generaran dependiendo de que cuantos N puntos se requiera para la FFT deseada.

Gracias a la versatilidad de MatLab, una de sus ventajas es que se puede pasar el código generado en este a ciertos lenguajes de programación como por ejemplo, en este caso podemos generar directamente desde MatLab la extensión del archivo .vhd, que son los que se utilizan en el lenguaje de Hardware VHDL.

El diagrama de flujo que se muestra en la Figura 3.33 contiene los pasos a seguir para poder generar de manera adecuada el archivo *.vhd para las FFTs de N puntos, donde automáticamente dependiendo del valor de N se modifica y se ajusta, de igual forma la entidad general del bloque Generador_Direcciones de la Figura 3.32, y por ultimo la entidad general del proyecto de la FFT de la Figura 3.28.

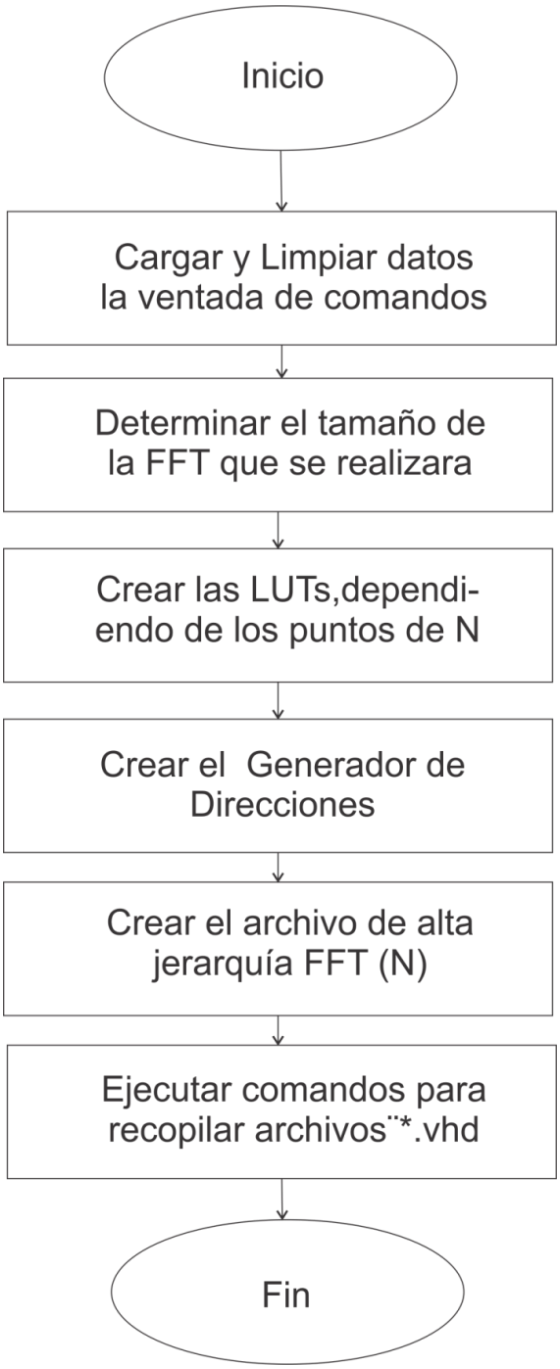

Figura 3.33 Diagrama de Flujo general para una FFT de N puntos.

En la Figura 3.34 se muestra el diagrama de flujo con los pasos con los cuales se genera la LUT_TWIDDLE desde el software MatLab.

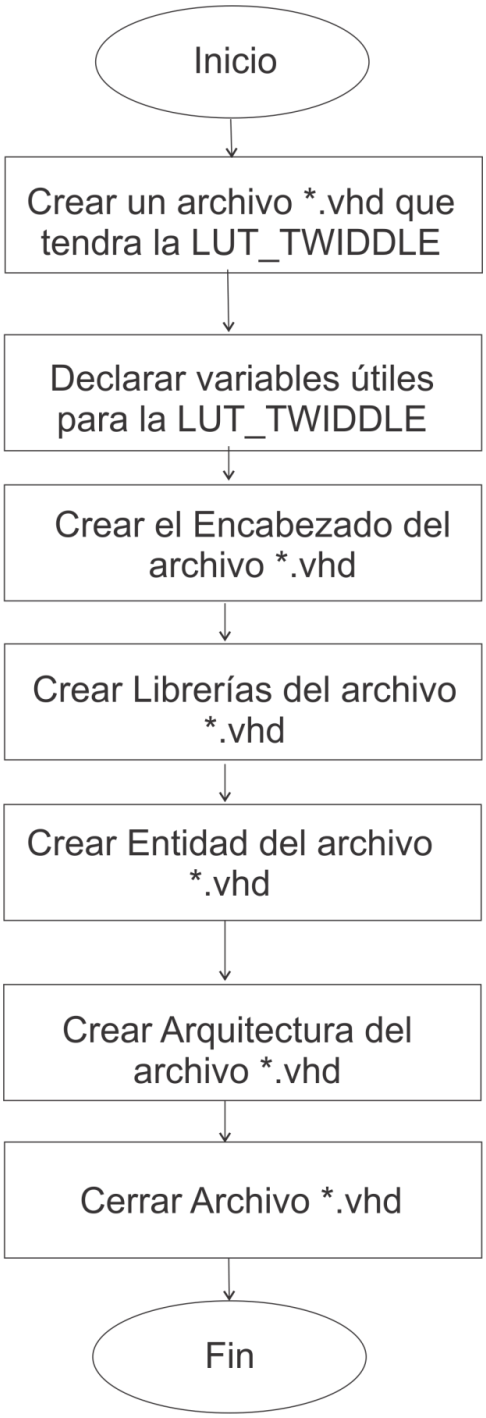

Figura 3.34 Diagrama de Flujo para generar la LUT_TWIDDLE.

En la Figura 3.35 se muestra el diagrama de flujo que genera la LUT_ETAPA.

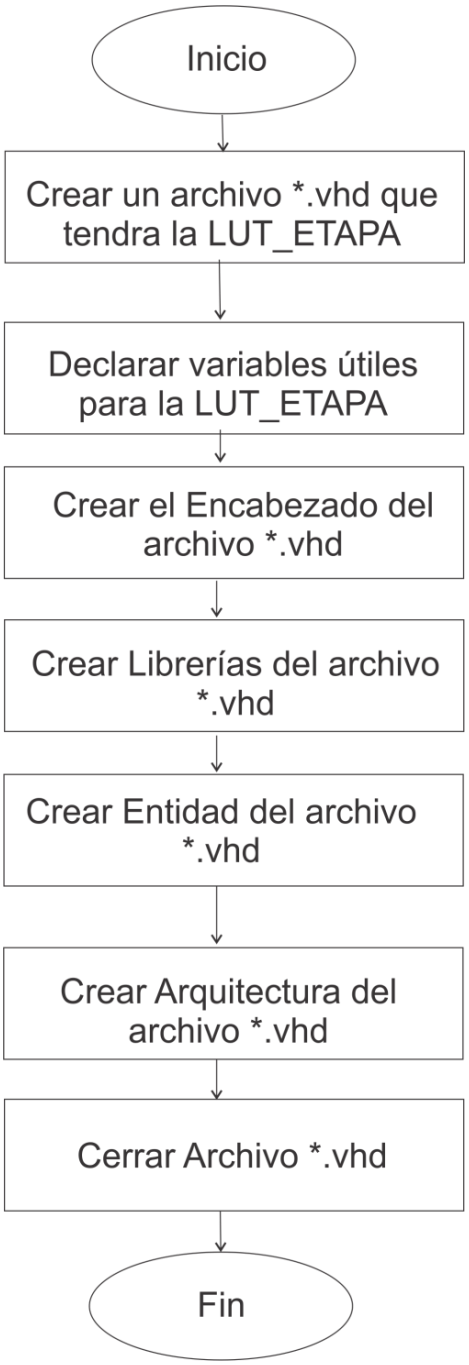

Figura 3.35 Diagrama de Flujo para generar la LUT_ETAPA.

En la Figura 3.36 se muestra el diagrama de flujo, con el cual se genera la LUT_REVERSE (N).

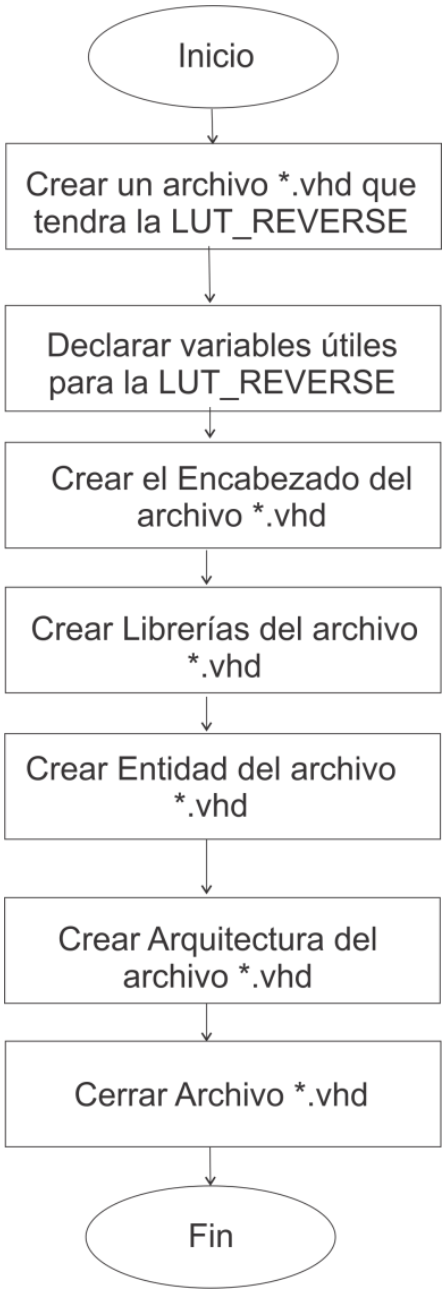

Figura 3.36 Diagrama de Flujo para generar la LUT_REVERSE.

En la Figura 3.37 se muestra el diagrama de flujo, con el cual se genera el archivo del Generado de direcciones.

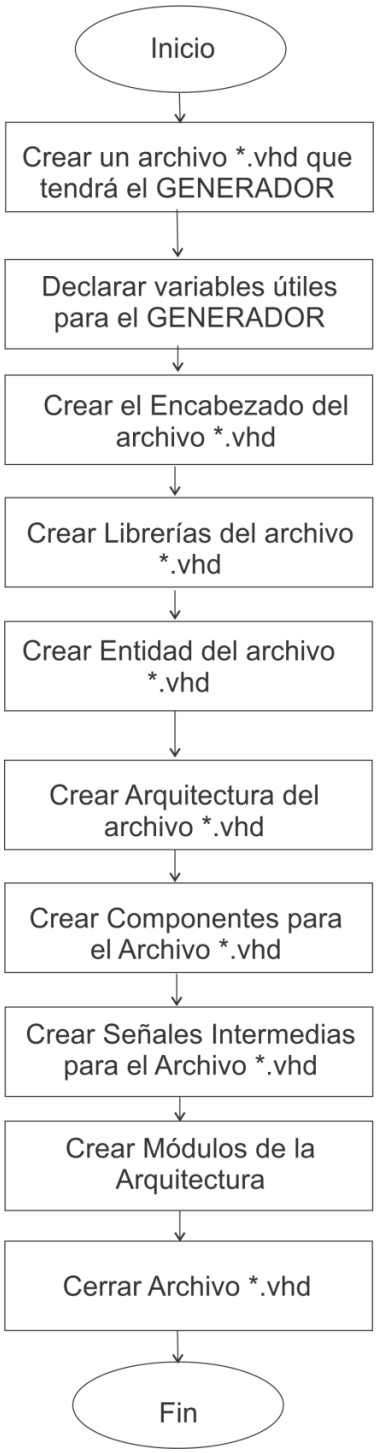

Figura 3.37 Diagrama de Flujo para el GENERADOR_DIRECCIONES.

Así mismo en la Figura 3.38 se muestra el diagrama de flujo que se utilizó para generar el Módulo de alta jerarquía que es el de la FFT_RADIX(N).

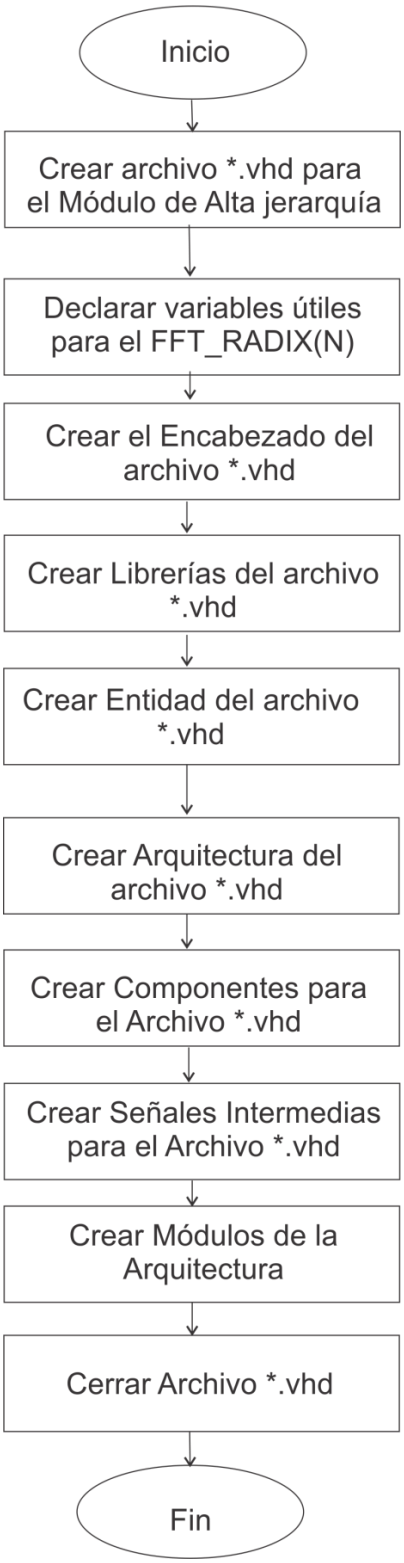

Figura 3.38 Diagrama de Flujo para el FFT_RADIX(N).

CAPÍTULO IV

Pruebas y Resultados

4.1 Pruebas y Resultados de la Aproximación polinomial de la Serie de Taylor Función Básica (Seno)

Una vez que se obtiene la grafica de la aproximación polinomial de la función Seno con la “Rutina Matlab para diseño de una aproximación polinomial del tipo sigmoide” que se menciona en el Capitulo 3, la Figura 4.39 muestra el resultado que se tiene que obtener al implementar la aproximación polinomial de la Serie de Taylor Función Básica (Seno) en Hardware.

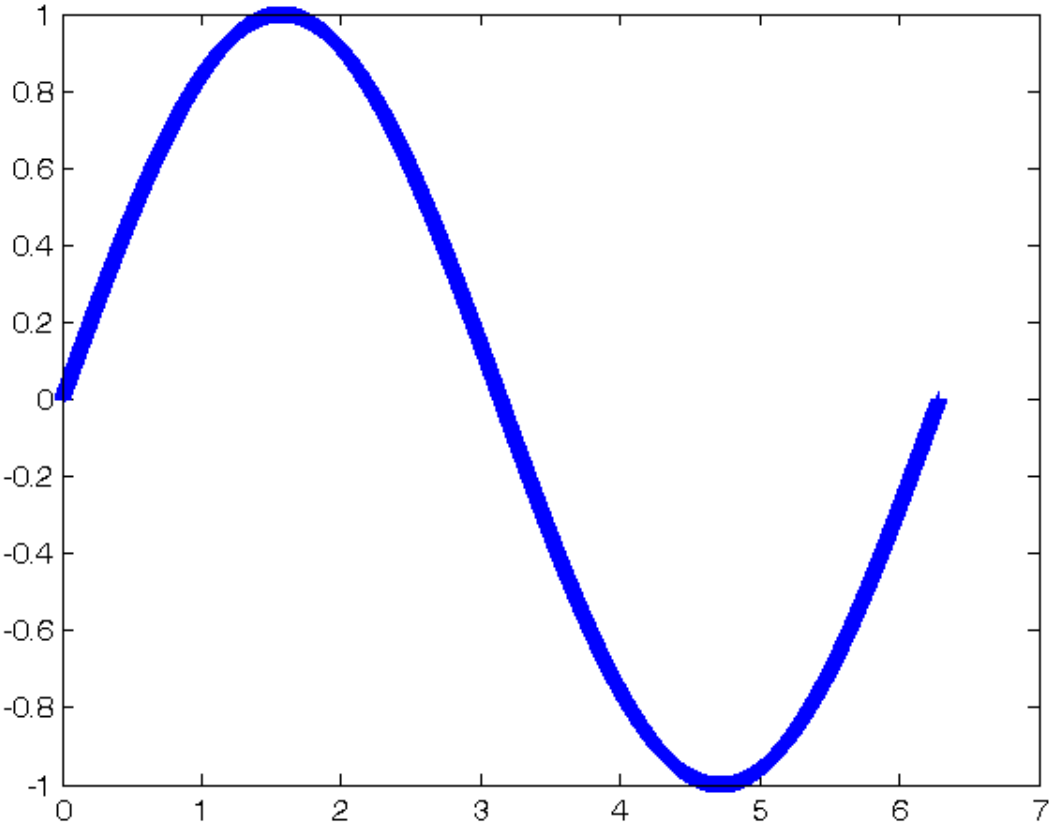

Figura 4.39 Resultado de la Rutina de MatLab, para una aproximación sigmoide.

Teniendo este resultado de la prueba realizada en Matlab, se realiza la implementación en hardware y su simulación para ver su comportamiento y compararlo con el resultado obtenido de la Figura 4.39.

4.1.1 Prueba de la Simulación para la Aproximación polinomial implementada en Hardware (FPGA)

En la Figura 4.40 se muestra el resultado de la simulación en Active-HDL de la función Seno, en esta parte se verifica que realmente la implementación esta realizando la evaluación correcta de la aproximación y de esta manera una vez verificada se pasa a la implemetación en el dispositivo programable FPGA.

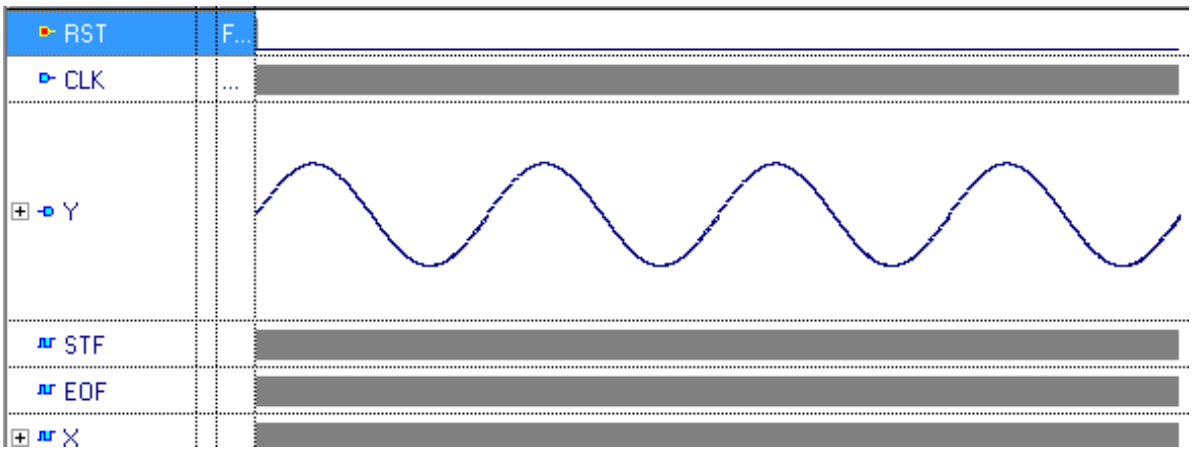

Figura 4.40 Simulación en Active-HDL , para una aproximación sigmoide.

Tomando en cuenta que la simulación es la esperada y tiene similitud con la prueba realizada en software que se muestra en la Figura 4.39. Se implementa para poder comprobar los resultados ya en el dispositivo programable FPGA.

4.1.2 Resultados de la Aproximación polinomial implementada en Hardware (FPGA) mostrada en un Osciloscopio Digital.

Una vez implementada la aproximación polinomian en un FPGA, con ayuda de un Osciloscopio Digital podremos verificar que efectivamente el resultado es el correcto, y como se muestra en la Figura 4.41 se ve que efectivamente la simulación que se realizo en Active-HDL concuerda con el resultado obtenido del Osciloscopio Digital.

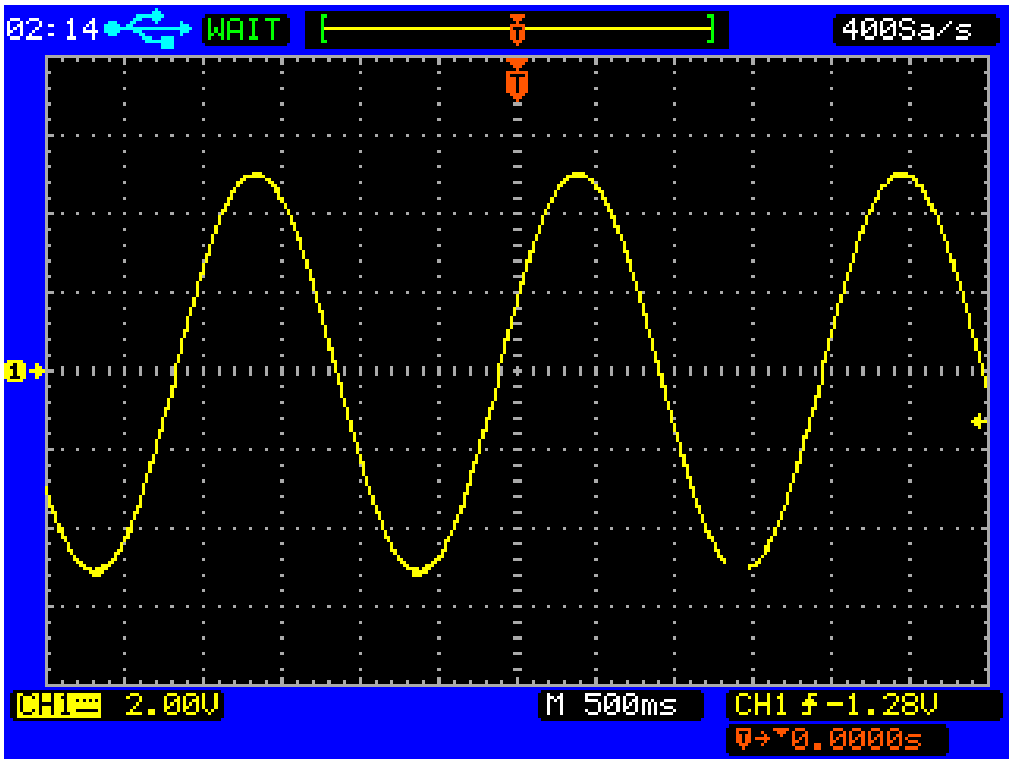

Figura 4.41 Evaluación de la aproximación polinomial, comprobada en un Osciloscopio Digital

4.2 Pruebas y Resultados de la Transformada Rápida de Fourier (FFT)

Al tener la implementación de la FFT en Hardware, que en este caso se utilizó el dispositivo programable de Digilent, un FPGA Nexys 2 Spartam 3E-500 FG320, la cual se muestra en la Figura 2.1, para comprobar los resultados, se utilizó una Onda Senoidal con una Frecuencia de 70Hz, tomando 256 muestras y con una Frecuencia de muestreo de 210 muestras/segundo, la cual se muestra en la Figura 4.42.

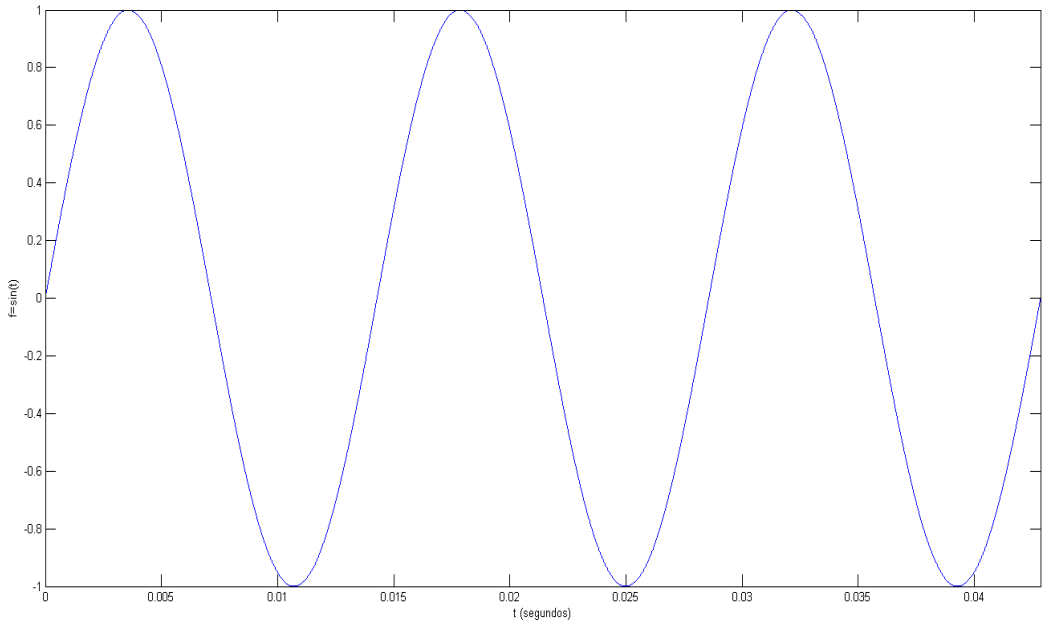

Figura 4.42 Onda Senoidal F=70Hz.

Con esta Onda Senoidal, se realizó una prueba con una FFT de N = 256 puntos, tanto en Software como en Hardware, esto para comprobar que la implementación en Hardware trabaja igual o mejor que la evaluada en el Software de MatLab; con la función que MatLab tiene establecida para realizar una FFT, en las siguientes secciones se muestran tanto las pruebas y los resultados que se obtuvieron de ambas evaluaciones.

Como se muestra en la Figura 4.43 existe una pequeña diferencia en los datos de entrada al Hardware ya que son enviadas mediante la comunicación serial y pueden llegar a tener algunos ajustes los cuales pueden llegar a alterar un poco la parte decimal de nuestros datos, es por esto que se puede tener una pequeña diferencia en los valores del resultado. En la Figura 4.44 se muestra el valor absoluto de la diferencia de los datos de entrada.

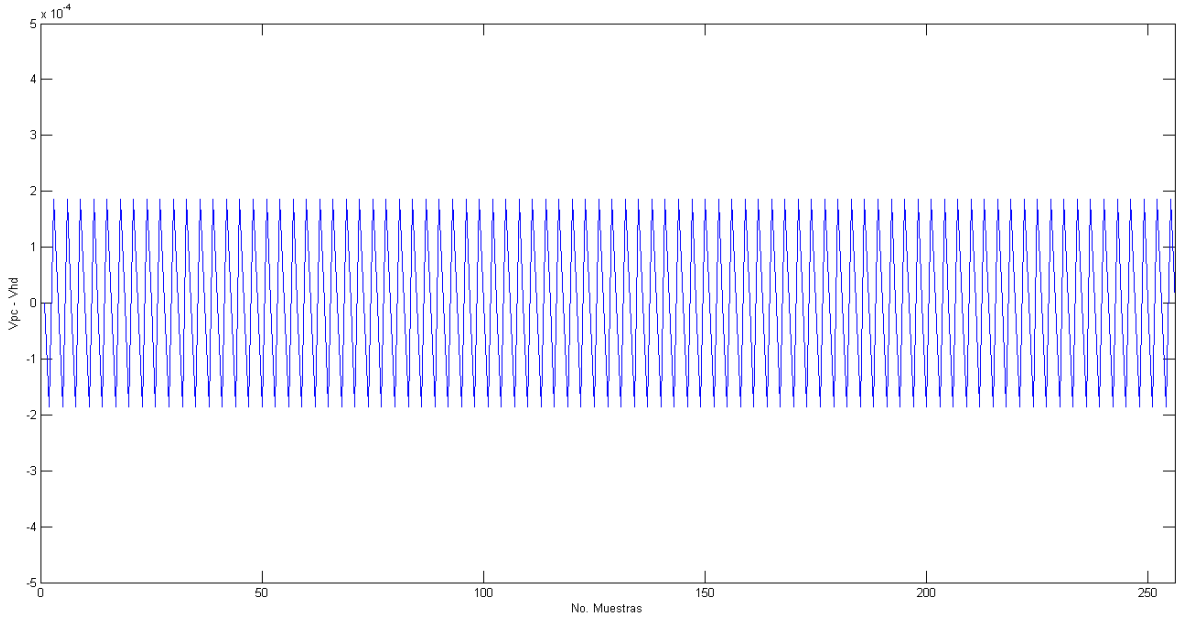

Figura 4.43 Diferencia entre datos de entrada PC – Hardware.

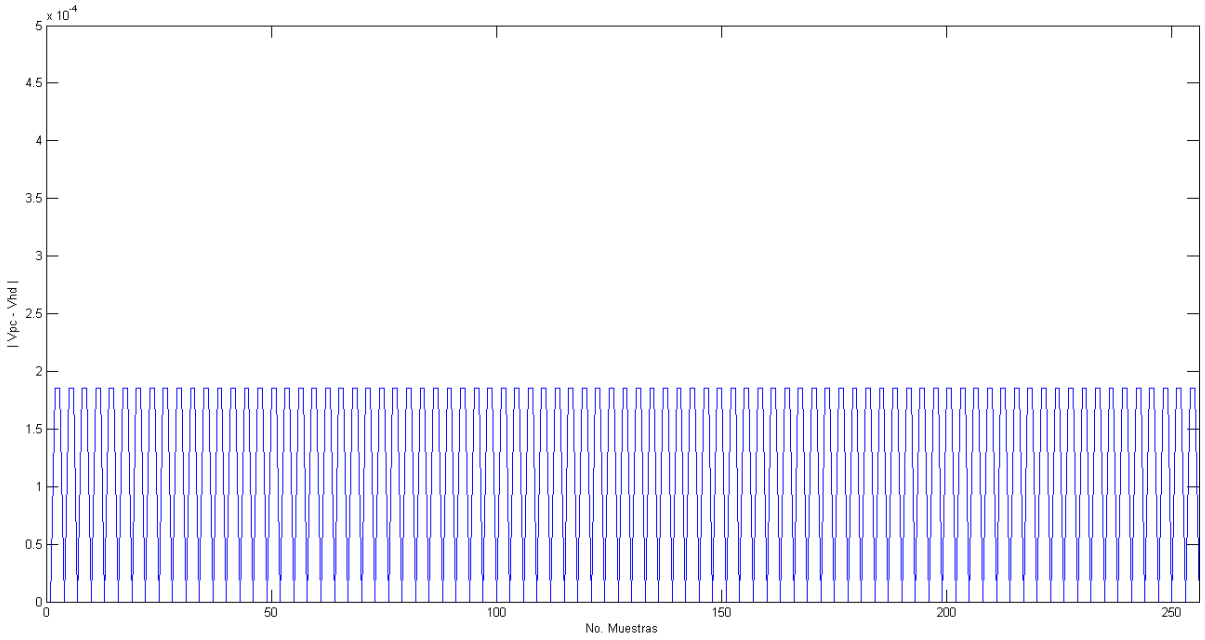

Figura 4.44 Diferencia absoluta entre datos de entrada PC – Hardware.

4.2.1 Pruebas y Resultados de la implementación de la FFT en MatLab.

Evaluando la Onda Senoidal mostrada la Figura 4.42 se obtuvo la Figura 4.45 donde se muestra la grafica del resultado de una FFT de N = 256 puntos, evaluada desde la PC con la función ya establecida en el Software MatLab.

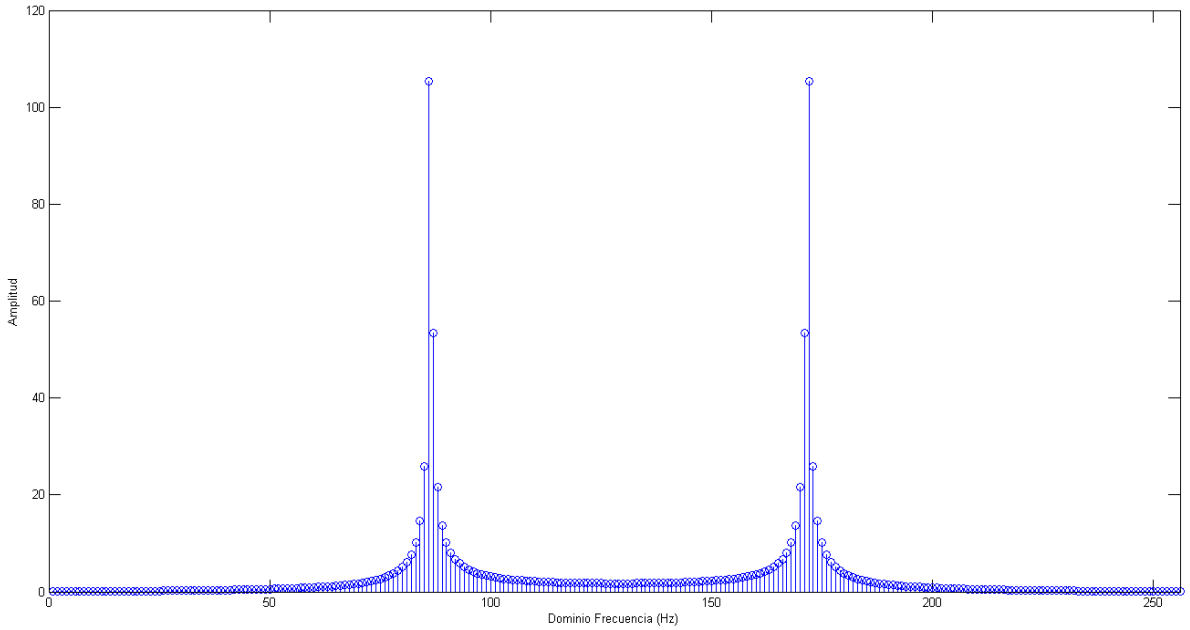

Figura 4.45 FFT 256 puntos Onda Senoidal F=70Hz, evaluada en Software.

4.2.2 Pruebas y Resultados de la implementación de la FFT en Hardware.

Una vez implementado en el FPGA el algoritmo de la FFT, se enviaron los datos de la Onda Senoidal de la Figura 4.42 mediante la comunicación serial y el resultado que se obtuvo se envío nuevamente a la PC para poder graficar su resultado mediante el software MatLab y como se muestra en la Figura 4.46 se obtuvo la grafica de la FFT evaluada en Hardware de N = 256 puntos similar a la grafica anterior.

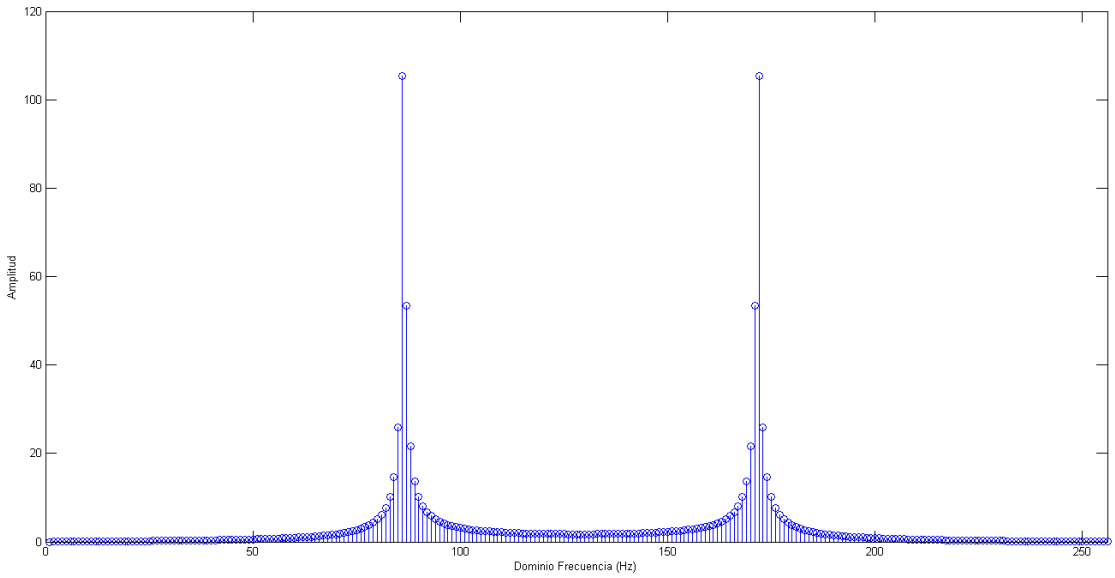

Figura 4.46 FFT 256 puntos Onda Senoidal F=70Hz, evaluada en Hardware.

Comparando las Figuras 4.47 y 4.48 se pudo ver que dio el mismo resultado tanto en Software como en Hardware; a pesar de esto existe una pequeña diferencia que a simple vista no se puede ver ya que es minima, de tal manera que para que se pudiera apreciar se grafico esta diferencia como se muestra en las Figuras 4.47 y 4.48 donde la Figura 4.47 muestra la diferencia real y en la Figura 4.48 se muestra la diferencia absoluta de esta evaluación, como se pudo ver la diferencia es menor a cero y diferencia mayor que resulto es de caso 0.03 en valor absoluto.

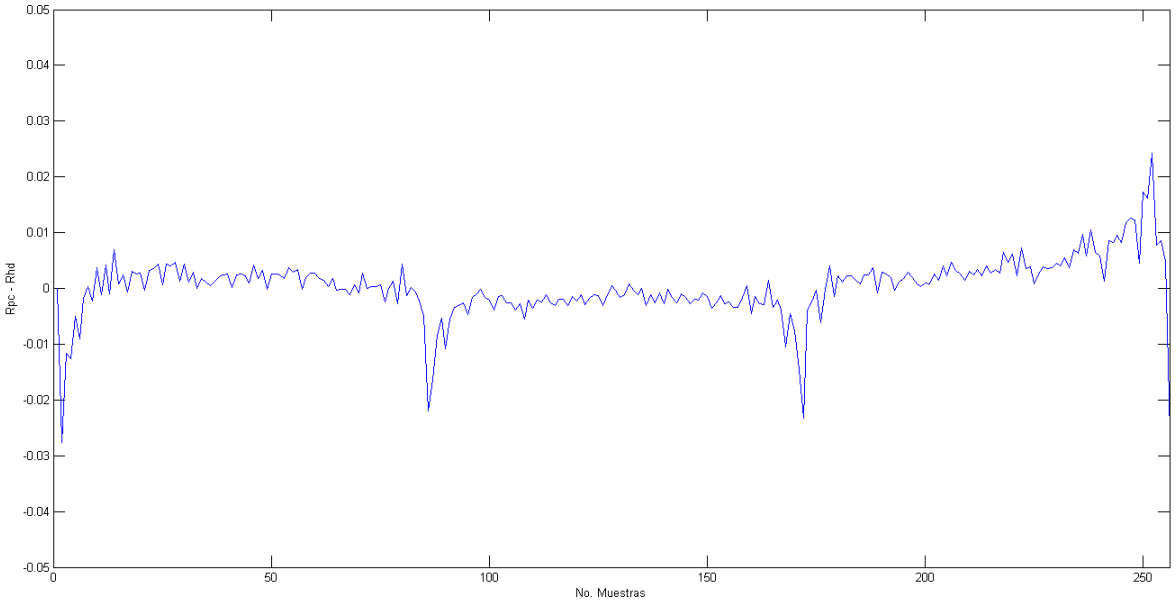

Figura 4.47 Diferencia Resultados PC – Hardware.

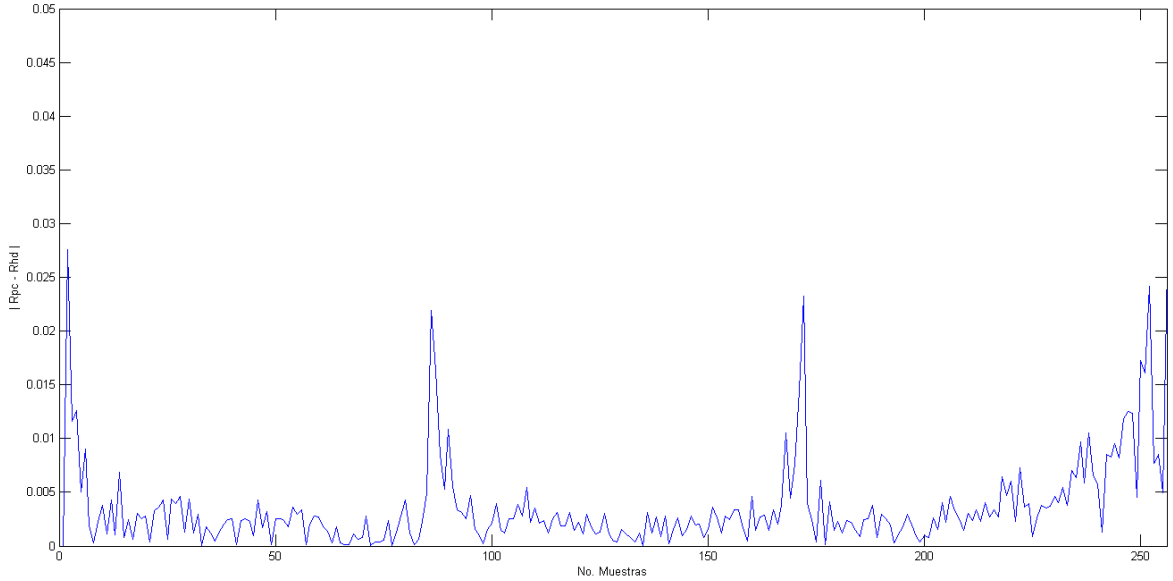

Figura 4.48 Diferencia Absoluta Resultados PC – Hardware.

CAPÍTULO V

Conclusión y Referencias

5.1 Conclusión

En esta Tesis se presentó la implementación de algunos algoritmos de métodos numéricos, como son Series de Taylor, Transformada de Fourier, entre otros. Se implementó la Transformada Rápida de Fourier (FFT) la cual permite una resolución más rápida a una Transformada Discreta de Fourier (DFT), la FFT realiza la mitad de las operaciones a diferencia de la (DFT). Al estudiar e implementar estos algoritmos fue con la finalidad de que nos ayudara en la solución más exacta y rápida para resolver ecuaciones matemáticas y poder realizar análisis en señales digitales a lo que se le llama procesamiento digital de señales, así como también se pudieron encontrar algunas ventajas de implementar estos métodos numéricos en hardware a diferencia de una PC convencional, por ejemplo una de las ventajas de implementar en Hardware es que se pudo realizar en tiempo real el análisis de ciertas señales, porque permiten trabajar de forma paralela, como en el caso de la FFT que se puedo realizar todas las operaciones implementadas al mismo tiempo a diferencia de la PC que puede tener cierto retraso en la evaluación. De esta manera se pudo encontrar que al implementar en hardware hace que podamos tener un resultado en menor tiempo que el que se obtiene en la PC convencional.

Gracias a este trabajo se pudo comprobar que los métodos numéricos nos ayudan a la aproximación de soluciones de ecuaciones que pueden tener gran cantidad de operaciones y variables, una vez que ya implementamos los algoritmos en este trabajo de tesis, se puede decir que tienen gran confiabilidad para poder proporcionar resultados correctos (con un margen de error, el cual es mínimo).

De los aprendizajes que obtuve durante la realización de este trabajo, fue el poder conocer un poco más a fondo la programación en hardware como también encontrar ciertas herramientas en programación de cómputo científico que permitieron sintetizar un poco más el trabajo realizado.

TRABAJO A FUTURO

Respecto al trabajo futuro existen diferente puntos sobre los que se pueden trabajar, uno de ellos seria aplicar estos algoritmos implementados en Hardware para realizar el procesamiento digital de diversas señales, probar diversos filtros, incluso poder aplicarlos al procesamiento de imágenes, ya sea para la reconstrucción, como para analizar de una forma más rápida comparado con el procesamiento en una Pc convencional, se pueden aplicar en la resolución de operaciones que contengan varias funciones trigonométricas, evaluadas en tiempo real. También se podrían aplicar para trabajar en el procesamiento digital de voz, audio, video y datos.

5.2 Referencias

Alonso, A. B. (2010). *Diseño sobre FPGA de una unidad Aritmética Decimal.* Director Jean-Pierre Deschamps, Universitat Rovira Virgili.

Bastidas*, La Cruz William (2002). Series de Potencias.*

Bonafonte, A. (2009). *Transformada de Fourier, Señales y Sistemas I.* Catalunya: Universidad Politécnica de Catalunya.

Cáceres, J. P. (2007). *Transformada de Fourier.*Stanford University.

Castillo-Atoche A., P.-C. M. (2006). *Implementación de la FFT en Hologramas de Fourier generados con FPGAs.* Ingenieria: Revista Académica de la FI-UADY.

Chapra, S. C. (5ta edición. 2007). *Métodos numéricos para ingenieros.* México: McGraw-Hill.

Christhofer, C. G. (s.f.). *Integración Numérica con Redes Neuronales.* H. Puebla de Zaragoza: Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación.

Cordoba, U. N. (2008). *Procesamiento Digital de Señales, Transformada de Fourier y el algoritmo FFT.* LabDSP(Laboratorio de Prcesamiento Digital de Señales).

Corporation, A. (1995-2012). *Altera*. Recuperado el 19 de Septiembre de 2012, de http://www.altera.com/

Cruz, A. S. (2011). *Importancia de los Métodos Numéricos en la Simulación de Procesos Químicos.* Aplicada, Matemática Superior, UTN-FRRo.

Domínguez, V. H. (2009). *Procesamiento de Imágenes de Mastografía con Implementación en Hardware.* Puebla: Benemérita Universidad Autónoma de Puebla.

Douglas L. Jones. (2006). *Decimation in time Radix-2 FFT.*

Francés, M. J. (2011). *Implementación de Métodos Numéricos para el análisis electromagnético de medios periódicos "Aplicación en longitudes de onda ópticas y optimización computacional".* Alicnate: Universidad de Alicante, Dpto. de Física, Ingeniería de Sistemas y Teoría de la Señal.

Gómez, F. (s.f.). *Procesamiento Digital de Señales, Análisis de Fourier en tiempo discreto.* Mexíco: UAM, Ingeniería Informática.

Guerra, I. J. (2006). Importancia de los Métodos Numéricos y la Teoría de Errores. Instituto Tecnológico de Nuevo León.

Guerrero Martha, S. V. (2008). Aplicación del método de elemento finito al análisis nodal. *Facultad de Ing. Mécanica y Eléctrica, UANL.*

Hidalgo, U. M. (2003). Modelado y Análisis de Sistemas Eléctricos bajo Condiciones de Operación no Senoidales. Michoacan: Facultad de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo

Ing. Alexander Quintero M, P. D. (2006). *Algoritmos de procesamiento de imágenes en FPGA.* Barranquilla, Colombia: Universidad del Norte.

International, I. (2009). *Hardware accelerated montecarlo financial simulation overlow cost FPGA cluster Parallel & Distributed Processing.*IPDPS.

Jesus Carlos Pedraza Ortega, G. H. (2011). *Three Dimensional Reconstruction Strategies Usign a PRofilometrical Approach based on Fourier Transform.* México-Querétaro: Universidad Autónoma de Querétaro,Facultad de Informática.

Julio, P. (s.f.). *Prototipado en FPGAs para inyección de fallas, Aplicación a sistemas distribuidos sobre bus CAN.* Italia: Alfa, EuropeAid.

Logroño, F. S. (2011). *Diseño e Implementación en FPGA del Método Recursivo Gram-Schmidt.* Valencia: Escuela Politecnica Superior de Gandia.

Maher, J. (2009). *Direct Digital Frequency Synthesizer with CORDIC Algorithm and Taylor Series Approximation for Digital Receivers.* France: European Journal of Scientific Research, Publishing.

Mazón, D. R. (2011). *Tesis Doctoral, Implementación de Funciones Elementales en Dispositivos FPGA.* Valencia: Universidad Politécnica de Valencia.

Micco De L., L. H. (2010). *Implemetación en FPGA de Ruido Gaussiano para simulaciones en Hardware.* Buenos Aires, Argentina: Facultad de Ingeniería UNMDP, CONICET.

Página Oficial de Xilinx: http://www.xilinx.com/products/silicon-devices/fpga/index.htm

Pasaye, J. J. (2010). *Procesamiento Digital de Señales, Transformada de Fourier.* FIE-UMSNH.

Posadas, D. L. (1998). *Transformada Rápida de Fourier e Interpolación en Tiempo Real, Algoritmos y aplicaciones, Proceso de datos en sistemas de tiempo real.* Valencia: Universidad Politécnica de Valencia.

Rionda, S. B. (2004). *Aplicación de los Métodos Numéricos a Problemas de Ingeniería.* Centro de Investigación en Matemáticas A.C.

Ríos, C. P. (2006). *Implementación de Arquitecturas para el cálculo de funciones trascendentales empleando el algoritmo CORDIC en un FPGA.* Lima- Perú: Pontificia Universidad Católica de Perú.

Rocha, G. (2005). *Métodos Numéricos "Aproximación numérica y errores. Serie de Taylor".*

S. Botello, M. M. (2007). Aplicación del Cómputo paralelo a la modelación de sistemas continuis en ciencias e ingeniería. *Metodos Numéricos en Ingeniería y Ciencias Aplicadas.* México: UMSNH.

Señales, 5. C. (1999). *Transformada Discreta de Fourier y Transformada Rápida de Fourier.*

Serafín Pérez López, E. S. (2006). *Diseño de sistemas digitales con VHDL.* Madrid España: Spain Paraninfo.

Tischer, I. (2009). Metodos Numéricos "El desarollo de Taylos". Escuela de Ingeniería y Computación, Universidad del Valle Cali.

Torres, F. J. (2008). *Lenguajes de descripción de hardware.* Universidad Autónoma de Guadalajara.

Vidal D. Ismael, H. D. (2007). *Evaluación e Implementación en Hardware DSP de un Algoritmo para Estimación de Frecuencias.* Santiago de Chile: Fac. Ingeniería.

XILINX. (26 de Agosto de 2009). Spartan-3E FPGA Family: Data Sheet.

XILINX. (n.d). *XILINX ALL PROGRAMMABLE.* Retrieved Septiembre 29,2012, from http://www.xilinx.com/products/silicon-devices/fpga/index.html