INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN Y DESARROLLO DE TECNOLOGÍA DIGITAL

"DISEÑO E IMPLEMENTACIÓN EN UN FPGA DE OSCILADOR CAÓTICO PARA APLICACIONES EN SISTEMAS DE SEGURIDAD"

TESIS

QUE PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS

PRESENTA:

ING. PABLO JOSUÉ OBESO RODELO

BAJO LA DIRECCIÓN DE:

DR. JOSÉ CRUZ NÚÑEZ PÉREZ

DR. ESTEBAN TLELO CUAUTLE

TIJUANA B.C., **JUNIO 2015**

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de Tijuana, B.C. siendo las 11:00 horas del día 12 del mes de junio del 2015 se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de CITEDI para examinar la tesis titulada:

DISEÑO E IMPLEMENTACIÓN EN UN FPGA DE OSCILADOR CAÓTICO PARA APLICACIONES EN SISTEMAS DE SEGURIDAD.

Presentada por el alumno:

OBESO	**RODELO**	**PABLO JOSUÉ**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	3	0	9	5	0

aspirante de:

MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES

Después de intercambiar opiniones, los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

DR. JOSÉ CRUZ NÚÑEZ PÉREZ

DR. ESTEBAN TLELO CUAUTLE

DR. MOISÉS SÁNCHEZ ADAME

M. en C. ANDRÉS CALVILLO TÉLLEZ

DR. LUIS TUPAK AGUILAR BUSTOS

PRESIDENTE DEL COLEGIO DE PROFESORES

DRA. MIREYA SARAÍ GARCÍA VÁZQUEZ

S.E.P.
INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN Y DESARROLLO
DE TECNOLOGÍA DIGITAL
DIRECCIÓN

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de Tijuana, Baja California, el día 12 del mes Junio del año 2015, el (la) que suscribe Pablo Josué Obeso Rodelo alumno (a) del Programa de MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES con número de registro B130950, adscrito al CENTRO DE INVESTIGACIÓN Y DESARROLLO DE TECNOLOGÍA DIGITAL, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. José Cruz Núñez Pérez y del Dr. Esteban Tlelo Cuautle, cede los derechos del trabajo titulado DISEÑO E IMPLEMENTACIÓN EN UN FPGA DE OSCILADOR CAÓTICO PARA APLICACIONES EN SISTEMAS DE SEGURIDAD al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección Av. del Parque 1310, Mesa de Otay, Tijuana, Baja California 22510, México, o a la dirección electrónica: posgrado@citedi.mx
Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Pablo Josué Obeso Rodelo

Nombre y firma

Dedicatoria

A mis padres:
Nicolás Obeso Armenta y Amelia Rodelo Pérez

A mis hermanos:
Carlos Alberto Obeso Rodelo, Diego Nicolás Obeso Rodelo y Karla Pahola Obeso Rodelo

A mi sobrina:
Nicole Dánae Arenas Obeso

Agradecimientos

A **mis padres** por su incondicional apoyo, orientación y confianza, gracias a eso he llegado a realizar una de mis más grandes metas, lo cual constituye la herencia más valiosa que pudiera recibir, agradezco también a mis tías **Ana** y **Erika** porque fueron pieza clave al apoyarme siempre en el transcurso de mi estancia en Tijuana.

A mis directores de tesis, el **Dr. José Cruz Núñez Pérez** y el **Dr. Esteban Tlelo Cuautle** por su guía en el desarrollo de este trabajo de tesis, sus sabios consejos y los diversos conocimientos que me transmitieron, también le agradezco al **Dr. José de Jesús Rangel Maldonado** por su apoyo y sus enseñanzas en mi estancia en INAOE y sobre todo les agradezco que a pesar de tener una relación de respeto entre asesor y alumno siempre me brindaron su amistad.

Agradezco al **Instituto Politécnico Nacional**, al **Centro de Investigación y Desarrollo de Tecnología Digital** y al **Consejo Nacional de Ciencia y Tecnología** por el apoyo brindado.

Agradezco a mi comité tutorial constituido por el **Dr. Luis Tupak Aguilar Bustos**, el **Dr. Moisés Sánchez Adame** y el **M. C. Andrés Calvillo Téllez** por dirigirme y retroalimentarme con sus comentarios y revisiones en este trabajo de investigación.

Agradezco a mis compañeros y amigos de laboratorio de Microondas Katherine, José, Aarón, Eduardo, Galaviz, Thaimi, a mis amigos del centro de investigación Daniel, Fernando, Ajelet, Jesús, Edilberto, Esteban, Luis, Andrés, Joshua, Diana, Ricardo, Oscar, Leopoldo, Carlos, Jovan, gracias por ser parte de esto y por los gratos momentos que compartimos en el transcurso de mi estancia en la maestría.

“DISEÑO E IMPLEMENTACIÓN EN UN FPGA DE OSCILADOR CAÓTICO PARA APLICACIONES EN SISTEMAS SEGURIDAD”

RESUMEN

En la actualidad es común que en distintos campos de la ingeniería se utilicen los sistemas caóticos y modelos basados en caos. Normalmente estos sistemas se realizan a partir de un modelo matemático, el cual es un sistema de ecuaciones de estados. Este modelo se puede representar mediante distintos circuitos integrados como amplificadores operacionales, sin embargo como dichos circuitos necesitan de componentes como resistencias, capacitores, inductores, estos tienden a tener problemas de error debido a la tolerancia de dichos componentes.

El presente trabajo muestra como principal aportación la metodología para realizar el diseño e implementación de dos tipos de osciladores caóticos de tiempo continuo: el oscilador caótico basado en series de funciones saturadas y el oscilador caótico basado en el circuito de Chua. Se presenta la metodología para que ambos sistemas sean capaces de la generación de 2 a 6 enrollamientos.

Los sistema caóticos se pueden resolver de distintas maneras, el uso de los métodos numéricos nos da una increíble ventaja la cual es discretizar el sistema mediante iteraciones, para este trabajo se optó por el uso de los método de Euler y Runge-Kutta de orden 4. A partir de la descripción matemática se realizó un diseño digital, realizando cada componente mediante código VHDL la descripción matemática.

Los diseños digitales se compilaron utilizando el Software Active-HDL, que además de compilar, permite realizar una co-simulación con Matlab-Simulink, lo cual facilita notablemente el análisis de los resultados antes de realizar la implementación físicamente, incluso la capacidad de almacenar los datos obtenidos como vectores en MATLAB. Además se presentaron los resultados experimentales de ambos sistemas generando multi-enrollamientos, obteniendo los resultados esperados.

Como una segunda aportación y para comprobar la viabilidad de las arquitecturas digitales de los osciladores caóticos diseñadas, se realizó una aplicación donde a partir de la sincronización de dos osciladores caóticos se llevó a cabo la transmisión de imágenes de diferente tamaño, añadiendo caos a una imagen en la etapa de transmisión y sustrayendo dicho caos en la etapa de recepción. Finalmente, los resultados experimentales confirman la utilidad que conlleva el realizar sistemas de comunicación caóticos para el procesamiento de imágenes mediante el uso de FPGAs.

Palabras clave: aritmética computacional, Chua, formas Hamiltonianas, FPGA, métodos numéricos, oscilador caótico de multi-enrollaimentos, sincronización, SNLF.

“FPGA DESIGN AND IMPLEMENTATION OF A CHAOTIC OSCILLATOR FOR APLICATIONS IN SECURITY SYSTEMS”

ABSTRACT

Today, the use of chaotic systems and chaos based models is widespread in several fields of engineering. Typically these systems are made from a mathematical model, which is a system of equations of state. This model can be represented by various integrated circuits and operational amplifiers, however, these circuits require components such as resistors, capacitors, inductors, and these circuits tend to have errors due to the tolerance components.

This work shows as a main contribution the methodology for the design and implementation of two types of continuous time chaotic oscillators: the chaotic oscillator based on series of saturated functions and chaotic oscillator based on Chua circuit. Both systems are capable of generating 2 to 6 scrolls.

Chaotic systems can be solved in different ways, the use of numerical methods gives us an incredible advantage which is discretize the system through iterations, for this work we chose to use the Euler method and Runge-Kutta of order 4. A digital design was based on the mathematical description, implementing each component by using VHDL code for mathematical description.

Digital designs were compiled using the Software Active-HDL, this software permits a co-simulation with Matlab-Simulink, which greatly facilitates the analysis of the results prior to physical implementation, including the ability to store data as vectors in MATLAB. Further, the experimental results of both multi-scroll chaotic oscillators was presented, obtaining the expected results.

As a second contribution and to test the feasibility of the designed chaotic oscillators architectures, an application showing the synchronization of two chaotic oscillators for the transmission of images of different sizes was made, by adding chaos to an image at the transmission stage and by subtracting chaos at the recover stage. Finally, the experimental results confirm the appropriateness on realizing chaotic communication systems for image processing by using FPGAs.

Keywords: Chua, computer arithmetic, FPGA, Hamiltonian forms, multi-scroll chaotic oscillator, numerical methods, SNLF, synchronization.

TABLA DE CONTENIDO

Lista de figuras

Lista de tablas

Glosario de términos

ATRACTOR	Comportamiento al que tiende un sistema después de un tiempo de evolución, independientemente de las condiciones iniciales del mismo.
CAOS	Comportamiento aparentemente errático e impredecible de algunos sistemas dinámicos, aunque su formulación matemática sea en principio determinista.
COMPILACIÓN	Proceso de traducción de un código fuente (escrito en un lenguaje de programación de alto nivel) a lenguaje máquina (código objeto) para que pueda ser ejecutado por la computadora.
CORRELACIÓN	Indica la fuerza y la dirección de una relación lineal y proporcionalidad entre dos variables estadísticas.
ENCRIPTACIÓN	Es el número de ciclos por segundo de una señal; está dada por: $f = 1/T$, donde T es el periodo.
FRECUENCIA	Es el proceso para volver ilegible información considerada importante. La información una vez encriptada sólo puede leerse aplicándole una clave.
MATLAB	Programa informático utilizado para realizar cálculos matemáticos y ejecución de funciones con extensión *.m.
MATRIZ PROGRAMABLE	Es una red de conductores distribuidos en filas y columnas con un fusible en cada punto de intersección, las matrices pueden ser fijas o programables
OSCILADOR	Aparato que produce oscilaciones eléctricas o mecánicas.
OSCILADOR CAÓTICO	Aparato que produce oscilaciones eléctricas o mecánicas con un comportamiento continuamente diferente y desordenado.
PROTOTIPO	Son una técnica de uso común en algunas ingenierías cuando los desarrollos son complejos, laboriosos y caros, o no están completamente especificados.

REALIMENTACIÓN POSITIVA	Es uno de los mecanismos de realimentación por el cual los efectos o salidas de un sistema causan efectos acumulativos a la entrada.
SIMULACIÓN	Es la imitación de un fenómeno ó comportamiento de un sistema utilizando un software computacional.
SIMULINK	Es un entorno de diagramas de bloque para la simulación multi-dominio y el diseño basado en modelos.
SINCRONIZACIÓN	Hacer que coincidan en el tiempo dos o más movimientos o fenómenos.
SÍNTESIS	Proceso por el cual se convierte una descripción de circuito en un archivo software estándar generado a partir de un software de compilación que se emplea en un dispositivo de programación.

Lista de Siglas y Acrónimos

ABEL	*Advanced Boolean Expression Language*, Lenguaje Avanzado de Expresiones Booleanas.
ASIC	*Application-Specific Integrated Circuit*, Circuito Integrado para Aplicaciones Específicas.
CNN	*Celular Neural Network*, Red Neuronal Celular.
CLB	Componentes lógicos programables.
DSP	*Digital Signal Processor*, Procesador Digital de Señales.
ED	Ecuaciones diferenciales.
EDO	Ecuaciones diferenciales ordinarias.
FPGA	*Field Programmable Gate Array*, Arreglo de compuertas lógicas programables por efecto campo.
GPAN	Generador de números pseudo-aleatorios.
GCPAN	Generador caótico de números pseudo-aleatorios.
HSMC	*High Speed Mezzaine Card*, Tarjeta intermediaria de alta velocidad.
IOB	*Input Output Banks*, Bancos de entrada-salida.
LUT	*Lookup Table*, Tabla de consulta.
MUX	*Multiplexer*, Multiplexor.
PWL	*Piece-wise linear*, Linealizadas a tramos.
RAM	*Random-Access Memory*, Memoria de Acceso Aleatorio.
ROM	*Read-Only Memory*, Memoria de solo lectura.
SNLF	*Satured Nonlienar Function Series*, Series de funciones no lineales saturadas.

VHDL	Combinación de VHSIC (*Very High Speed Integrated Circuit*, Circuitos Integrados de Velocidad Muy Alta) y HDL (*Hardware Description Language*, Lenguaje de Descripción de Hardware).

Capítulo 1

Introducción

1.1 ANTECEDENTES

En los últimos años se ha intensificado el estudio de los sistemas dinámicos debido a su gran interés y a sus potenciales aplicaciones en la ciencia y la tecnología.

Los sistemas caóticos fueron descubiertos accidentalmente a partir de algunas observaciones por el meteorólogo Edward Lorenz quien, en 1963, trabajaba en un programa para predecir el clima y el comportamiento de la atmósfera. Se dio cuenta que con valores ligeramente diferentes de temperatura predecía comportamientos radicalmente distintos en el clima. En un inicio se creyó que se trataba de errores numéricos provenientes del cálculo en la computadora. Tiempo después se reunió la suficiente evidencia de que lo que pasaba era un fenómeno único de alta sensibilidad del sistema a las condiciones iníciales, que ahora se sabe caracteriza, entre otras cosas, a un sistema caótico [1].

El caos no es un completo desorden, es un desorden determinista que siempre es previsible en un corto plazo, se genera a partir de un sistema dinámico no lineal, que corresponde a un comportamiento aperiódico, caótico y sensible a pequeños cambios de valores iniciales y parámetros de control [2]. La teoría del caos permite comprender fenómenos de la naturaleza, como las formas que exhibe y los patrones de conducta a los que obedece. Más allá de esto, aparece también como una herramienta valiosa para entender el comportamiento de la conducta humana y social, los fenómenos económicos, así como la evolución de la tecnología y de la actividad industrial. Ante estas perspectivas parece que esta teoría podría aplicarse como modelo para explicar la conducta de los sistemas reales [3].
La generación de secuencias pseudo-aleatorias siempre ha atraído una atención considerable ya que los número aleatorios juegan un rol muy importante en muchas áreas del conocimiento tales como ingeniería, criptografía, telecomunicaciones, física, economía, estadística, métodos de Monte Carlo, etc.) [4].

Hay dos enfoques cuando se aplica la dinámica caótica en criptografía, el primero utiliza sistemas caóticos para la generación de secuencias pseudo-aleatorias, que posteriormente son empleadas como flujos de códigos para enmascarar la información en múltiples maneras. En el segundo enfoque, la información se usa como estado inicial y el código se va desprendiendo a partir de la

órbita que ha sido generada, por consecuencia el primer enfoque corresponde al flujo de códigos, mientras que el segundo al bloqueo de códigos, ambos como claves de criptografías secretas y públicas [5].

Teóricamente, el caos basado en Generadores de Números Pseudo-Aleatorios (GPAN) se caracteriza por que cuenta con una buena aleatoriedad y períodos infinitos, mientras que los caracteres no lineales aumentan significativamente la complejidad de las estructura de los GPANs' [6].

Un sistema que presenta un comportamiento caótico se distingue porque si le hace a partir de dos condiciones iniciales levemente diferentes, las trayectorias que sigue el sistema en ambos casos se separan exponencialmente en el tiempo. Este fenómeno, que aparece solamente cuando las ecuaciones que gobiernan el sistema son no lineales, se conoce como la sensibilidad a las condiciones iniciales. El matemático francés Poincaré (1854 – 1912), fue el primero que reconoció este fenómeno y lo ha descrito de la siguiente manera:

" ... puede ocurrir que pequeñas diferencias en las condiciones iniciales produzcan diferencias muy grandes en el fenómeno final. Un pequeño error en el comienzo produce un error enorme al final. La predicción resulta imposible, y nos encontramos, por lo tanto, con un fenómeno fortuito."

Si la predicción resulta imposible, es evidente que un sistema caótico se asemeja a un sistema estocástico. Sin embargo, la fuente de irregularidad es bastante diferente. En el caos, la irregularidad es parte de la dinámica intrínseca del sistema que no puede atribuirse a influencias externas impredecibles. De hecho, los sistemas dinámicos caóticos obedecen ecuaciones deterministas [7].

1.2 PROBLEMÁTICA

En los últimos años, la revolución de los medios de intercambio de información, la cantidad y el alto débito de información a través de la Internet, los satélites, móviles y cualquier otro tipo de red, hacen fácil el acceso al contenido informático. El mayor problema es proteger la información confidencial; la mayor parte de los sistemas de cifrado actuales son incapaces de resistirse a la evolución de los ataques de hackers (interceptores enemigos, espías, intrusos, criptoanalistas, decodificadores, enemigos), ya sea que la información esté almacenada localmente en una máquina o sea transmitida a través de la red. Por lo anterior el desarrollo de criptosistemas más robustos se ha convertido en una necesidad importante, además que esta tendencia está acompañada de la necesidad del uso eficiente del espectro en frecuencia, debido a la gran cantidad de tecnologías emergentes que funcionan a altas frecuencias.

Como muchos sistemas basados en la generación de secuencias pseudo-aleatorias para la ocultación de mensajes, la aplicación de osciladores caóticos como generadores de dichas secuencias, se ha convertido en un campo de estudio importante por muchos años.

En México el desarrollo de esta tecnología aun es un reto importante, ya que en revistas, congresos de investigación tecnológica o en la literatura en general no se han presentado diseños digitales

completos, por lo cual esta comunidad está quedando a deber en esta área específica. Se está consciente que la encriptación de la información a nivel software tiene ya gran desarrollo en la industria, sin embargo, es importante crear circuitos digitales que realicen estas labores, representando así una oportunidad en el campo de las telecomunicaciones. Por lo que el estudio de la implementación digital de este tipo de sistemas representa una oportunidad de mejora en cuanto a portabilidad y simplicidad, siendo posible probar diseños antes de implementarlos.

La amplia existencia de funciones caóticas proporciona incontables opciones que en general, incrementan los métodos para la generación de números pseudo-aleatorios para mejorar la seguridad de las comunicaciones. Por lo tanto, se han propuesto muchos Generadores Caóticos de Números Pseudo-Aleatorios (GCPAN) en la literatura, pero la realización de hardware, especialmente, la implementación del chip todavía es un gran reto.

Las comunicaciones seguras usando sincronización entre sistemas caóticos es un nuevo concepto. Las grandes ventajas de este tipo de comunicación segura por hardware han llevado el progreso de este campo rápidamente. Los sistemas de comunicación se encuentran donde quiera que se transmita información de un punto a otro; son necesarios para los negocios, la industria, la banca y la circulación de información al público.
En el mundo de la tecnología de la información y de la comunicación, el caos parece jugar un papel importante, en la actualidad ha aumentado la demanda de servicios de comunicación, que es soportada por un rápido incremento en la transmisión y capacidad en redes de comunicación. Dos cuestiones en estas redes son privacidad y seguridad. El software adecuado es usual para la codificación de datos, pero el continuo incremento en la velocidad de las computadoras amenaza la seguridad de estos procesos.
Una contribución para resolver este problema es usar señales, portadoras caóticas generadas por componentes que funcionan en el régimen no-lineal, tales como circuitos electrónicos no lineales, láseres de semiconductor y, a nivel de software, ecuaciones diferenciales no lineales [8].
También dentro de las comunicaciones, la modulación usando señales caóticas (señales generadas por un sistema no lineal en estado caótico) resulta potencialmente muy interesante en aplicaciones de espectro ensanchado y comunicaciones seguras debido a su naturaleza de banda ancha y aspecto similar al ruido. Además, las señales caóticas resultan sencillas de generar.

Respecto a la codificación de la información, el carácter fractal del atractor caótico permite asociar bits a distintas regiones del mismo de manera tal que un cambio de una región a otra pueda realizarse mediante una interacción de valor prácticamente nulo con el sistema caótico. Por lo tanto, si se consigue definir una gramática apropiada, un atractor caótico puede ser aprovechado para codificar mensajes mediante bits de información con un valor activo de conmutación despreciable [9].

Por otro lado, el caos determinista también puede ser utilizado en comunicación segura. El caos tiene características de frecuencia que lo hacen similar a un ruido aleatorio, propiedad que lo hace resistente a las técnicas habituales de filtrado para separar información sobrepuesta sobre una misma señal. Si un mensaje representado en bits de baja intensidad se superpone a una portadora aleatoria, la portadora enmascarará el mensaje de manera tal que resultará difícil explicar si se está transmitiendo un mensaje y, mucho más difícil, extraerlo. El modo natural de recuperar el mensaje por el receptor sería reproducir la portadora aleatoria y restarla de la señal recibida, de esta manera

el mensaje reaparecería nuevamente. La dificultad radica en que una señal aleatoria, por su propia naturaleza, no puede ser reproducida y, en consecuencia, el mensaje quedará oculto para siempre [10].

1.3 JUSTIFICACIÓN

Un oscilador caótico puede utilizarse para la generación de secuencias pseudoaleatorias y una de sus principales aplicaciones es en el campo de la criptografía para el desarrollo de criptosistemas más robustos y seguros.

Ya que este trabajo de tesis se enfoca en realizar el diseño digital e implementación de un oscilador caótico en FPGA, una vez que la tarjeta FPGA está configurada se puede especificar o modelar tanto la estructura como la función del circuito digital lo cual reduce notablemente la complejidad de la implementación del chip para aplicaciones más específicas. Esto debido al reto que ofrece la realización en hardware y la complejidad de la implementación de un diseño analógico, tomando en cuenta un uso eficiente de potencia y la capacidad de trabajar con frecuencias más altas.

Una de las ventajas de los sistemas digitales es que son muy adecuados a su fabricación en serie debido a que no presentan los problemas de tolerancia tan crítico que tienen sus equivalentes analógicos, también aprovechan componentes de la industria de la microeléctrica, de los computadores, por lo que también se benefician de las economías de escala que se producen en esta industria.

Además también se cuenta con herramientas como programas de simulación y síntesis de sistemas dinámicos. Estas herramientas de programación (software) se utilizan para verificar el comportamiento del modelo de hardware antes que sea construido, ya sea mediante simulación o co-simulacion tipo sistema, para posteriormente realizar la síntesis del modelo en un circuito, aplicando una tecnología de componente en particular; representando así una herramienta muy versátil para analizar muchas de las características asociadas a osciladores caóticos como: orbitas estables, bifurcaciones, atractores. Esto ayudará a la integración de sistemas distintos, ya que son comunes las tareas de conmutación y control. Además las interfaces son fáciles de realizar y estandarizar, implementando así una herramienta de análisis y diseño de osciladores caóticos.

Otras herramientas muy poderosas en las cuales se basa esta tesis para la solución de los sistemas caóticos son los métodos numéricos. Son capaces de manipular sistemas de grandes ecuaciones, manejar no linealidades y resolver geometrías complicadas, a menudo, imposibles de resolver en forma analítica. Los métodos numéricos son un vehículo eficiente para aprender a servirse de las computadoras. Es bien sabido que una forma efectiva de aprender programación consiste en escribir programas de computadora. Debido a que la mayoría de los métodos numéricos están diseñados para usarlos en las computadoras, son ideales para tal propósito.

Los métodos numéricos son un medio para reforzar la comprensión de las matemáticas, ya que una de sus funciones es convertir las matemáticas superiores en operaciones aritméticas básicas, de esta manera se puede profundizar en los temas que de otro modo resultarían oscuros. Esta perspectiva

dará como resultado un aumento de la capacidad de comprensión y entendimiento de lo que se estudia.

1.4 OBJETIVO GENERAL Y OBJETIVOS ESPECÍFICOS

El objetivo general de esta tesis de maestría consiste en el diseño e implementación de un oscilador caótico en una tarjeta FPGA a través de métodos numéricos para posteriores aplicaciones en sistemas de seguridad.

Los objetivos específicos que se persiguen en este trabajo de investigación son:

- ✓ Realizar estudio bibliográfico del estado del arte sobre osciladores caóticos. Realizando una revisión de los principales modelos matemáticos de circuitos osciladores, con énfasis en los modelos caóticos.

- ✓ Describir los programas de simulación que se utilizarán en el marco de este trabajo de tesis, se trata de Matlab/Simulink y así como también con el lenguaje VHDL para implementación en la tarjeta de tipo DSP o FPGA. Realizar implementación de un oscilador caótico en FPGA mediante la herramienta DSP-Builder.

- ✓ Sintetizar trabajos realizados con la consideración de modelos de oscilación caóticos. Se presentan los enfoques matemáticos elegidos para aplicar el diseño de un oscilador caótico a través de simulación de tipo sistema.

- ✓ Implementar en simulación, co-simulación y en un FPGA un circuito oscilador caótico basado en SNLF empleando señales elementales de comunicación digital, buscando precisión y optimización en el diseño mediante el código VHDL y realizar diseños de multi-enrollamientos para oscilador caótico con ayuda de las SNLF.

- ✓ Implementar en simulación, co-simulación y en un FPGA un circuito oscilador caótico basado en el circuito de Chua y realizar diseños de multi-enrollamientos.

- ✓ Análisis y comparación de los resultados obtenidos de las arquitecturas diseñadas, comparando aspectos de rapidez computacional, consumo en memoria y recursos lógicos utilizados.

- ✓ Sincronizar dos sistemas caóticos para transmisión de información.

- ✓ Realizar una aplicación en la cual se transmita una imagen monocromática y otra en escala de grises a través de los sistemas caóticos sincronizados.

1.5 APORTACIONES

La contribución principal del presente trabajo de investigación consiste en el diseño e implementación de arquitecturas digitales de osciladores caóticos basados en SNLF y en el circuito de Chua, cada caso resuelto a través de los métodos numéricos de Euler y Runge-Kutta, ampliando también estas arquitecturas para generar multi-enrollamientos, las arquitecturas mencionadas son implementadas en una tarjeta FPGA.

- Se diseñaron arquitecturas digitales de dos casos de osciladores caóticos uno basado en SNLF y otro en el circuito de Chua, ambos casos resueltos mediante el método de Euler y Runge-Kutta.

- Se diseñaron modelos capaces de generar multi-enrollamientos de ambos tipos de osciladores caóticos.

- Se implementaron todas las arquitecturas en FPGA.

- Se logró la sincronización de dos osciladores caóticos basados en SNLF mediante formas hamiltonianas y observadores, para los casos de 2 y 6 enrollamientos, estas sincronizaciones se diseñaron en forma de arquitecturas mediante código VHDL.

- Se probó en forma de simulación y co-simulacion la factibilidad de la arquitectura de los osciladores sincronizados, y se logró transmitir dos imágenes de diferente tamaño, obteniendo los resultados esperados.

1.6 ORGANIZACIÓN DEL DOCUMENTO

Este documento de tesis está organizado en seis capítulos. En el capítulo 1, se presentan los antecedentes teóricos, se plantea la problemática, justificación, se describen los objetivos generales y específicos, así como las aportaciones.

En el capítulo 2 se presenta teoría general de oscilación, teoría del caos, el estado del arte de los osciladores caóticos, la descripción de los osciladores caóticos basados en SNLF y el circuito de Chua, se describen los métodos numéricos usados en este marco de tesis y su importancia, también se presentan teoría de sincronización de sistemas caóticos.

En el capítulo 3 se presentan las simulaciones correspondientes a los dos tipos de sistemas caóticos utilizados, específicamente el de SNLF y el del circuito de Chua, ambos resueltos mediante el método de Euler y Runge-Kutta orden 4, se muestran sus extensiones a multi-enrollamientos para ambos tipos de osciladores. Además se muestra una manera alternativa de simular el oscilador caótico basado en el circuito de Chua utilizando la herramienta Simulink.

En el capítulo 4 se describen las características de la tarjeta FPGA a utilizar y su tarjeta de adquisición de datos HSMC, se lleva cabo las descripción de la metodología empleada para realizar la implementación en un FPGA-DSP Cyclone III Edition de Altera del oscilador caótico de Chua

mediante la herramienta DSP-Builder y Simulink. Además se muestra otra metodología para la implementación en FPGA de los osciladores caóticos basados en SNLF y el circuito de Chua a través del diseño de arquitecturas mediante código VHDL, ambos tipos de osciladores resueltos mediante el método de Euler y Runge-Kutta, y al final se muestran los resultados en co-simulacion y experimentales.

En el capítulo 5 se muestra la sincronización de dos sistemas caóticos basados en SNLF mediante formas hamiltonianas y observadores en un sistema maestro-esclavo, el sistema de la sincronización es implementado mediante código VHDL en el software Active-HDL, se simula y se co-simula para ver los resultados. Se prueba la eficacia del sistema maestro-esclavo, realizando una aplicación compacta donde se transmiten dos imágenes de diferente tamaño embebidas dentro del mismo diseño, y se observan los resultados con ayuda de la simulación y co-simulacion entre Active-HDL y Matlab.

Finalmente en el capítulo 6, se plasman las conclusiones obtenidas en base a los resultados obtenidos en este trabajo de investigación. Finalmente en este capítulo se mencionan los trabajos futuros y recomendaciones.

Capítulo 2

Marco teórico

2.1 INTRODUCCIÓN

En la actualidad, los sistemas caóticos y aplicaciones basadas en el caos se utilizan comúnmente en campos de la ingeniería. Una de sus principales aplicaciones se encuentra en el diseño e implementación de generadores de señales caóticas, los cuales tienen un papel importante, particularmente en la comunicación caótica y criptografía.

La literatura indica que los generadores de señales caóticas diseñados en hardware se pueden implementar ya sea de manera digital o analógica, siendo la manera digital la más ventajosa. Se han propuesto diseños de sistemas caóticos de tesis de maestría, pero su implementación en hardware todavía es un campo no muy conocido. En este trabajo se realiza la implementación de sistemas caóticos de manera digital, mostrando una nueva metodología de diseño, a través de modelos matemáticos resueltos mediante métodos numéricos. Los sistemas caóticos que se utilizan en este marco de tesis son basados en SNLF (Series de funciones no lineales saturadas) y el circuito de Chua, siendo estos dos los sistemas más populares que exhiben caos debido a su robustez, facilidad de implementación, su simple no linealidad estática y sus diversas características de bifurcación lo que contribuyen a su éxito entre teóricos y diseñadores de la teoría del caos. Por otra parte las aplicaciones de dichos sistemas conllevan a que el problema de sincronización reciba un enorme interés. Esta propiedad permite tener interesantes aplicaciones en diferentes campos, como por ejemplo, generación de señales de banda ancha, redes neuronales celulares (CNNs) y sobre todo para diseñar sistemas de comunicaciones más seguros.

2.2 TEORÍA DE OSCILADORES

Un oscilador es un circuito que genera una señal periódica, es decir, que produce una señal transitoria a la salida sin tener ninguna entrada periódica. Los osciladores se clasifican en armónicos, cuando la salida es sinusoidal, o de relajación, si generan una onda cuadrada [11].

Existen muchos tipos de osciladores, y configuraciones diferentes de circuitos que producen oscilaciones. Algunos osciladores producen señales sinusoidales, otros producen señales no sinusoidales. Los osciladores no sinusoidales, tales como los osciladores de pulso y rampa (o diente

de sierra), encuentran su uso en aplicaciones de temporización y control. Los osciladores de pulso se encuentran comúnmente en los relojes de sistemas digitales, y los osciladores de rampa se encuentran en circuitos de barrido horizontales de osciloscopios y televisores. Los osciladores sinusoidales se utilizan en muchas aplicaciones, por ejemplo en equipos electrónicos de entretenimiento (tales como radios, televisores, y reproductores de video), en equipos de prueba (por ejemplo analizadores de redes y generadores de señales), y en los sistemas inalámbricos [12].

Generalmente, un circuito oscilador está compuesto por: 1) un circuito oscilante, 2) un amplificador y 3) una red de realimentación tal y como se muestra en la figura 2.1.

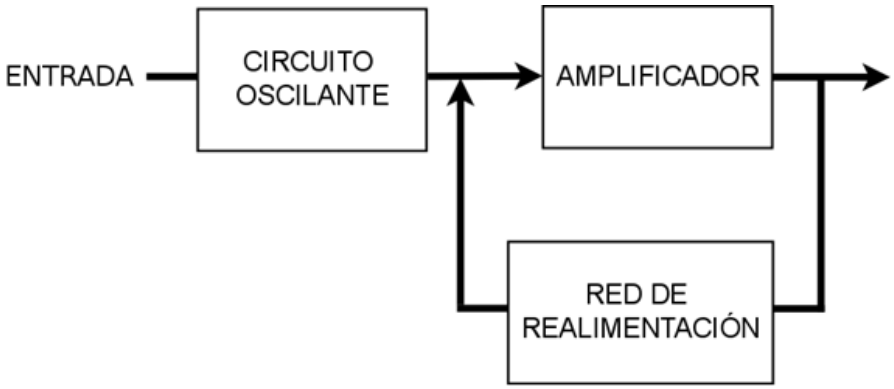

Figura 2.1. Diagrama esquemático de un oscilador.

El circuito oscilante suele estar compuesto por un inductor (o bobina) y por un condensador (capacitor) conectados en paralelo (véase la figura 2.2). El funcionamiento de los circuitos osciladores suele ser muy similar en todos ellos; el circuito oscilante produce una oscilación, el amplificador la aumenta y la red de realimentación toma una parte de la energía del circuito oscilante y la introduce de nuevo en la entrada produciendo una realimentación positiva.

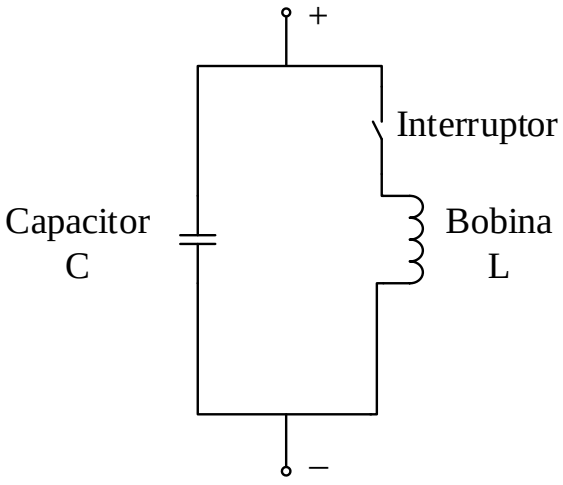

Figura 2.2. Circuito oscilante.

No debe confundirse "circuito oscilante" con "oscilador". El circuito oscilante es el encargado de producir las oscilaciones deseadas; sin embargo, no es capaz de mantenerlas por sí solo. El oscilador es el conjunto que forman el circuito oscilante, el amplificador y la red de realimentación juntos.

Supóngase el circuito de la figura 2.2 compuesto por un condensador y un inductor conectados en paralelo. En primer lugar, se conecta el condensador a una batería. Entonces, comienza a circular corriente eléctrica que va a provocar que el condensador se cargue. Llegado este momento, la corriente eléctrica dejaría de circular y el condensador se encontraría totalmente cargado. A continuación se mueve el interruptor y se conecta el condensador con la inductancia. En este mismo instante, la bobina, en principio, se opone al paso de la corriente. Sin embargo, comienza a circular corriente de forma progresiva haciendo que el condensador se descargue y creando un campo magnético en la bobina. Al cabo de cierto tiempo, la corriente eléctrica comienza a cesar de forma progresiva y, por lo tanto, el campo magnético se reduce. Se crea entonces una tensión inducida en la bobina que hace que el condensador se cargue de nuevo, pero esta vez con la polaridad contraria. Una vez que el condensador se encuentra totalmente cargado se vuelve al estado inicial, aunque esta vez con el condensador cargado de forma inversa a su estado anterior. Comienza pues otra vez el proceso de descarga progresiva del condensador sobre la inductancia y de nuevo vuelve a cargarse el condensador. Esto ocasiona que la corriente vaya y venga de un elemento a otro. Esto es lo que se conoce como circuito oscilante [13].

Este circuito oscilante podría ser un oscilador si fuese capaz, por sí solo, de mantener su oscilación indefinidamente. Sin embargo, en la realidad existe una pérdida de energía que hace que la corriente oscilante se vaya atenuando progresivamente hasta llegar a desaparecer. Esto es debido a que la inductancia posee una cierta resistencia óhmica que hace que con el paso de la corriente se vaya perdiendo cada vez una pequeña cantidad de energía y convirtiéndose en calor.
La frecuencia con la que oscila el circuito depende evidentemente del condensador y de la inductancia que se coloque; cuanto mayor sea el condensador y la inductancia, menor va a ser la frecuencia. Una vez dispuestos ambos elementos en el circuito, estos son fijos y, por tanto, la frecuencia de oscilación es una característica de dicho circuito, la cual recibe el nombre de "frecuencia propia del circuito oscilante". En realidad es bastante complicado acertar en la elección del condensador y de la inductancia a la hora de obtener una determinada frecuencia. Lo que se suele hacer es poner, por ejemplo, un condensador con capacidad variable que, una vez funcionando en el circuito, se ajusta dicho condensador hasta obtener el valor de la frecuencia de oscilación deseada.

Un circuito oscilante por sí solo no es capaz de mantener por mucho tiempo sus oscilaciones y, por tanto, no es de ninguna utilidad. Para solventar este problema lo que se hace es proporcionar una "ayuda extra" desde el exterior que compensa las pérdidas de energía debido a la resistencia óhmica de la bobina. Es aquí donde se necesita de un amplificador, no es necesario una gran amplificación, es suficiente con que sea capaz de reintegrar al circuito resonante inductor-condensador las pequeñas pérdidas que se producen durante su funcionamiento, consiguiendo así que el circuito oscile de forma indefinida mientras que la fuente de energía "extra" sea capaz de suministrarle energía [14].

2.2.1 Tipos de osciladores

El espectro de frecuencias en el que se emplean los osciladores para producir señales de ondas senoidales es muy amplio (desde menos que 1Hz hasta cientos de gigahertz). Sin embargo, ningún diseño único de oscilador es práctico para producir señales a través del rango completo. En lugar de ello, se emplea una variedad de diseños, cada uno de los cuales genera salidas de ondas senoidales con más eficiencia en varias partes del espectro de frecuencia.
Los osciladores que emplean circuitos de inductancia-capacitancia (LC) como elementos oscilatorios resuenan a una frecuencia en particular (producen una variación senoidal natural). Estos osciladores son muy populares para producir salidas de alta frecuencia (RF) (por ejemplo de 10kHz a 100MHz). Los osciladores LC que se usan más frecuentemente son los de Hartley y de Colpitts. Sin embargo los osciladores LC en conjunto no se adaptan bien para producir salidas de ondas senoidales de baja frecuencia. Por lo tanto, es usual emplear osciladores de resistencia-capacitancia (RC) para generar ondas senoidales de baja frecuencia (desde 1 Hz hasta aproximadamente 1 MHz).

Los dos osciladores RC más comunes son el puente de Wien y el de corrimiento de fase. El diseño de puente de Wien se emplea en casi todos los osciladores que producen señales en el rango de audiofrecuencias (20 a 20,000 Hz). Este tipo de oscilador es de diseño sencillo, tamaño compacto y notablemente estable en su frecuencia de salida. Además, su salida está relativamente libre de distorsión. Sin embargo, la salida de frecuencia máxima de los puentes de Wien típicos, es de aproximadamente 1MHz.

Los osciladores de corrimiento de fase también emplean un circuito sencillo y producen salidas de ondas senoidales que están bastante libres de distorsión. La ventaja principal sobre los osciladores de Wien es que tienen un rango mayor de frecuencia. Pero también tienen la desventaja de no ser tan estables en su frecuencia como los osciladores de puente de Wien.

Otros osciladores menos comunes son el oscilador de cristal y el oscilador de resistencia negativa. Los osciladores de cristal emplean un cristal piezoeléctrico para generar una señal senoidal de frecuencia constante. La frecuencia de salida es extremadamente estable con el tiempo, pero no se puede sintonizar o ajustar. Por lo tanto, los osciladores de cristal se emplean solo en aplicaciones que necesitan una frecuencia fija de alta estabilidad. Los osciladores de resistencia negativa se emplean principalmente para producir señales de muy alta frecuencia. La tabla 2.1 resume los rangos de frecuencia de los tipos más comunes de osciladores [15].

Tabla 2.1. Rangos de frecuencias para distintos tipos de osciladores.

Tipo	**Rangos aproximados de frecuencia**
Puente de Wien (RC)	1 Hz a 1 MHz
Corrimiento de fase (RC)	1 Hz a 10MHz
Hartley (LC)	10 kHz a 100MHz
Colpitts (LC)	10 kHz a 100MHz
Resistencia negativa	> 100MHz
Cristal	Frecuencia fija

2.2.2 Criterio de Oscilación

Para determinar el criterio de oscilación se puede asimilar el oscilador a un circuito con retroalimentación positiva, como el que se muestra en la figura 2.3, x_i y x_o son las señales de entrada y salida, mientras que x_r y x_e son la señal de realimentación y la señal de error respectivamente.

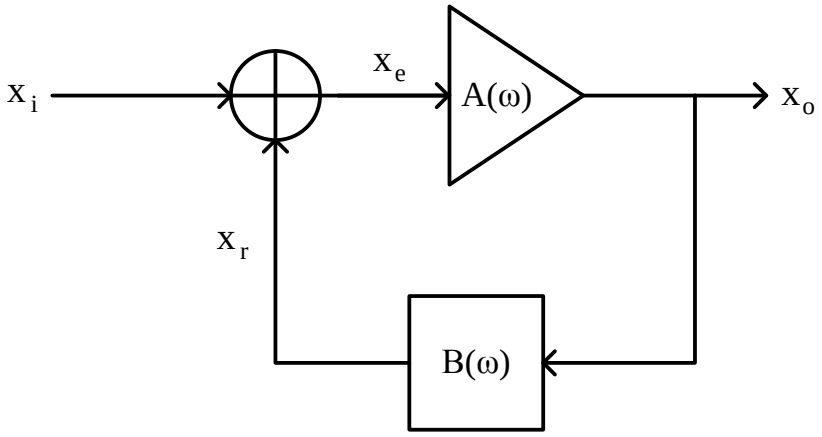

Figura 2.3. Diagrama de bloques de un circuito lineal con retroalimentación positiva.

A es la ganancia del amplificador inicial ó ganancia en lazo abierto, β es el factor de realimentación y $A\beta$ es la ganancia de lazo. Todos son números complejos cuyo módulo y fase varían con la frecuencia angular, ω. La ganancia del circuito realimentado es

$$\frac{x_o}{x_i} = \frac{A}{1 - A\beta}. \tag{1}$$

El comportamiento del circuito se puede predecir conociendo el módulo $|A\beta|$ y la fase $\phi_{A\beta}$, de la ganancia de lazo.

- Si $|A\beta| < 1$, el circuito es estable sea cual sea el valor de $\phi_{A\beta}$.
- Si a una frecuencia determinada $A\beta = 1$, es decir $|A\beta| = 1$ y $\phi_{A\beta} = 0$, cualquier oscilación presente en la entrada a esa frecuencia se mantiene indefinidamente, a la misma amplitud.
- Si a una frecuencia determinada $A\beta > 1$, es decir $|A\beta| > 1$ y $\phi_{A\beta} = 0$, cualquier oscilación presente en la entrada a esa frecuencia se amplifica indefinidamente hasta que la saturación del amplificador lo devuelve a la condición anterior. Como la saturación es un fenómeno no lineal, esto mismo provoca la aparición de armónicos.

Si el circuito tiene $A\beta > 1$ se puede prescindir de la señal de entrada puesto que el ruido, siempre presente, contiene componentes a todas las frecuencias. La componente de ruido a la frecuencia en la que se cumpla esta condición, conocida como condición de arranque, se amplifica indefinidamente hasta la saturación del amplificador o hasta que un circuito auxiliar consiga que para esa frecuencia $A\beta = 1$. A partir de entonces la amplitud de la oscilación se mantiene, por eso

a la condición $A\beta = 1$ se la denomina condición de mantenimiento. Estas condiciones para que un circuito oscile se conocen como criterio de Barkhausen.

El circuito externo para establecer la condición de mantenimiento mide la amplitud de la oscilación y varía la ganancia del amplificador de forma inversamente proporcional. Si se emplea, se obtiene un tono más puro, con menos armónicos, que si se deja a la saturación del amplificador la limitación de la amplitud. Sin embargo, la pureza de la oscilación depende de otros factores adicionales.
En general el funcionamiento del oscilador es no lineal, no obstante la condición de arranque se puede estudiar con un modelo lineal del amplificador porque trabaja con señales muy pequeñas.

2.2.3 Análisis de condiciones de oscilación

El método de análisis consiste primero en identificar el lazo de realimentación y el sentido del lazo. Después el lazo debe abrirse en un punto cualquiera, situar al inicio un generador de tensión auxiliar, V_x, y al final un impedancia, Z_{in}, equivalente a la impedancia de entrada que se ve desde el inicio, tal como se muestra en la figura 4.

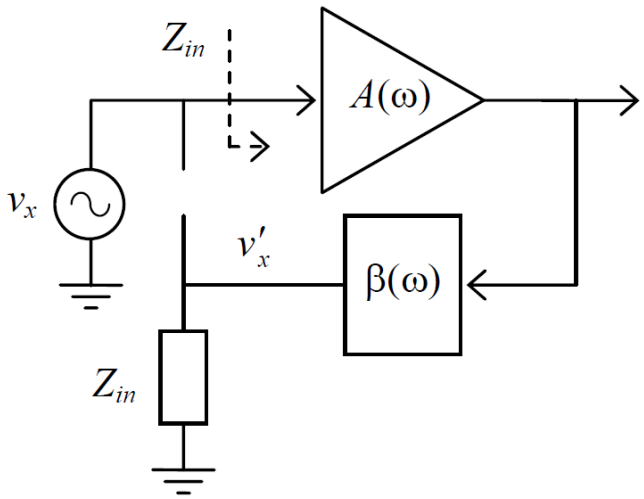

Figura 2.4. Ruptura del lazo de realimentación para calcular la ganancia de lazo.

A continuación debemos calcular la señal que llega al final del lazo, v'_x, y la ganancia de lazo como:

$$A\beta = \frac{v'_x}{v_x}, \tag{2}$$

Finalmente, aplicando el criterio de Barkhausen: $\phi_{A\beta} = 0$ y $A\beta > 1$, obtendremos la frecuencia de oscilación y la condición de arranque.
La ganancia de lazo, Aβ, es independiente del punto en que rompamos el lazo, pero la dificultad de su cálculo a menudo no lo es. Por ello se elige un punto en que $Z_{in} = \infty$ puede simplificar mucho este cálculo. Alternativamente, se puede escoger un punto en que la impedancia de salida al final del lazo es nula, de forma que el valor de Z_{in} sea irrelevante.

2.2.4 Respuesta libre de oscilador

Es posible asimilar un oscilador a un circuito RLC. Para explicarlo debemos calcular la respuesta libre del circuito que se ha representado en la figura 2.5.

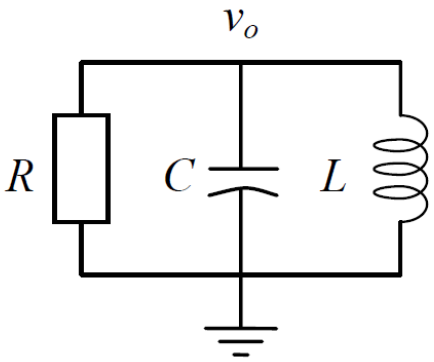

Figura 2.5. Circuito RLC paralelo.

Para obtener una ecuación que defina el comportamiento del circuito se hace un análisis en el tiempo aplicando la ley de corrientes de Kirchhoff:

$$\frac{v_0}{R} + C\frac{dv_0}{dt} + \frac{1}{L}\int v_0 dt = 0, \tag{3}$$

derivando y multiplicando por L la ecuación (3), se obtiene:

$$LC\frac{d^2v_0}{dt} + \frac{L}{R}\frac{dv_0}{dt} + v_0 = 0.$$

La solución de la ecuación diferencial (4) se obtiene al resolver su ecuación característica asociada:

$$LCs^2 + \frac{L}{R}s + 1 = 0, \tag{5}$$

obteniendo las raíces de la ecuación (5) se tiene:

$$s_{1,2} = -\frac{1}{2RC}(1 \pm \sqrt{1 - \frac{4R_2C}{L}}, \tag{6}$$

de la ecuación (6) se pueden obtener cinco posibles soluciones para diferentes condiciones tal y como lo muestra la tabla 2.2.

Tabla 2.2. Posibles soluciones del circuito RLC.

Condición	Raíces	Comportamiento de $v_o(t)$
1) $\boldsymbol{R > 0\ y\ R^2 < \frac{L}{4C}}$	$s_1 = -a_1, s_2 = -a_2$	$v_0 = A_1 e^{-a_1 t} + A_2 e^{-a_2 t}$
2) $\boldsymbol{R > 0\ y\ R^2 > \frac{L}{4C}}$	$s_{1,2} = -a \pm j\omega_0$	$v_0 = Acos(\omega_0 t + \varphi_0) e^{-at}$
3) $\boldsymbol{R = \pm\infty}$	$s_{1,2} = \pm j\omega_0$	$v_0 = Acos(\omega_0 t + \varphi_0)$
4) $\boldsymbol{R < 0\ y\ R^2 > \frac{L}{4C}}$	$s_{1,2} = a \pm j\omega_0$	$v_0 = Acos(\omega_0 t + \varphi_0) e^{at}$
5) $\boldsymbol{R < 0\ y\ R^2 < \frac{L}{4C}}$	$s_1 = a_1, s_2 = a_2$	$v_0 = A_1 e^{a_1 t} + A_2 e^{a_2 t}$

La figura 2.6 muestra las señales del comportamiento en el tiempo para cada condición descrita en la tabla 2.2.

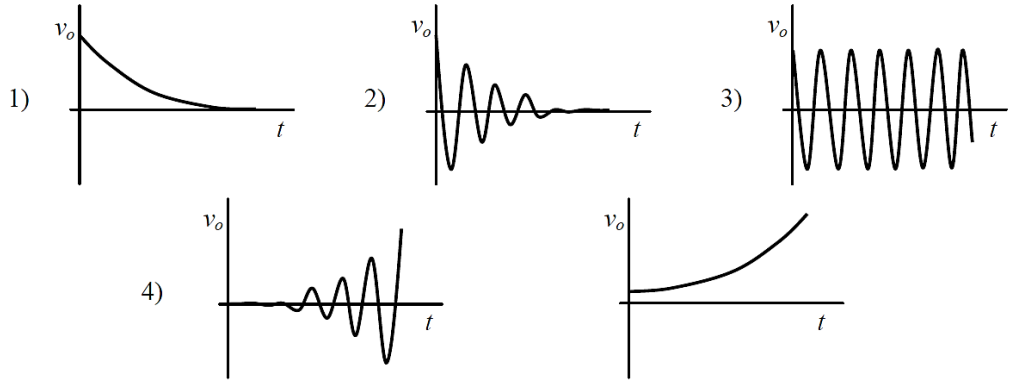

Figura 2.6. Comportamiento de v_o en el tiempo para cada condición.

Las soluciones 1) y 2) son estables, son las habituales en circuitos RLC pasivos. La solución 3) corresponde a oscilaciones de amplitud constante que se mantienen indefinidamente. Las soluciones 4) y 5) son inestables, la 4) corresponde a oscilaciones de amplitud creciente y la 5) corresponde a una tensión que crece continuamente.

Del resultado anterior se deduce que un oscilador puede entenderse como un circuito LC asociado a una resistencia negativa. Dicha resistencia es necesaria para compensar la energía disipada en las resistencias parásitas asociadas al condensador y a la bobina, principalmente a esta última, en cada oscilación. Inicialmente la resistencia equivalente total debe ser negativa, para obtener oscilaciones de amplitud creciente, es la condición de arranque. Después la amplitud del oscilador se estabiliza cuando la resistencia equivalente es infinita y en ese caso la frecuencia de oscilación es la frecuencia de resonancia del circuito LC dada por:

$$\omega_{osc} = \frac{1}{\sqrt{LC}}. \tag{7}$$

2.2.5 Osciladores de Colpitts y Hartley.

El oscilador de Colpitts mostrado en la figura 2.7(a), es uno de los más utilizados. El circuito de retroalimentación consta de *L*, C_1 y C_2. El oscilador de Colpitts se utiliza en circuitos de muy alta frecuencia. Su análisis produce:

La frecuencia de oscilación, que está dada por:

$$\omega_0{}^2 = \frac{1}{h_{ie}R_0C_1C_2} + \frac{C_1 + C_2}{LC_1C_2}, \tag{8}$$

donde R_0 es la carga del oscilador o impedancia de salida. La condición de magnitud genera:

$$h_{fe} \geq \frac{C_2}{C_1} + \frac{h_{ie}}{R_0} \cdot \frac{C_1}{C_2}, \tag{9}$$

si R_0 es suficientemente grande, las ecuaciones (8) y (9) se reducen a:

$$w_0 \approx \sqrt{\frac{C_1 + C_2}{LC_1C_2}}, \tag{10}$$

$$h_{fe} > \frac{C_2}{C_1}. \tag{11}$$

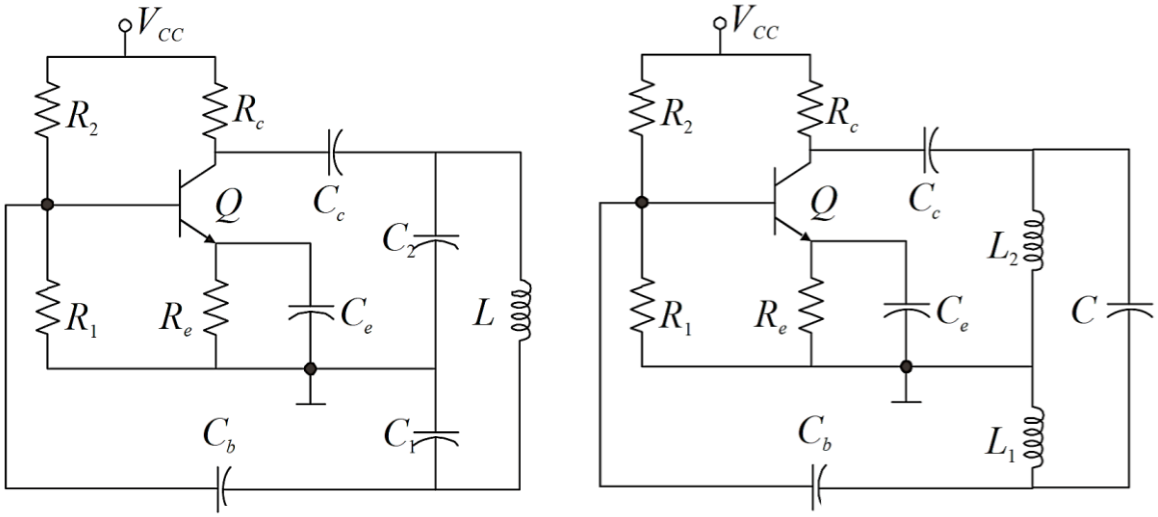

Figura 2.7. (a) Oscilador de Colpitts. (b) Oscilador de Hartley.

El oscilador de Hartley, que se ilustra en la figura 2.7(b), es prácticamente idéntico al de Colpittsm salvo que las capacitancias C_1 y C_2 se sustituyen por los inductores L_1 y L_2, y la inductancia L en los circuitos sintonizados se reemplaza por la capacitancia C. Si se analiza el circuito oscilador de Hartley se obtiene:

La frecuencia de oscilación, que está dada por:

$$\omega_0{}^2 = \frac{R_0 h_{ie}}{h_{ie} R_0 C(L_1 + L_2) + L_1 L_2}, \tag{12}$$

y el requisito mínimo de h_{fe} del transistor es:

$$h_{fe} \geq \frac{h_{ie} L_2}{R_0 L_1} + \frac{L_1}{L_2}. \tag{13}$$

En estas expresiones se supone que los amplificadores de los osciladores de Colpitts y de Hartley se operan en modo de clase A. Para lograr una estabilidad de frecuencia adecuada, se deben utilizar circuitos con un elevado factor de calidad Q [16].

2.2.6 Osciladores de resistencia negativa

Un dispositivo que convierte energía eléctrica en calor o energía mecánica se puede representar en un circuito mediante una resistencia positiva equivalente. Por otra parte, uno que convierte otras formas de energía eléctrica, se puede representar mediante una resistencia negativa. En un circuito elemental de corriente continua, que contenga una batería y un resistor de carga, por ejemplo, la resistencia que se calcularía si se aplicara la ley de Ohm en las terminales de la batería (con el cuidado adecuado a las convenciones de signo) es negativa. Esto quiere decir que la batería es una fuente de energía eléctrica y no un sumidero.

Los diodos túnel y Gunn, los transistores de unión única y ciertas combinaciones de dos o más transistores son capaces de absorber potencia de corriente directa. y convertir parte de ella en salida sinusoidal de frecuencia elevadas. Estos dispositivos exhiben así una resistencia negativa dentro de cierto ancho de banda infinito.

La figura 2.8 ilustra la curva característica de salida con resistencia negativa controlada en voltaje (estable en voltaje) para un diodo activo.

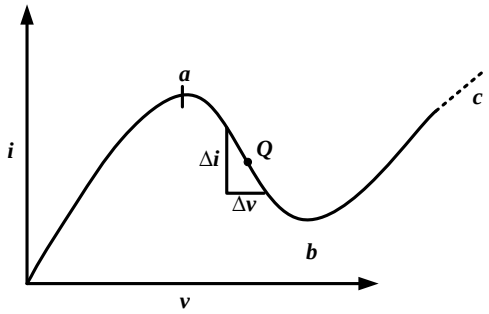

Figura 2.8. Curva característica de resistencia negativa con voltaje estable.

La resistencia estática del dispositivo es *v/i* y siempre es positiva, por lo que el diodo siempre absorbe potencia en c.d. Por otra parte, la resistencia incremental o de señal débil $r_n=dv/di$, es positiva en las regiones *0 – a* y *b – c*, pero negativa en la región *a – b*. Si el dispositivo se polariza en un punto *Q*, cualquier señal de c.a. aplicada al diodo vería una resistencia negativa y podrían generarse oscilaciones.

Para diseñar un oscilador se conecta un circuito paralelo resonante con componentes *L*, *C* y R_t como el mostrado anteriormente en la figura 2.5, este circuito se conecta en paralelo con la resistencia negativa, como en la figura 2.9. El ruido térmico hará que arranquen las oscilaciones. El punto de operación oscilará alrededor de *Q* y la resistencia incremental instantánea del diodo puede llegar a ser positiva en parte de cada ciclo, cuando la amplitud de la oscilación llegue a ser lo suficientemente grande.
Sea R_n la resistencia negativa de diodo promediada en un ciclo. Si el voltaje rms en el diodo es *V*, la carga absorbe V^2/R_t y el diodo entrega $V^2/|R_t|$ watts. Si la potencia entregada es mayor que la absorbida, la amplitud de la oscilación crecerá. En cada ciclo, esta amplitud excitará al diodo aún más dentro de las regiones alrededor de los puntos *a* y *b*, donde $|R_n|$ llega a ser sumamente grande. Esto hará que $|R_n|$ crezca y decrezca la potencia entregada, hasta que se establezca el equilibrio.

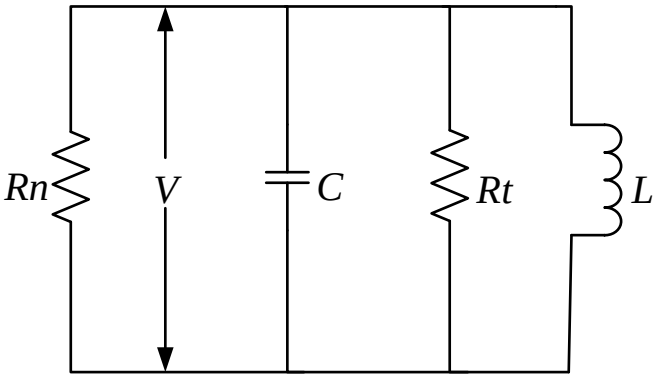

Figura 2.9. Oscilador elemental a resistencia negativa.

Matemáticamente [17], el voltaje instantáneo *v* a través de los elementos en paralelo de la figura 2.9, está dado por:

$$v(t) = e^{-at}(B_1 cos\omega_d t + B_2 sen\omega_d t), \tag{14}$$

donde B_1 y B_2 son constantes;

$$\alpha = \frac{1}{2RC}, \tag{15}$$

$$\omega_o = \frac{1}{\sqrt{LC}}, \tag{16}$$

$$\omega_d = \sqrt{\omega_o{}^2 - \alpha^2}. \tag{17}$$

La resistencia efectiva de R_t y R_n en paralelo es R, donde:

$$R = \frac{R_n R_t}{R_n + R_t} = \frac{-|R_n| R_t}{-|R_n| + R_t}. \tag{18}$$

Para R_t mayor que $|R_n|$, R y α son negativas y la amplitud de *v(t)* crece con el tiempo.
Para R_t menor que $|R_n|$, R y α son positivas y decaen las oscilaciones. Cuando R_t es igual a $|R_n|$, estas ecuaciones tienden a infinito, aunque el circuito mantiene un voltaje sinusoidal estacionario.

Como la resistencia de entrada del circuito tanque decrece a ambos lados de la frecuencia de resonancia, el circuito de la figura 2.9 tiene la mayor posibilidad de oscilar en la frecuencia de resonancia de *L* y *C*. En la práctica, las oscilaciones pueden presentarse en cualquier parte de una banda de frecuencias determinada por el *Q* del circuito tanque y, por esta razón, el espectro de salida de un oscilador a diodo de resistencia negativa no es lo suficientemente limpio para usarse en frecuencias de radio.

2.2.7 Ejemplo: Oscilador diodo túnel

Este oscilador hace uso del diodo túnel para producir oscilaciones. El circuito practico de un oscilador diodo túnel es mostrado en la figura 2.10.

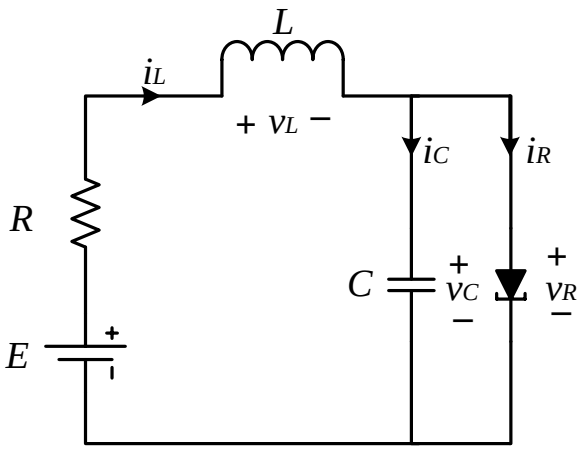

Figura 2.10. Circuito oscilador con diodo túnel.

La relación constitutiva del diodo es $i_R = k(v_R)$.
Usando las leyes de corrientes y voltajes de Kirchhoff, además de tomar como variables de estado $x_1 = v_C$ y $x_2 = i_L$, que son el voltaje del capacitor y corriente del inductor respectivamente, así el circuito de la figura 2.10 se describe con el sistema de ecuaciones dinámico:

$$\dot{x_1} = \frac{1}{C}(-h(x_1) + x_2), \tag{19}$$

$$\dot{x_2} = \frac{1}{L}(-x_1 + Rx_2 + v), \tag{20}$$

donde $i = h(v)$ y $v = E$ definen la curva característica de voltaje-corriente del diodo tunel, la cual se muestra en la figura 2.8. Los puntos de equilibrio son las raíces de la ecuación:

$$h(x_1) = \frac{E}{R} - \frac{1}{R}x_1, \tag{21}$$

lo cual indica que según la relación E/R puede haber uno o tres puntos de equilibrio también mostrados en la figura 2.8.

2.3 TEORÍA DEL CAOS

A grandes rasgos, el caos es un comportamiento imprevisible a largo plazo, que viene de un sistema dinámico determinista debido a la sensibilidad de las condiciones iniciales. La teoría del caos permite comprender fenómenos de la naturaleza, las extrañas formas que esta exhibe y los patrones de conducta a los que obedece. Más allá de esto, aparece también como una herramienta valiosa para entender el comportamiento de la conducta humana y social, los fenómenos económicos, así como la evolución de la tecnología y de la actividad industrial. Ante estas perspectivas no parece estar lejos el utilizar esta teoría como modelo para explicar la conducta de los sistemas reales [18].

La teoría del caos aplicada en sistemas dinámicos no lineales, está lejos de ser la panacea que algunos investigadores imaginaron en sus comienzos. Sin embargo, sus conceptos pueden aplicarse a problemas bien dirigidos por el uso de métodos matemáticos, que son rigurosamente escogidos y adaptados a los sistemas bajo estudio.

Para que un sistema dinámico simple logre tener un comportamiento imprevisible, basta incluir en sus entradas términos aleatorios. Las primeras ideas de lo que hoy se denomina "procesos estocásticos o aleatorios" aparecen a final del siglo XIX e inicios del XX y los primeros trabajos estuvieron dedicados a estudiar fenómenos físicos y tecnológicos particulares, dado que en aquel momento, la teoría clásica de probabilidades no incluía el estudio de esquemas aleatorios dependientes del tiempo. Un problema de este tipo, que incluye un término "de ruido", es común en la naturaleza y abre un campo de estudio extenso que puede ser considerado como otro problema fundamental de solución complicada a partir de los análisis de las ecuaciones dinámicas. Ese problema había sido olvidado por la Mecánica Clásica que se preocupaba solamente de los modelos llamados "deterministas". En general, la aleatoriedad es tomada como algo indeseable [19].

El movimiento caótico no es un fenómeno raro. Para enunciar cuáles son las condiciones necesarias que tiene que satisfacer un sistema para que su comportamiento sea caótico es conveniente describir

su movimiento mediante sistemas dinámicos. Las condiciones necesarias para la existencia de un movimiento caótico son:

1. El sistema debe tener al menos tres variables independientes.
2. Las ecuaciones de movimiento deben contener al menos un término no lineal que acople algunas de las variables.

El hecho de que solamente se requieren tres variables para el caos fue sorprendente cuando se descubrió. La condición de no linealidad es quizás menos sorprendente [20]. Las soluciones de las ecuaciones diferenciales lineales se pueden expresar siempre como una superposición de funciones armónicas una vez que la solución de la particular haya desaparecido. El término no lineal hace inestable a estas soluciones armónicas para ciertos valores de los parámetros. Y son varios los ejemplos de circuitos electrónicos utilizados para el estudio de Caos; el sistema de Lorenz, Rössler, Chua, entre otros.

Un campo tan complejo como el de la criptología sólo puede beneficiarse de la adición de otros medios de investigación, sobre esta base, se piensa en la colocación de los conceptos de caos a disposición de la criptografía algorítmica, para poder generar secuencias pseudo-aleatorias, a través de los modelos matemáticos adecuados para mantener segura la información, y éste así cumpla las condiciones del caos.

Los circuitos se presentan como una herramienta de una gran utilidad para estudiar una gran variedad de procesos, actuando como complemento entre el experimento en sí y la simulación numérica por computadora. Entre las ventajas que ofrece la simulación con circuitos se encuentran tanto el alto grado de desarrollo de componentes electrónicos como el bajo costo de los dispositivos, lo importante es que se pueden estudiar dichos circuitos, obteniendo su modelo matemático para diseños más eficientes, en este trabajo de tesis se a partir del modelo matemático se realiza un diseño digital del circuito o del sistema caótico de estudio.

2.4 OSCILADOR CAÓTICO BASADO EN SERIES DE FUNCIONES SATURADAS

Un oscilador caótico basado en series de funciones saturadas (SNLF por sus siglas en inglés: Satured Nonlinear Function) se describe por el siguiente sistema de ecuaciones [21]:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -ax - by - cz + d_1 f(x; k, \alpha, p, q), \end{aligned} \tag{22}$$

donde *a*, *b*, *c* y d_1 son coeficientes reales positivos con valores entre 0 y 1, el sistema además está compuesto por tres variables de estado *x*, *y* y *z* las cuales dependen una de la otra, por lo que son necesarias condiciones iniciales.

En la tercera ecuación del sistema de ecuaciones (22) se puede observar que está incluida la función no lineal saturada (SNLF) *f(x; k, h, p, q)*, esta parte es la que causa la no linealidad del sistema. El

propósito de esta función es retroalimentar y mantener oscilando el sistema. La SNLF se puede describir a través de una aproximación por funciones linealizadas a tramos (PWL por sus siglas en inglés: piece-wise linear), y se desarrolla con base al número de enrollamientos que se desean generar, tomando en cuenta que a medida que el número de enrollamientos aumenta, el número de pendientes y niveles de saturación también. La aproximación PWL de la SNLF está dada por:

$$f(x;k,\alpha,h,p,q)=\sum_{i=-p}^{q} f_i(x;\alpha,h,k). \tag{23}$$

La función saturada no lineal más básica tiene dos niveles de saturación, por lo que esta genera dos enrollamientos, la función es mostrada en la figura 2.11 y su descripción PWL está dada por:

$$f(x)=\begin{cases} k & x>\alpha \\ sx & -\alpha\le x\le\alpha \\ -k & x<-\alpha \end{cases}, \tag{24}$$

donde *k* es el nivel de saturación, *α* el punto de quiebre del nivel de saturación y *sx* la pendiente.

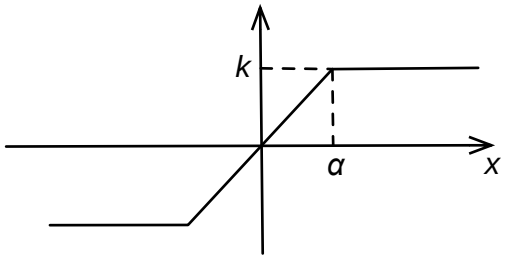

Figura 2.11. SNLF básica con dos niveles de saturación.

2.4.1 Diseño de multi-enrollamientos

La generación de múltiples enrollamientos de un oscilador caótico basado en series de funciones saturadas depende exclusivamente de la extensión del número de niveles de saturación *k* a partir de la función mostrada en la figura 2.11, para ello se describe cada función con sus respectivos niveles de saturación mediante funciones linealizadas a tramos (PWL), en las ecuaciones (25)-(28) se muestran las descripciones PWL para la generación de 3, 4, 5 y 6 enrollamientos respectivamente, en las figuras 2.12-2.16 se muestran las SNLF descritas anteriormente mediante PWL.

Para la descripción PWL de las SNLF mostradas en las figuras 2.12-2.16 se hace un análisis utilizando trigonometría básica, se definen por intervalos las función y se descompone en varias funciones:

$$f(x)=\begin{cases} (2q)k & x>qh+\alpha \\ s(x-h)+k & h-\alpha\le x\le h+\alpha \\ s(x+h)-k & -h-\alpha\le x\le -h+\alpha \\ 0 & -h+\alpha\le x\le h-\alpha \\ -(2q)k & x>-ph-\alpha \end{cases} \tag{25}$$

$$f(x)=\begin{cases}(2q+1)k & x>qh+\alpha\\ s(x+h)-2k & -h-\alpha\le x\le -h+\alpha\\ sx & -\alpha\le x\le \alpha\\ s(x-h)+2k & h-\alpha\le x\le h+\alpha\\ -k & -h+\alpha<x<-\alpha\\ k & \alpha<x<h-\alpha\\ -(2p+1)k & x<-ph-\alpha\end{cases} \tag{26}$$

$$f(x)=\begin{cases}(2q)k & x>qh+\alpha\\ s(x+3h)-3k & -(q+1)h-\alpha\le x\le -(q+1)h+\alpha\\ s(x+h)-k & -h-\alpha\le x\le -h+\alpha\\ 0 & -h+\alpha\le x\le h-\alpha\\ sx & -\alpha\le x\le \alpha\\ s(x-h)+k & h-\alpha\le x\le h+\alpha\\ s(x-3h)+3k & (q+1)h-\alpha<x<(q+1)h+\alpha\\ -2k & -(q+1)h+\alpha\le x\le -h-\\ 2k & h+\alpha<x<(q+1)h-\alpha\\ -(2p)k & x<-(p+1)h-\alpha\end{cases} \tag{27}$$

$$f(x)=\begin{cases}(2q+1)k & x>qh+\alpha\\ s(x-h)+2k & h-\alpha\le x\le h+\alpha\\ s(x-2h)+4k & qh-\alpha\le x\le qh+\alpha\\ sx & -\alpha\le x\le \alpha\\ s(x+2h)-4k & -qh-\alpha\le x\le -qh+\alpha\\ s(x+h)-2k & -h-\alpha\le x\le -h+\alpha\\ -3k & -qh+\alpha<x<-h-\alpha\\ -k & h-\alpha<x<-\alpha\\ k & \alpha<x<h-\alpha\\ 3k & h+\alpha<x<qh-\alpha\\ -2p+1k & x<-ph-\alpha\end{cases} \tag{28}$$

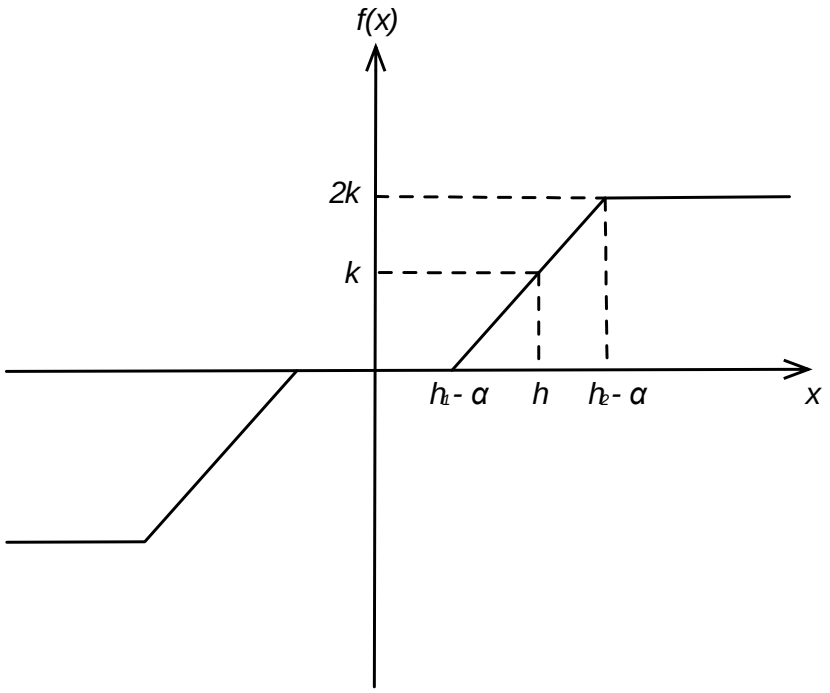

Figura 2.12. SNLF básica con tres niveles de saturación.

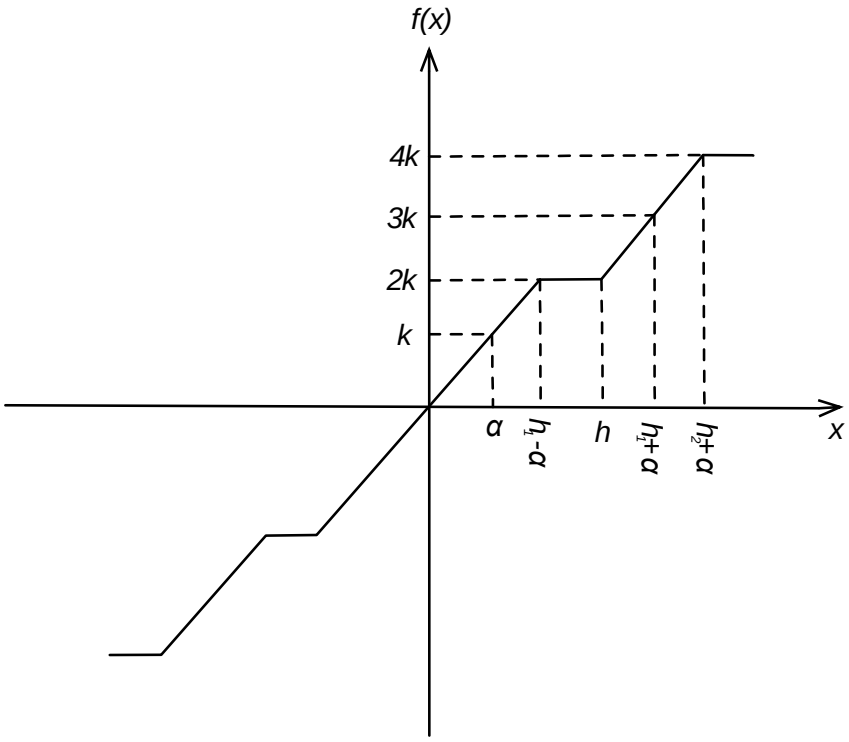

Figura 2.13. SNLF básica con cuatro niveles de saturación.

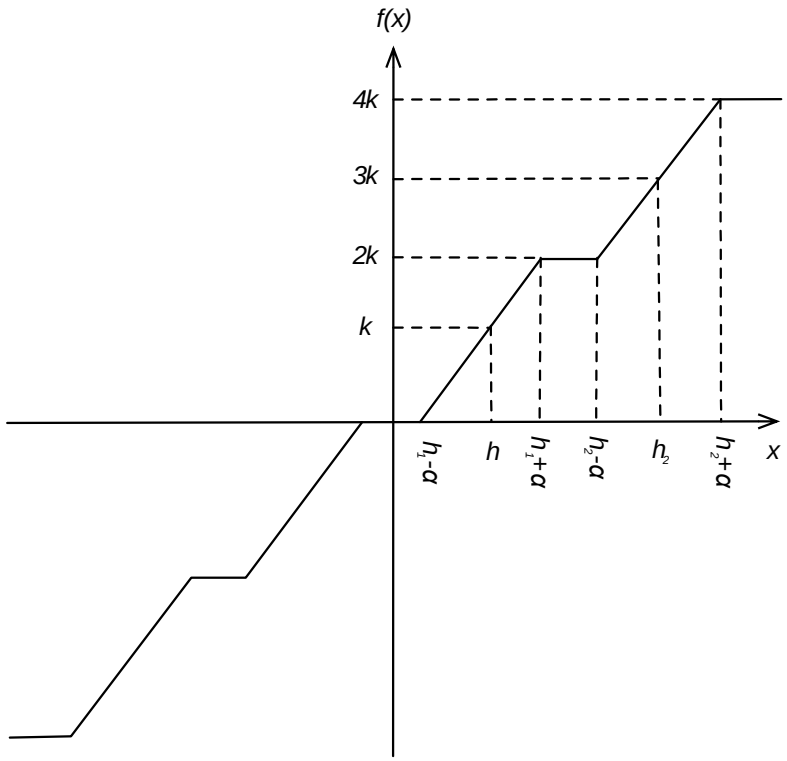

Figura 2.14. SNLF básica con cinco niveles de saturación.

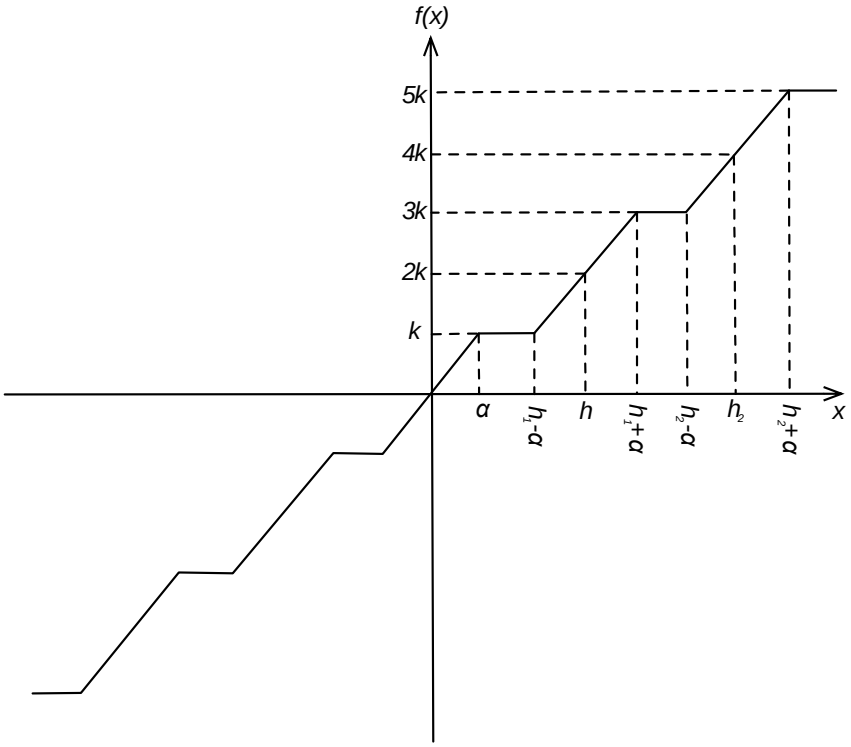

Figura 2.15. SNLF básica con seis niveles de saturación.

2.5 OSCILADOR CAÓTICO BASADO EN EL CIRCUITO DE CHUA

El circuito de Chua es uno de los modelos más populares que exhiben caos debido a su robustez, facilidad de implementación, su simple no linealidad estática y sus diversas características de bifurcación lo que contribuyen a su éxito entre teóricos y diseñadores de la teoría del caos. Por la riqueza en cuanto a su comportamiento, el circuito Chua ha sido y es objeto de mucha investigación científica, convirtiéndose en un paradigma universal para el caos. El circuito está compuesto por dos partes: 1) representación del comportamiento típico de un oscilador con respuesta amortiguada (dos condensadores, una bobina y una resistencia) y 2) representación de la no linealidad del circuito, a este elemento se le denomina diodo de Chua. Este elemento actúa como la fuente de energía de todo el circuito, su propósito es retroalimentarlo y mantenerlo oscilando [22].

En la figura 2.16 se muestra el circuito tipo Chua, donde (L y C_2) actúan como tanque resonante, (R y C_1) como filtro pasa-bajas y añadiéndole en paralelo el elemento no lineal, el diodo de Chua.

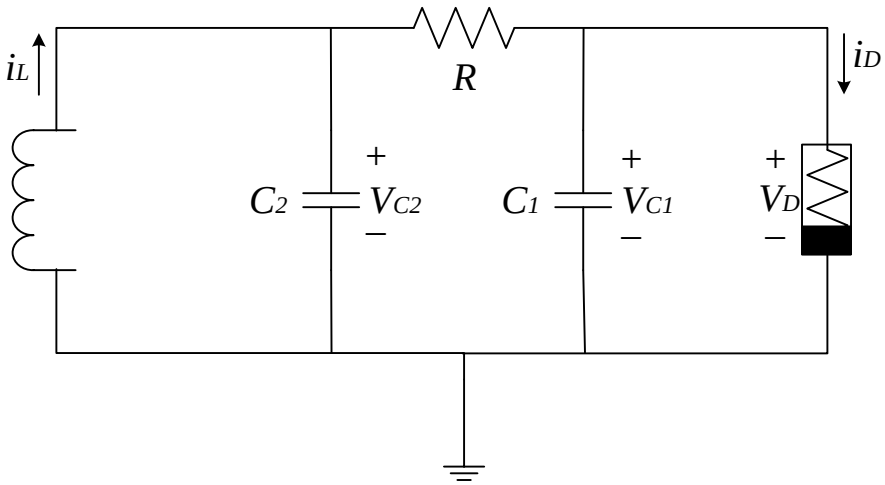

Figura 2.16. Circuito Oscilador de Chua.

Aplicando las leyes de Kirchoff al circuito de la figura 2.16 se obtienen las ecuaciones que describen el comportamiento dinámico:

$$C_1 \frac{dV_{C_1}}{dt} = G\left(V_{C_2} - V_{C_1}\right) - g\left(V_{C_1}\right), \tag{29}$$

$$C_2 \frac{dV_{C_2}}{dt} = G\left(V_{C_1} - V_{C_2}\right) + i_L, \tag{30}$$

$$L \frac{di_L}{dt} = -V_{C_2}, \tag{31}$$

donde la conductancia $G = \frac{1}{R}$, V_{C_1}, V_{C_2} , i_L son respectivamente los voltajes en el capacitor C_1 y C_2, la corriente en el inductor y

$$g\left(V_{C_1}\right) = G_p V_{C_1} + \frac{1}{2}\left(G_a - G_p\right)\left[\left|V_{C_1} + P\right| - \left|V_{C_1} - P\right|\right] \tag{32}$$

es la función de respuesta del elemento no lineal (el diodo de Chua). La ecuación (32) está representada por la curva voltaje–corriente de la figura 2.17, la cual está compuesta de tres rectas con pendiente negativa, siendo *Gp* y *Ga* las pendientes de cada segmento.

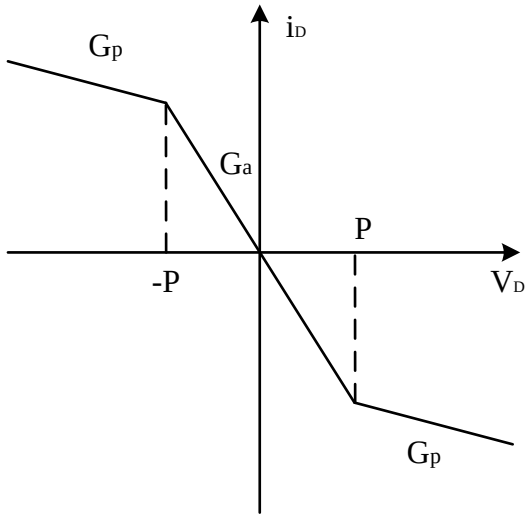

Figura 2.17. Curva característica voltaje-corriente del diodo de Chua.

Hasta ahora, se han presentado las ecuaciones del circuito de Chua (29)-(31) en términos de siete parámetros *L*, *C2*, *C1*, *G*, *P*, *Ga* y *Gp*. Se puede reducir el número de parámetros normalizando la resistencia no lineal, de tal manera que sus puntos de quiebre se encuentren ±1 V en lugar de ±*P* V.

Además se pueden escribir las ecuaciones de Chua en forma adimensional normalizada haciendo el siguiente cambio de variables: $x = \frac{V_{C_1}}{P}$, $y = \frac{V_{C_2}}{P}$, $z = \frac{I_L}{(PG)}$, $\tau = \frac{tC_2}{G}$. De esta manera se tiene:

$$\frac{dx}{d\tau} = \alpha(y - x - g(x)) \tag{33}$$

$$\frac{dy}{d\tau} = x - y - z \tag{34}$$

$$\frac{dz}{d\tau} = -\beta\, y \tag{35}$$

$$g(x) = m_1 x + \frac{1}{2}(m_0 - m_1)[|x+1| - |x-1|], \tag{36}$$

donde $\alpha = \frac{C_2}{C_1}$, $\beta = \frac{C_2}{(LG_2)}$, $m_0 = RG_a$ y $m_1 = RG_p$.

Con el cambio de notación, la función del diodo de Chua queda tal y como se muestra en la figura 2.18.

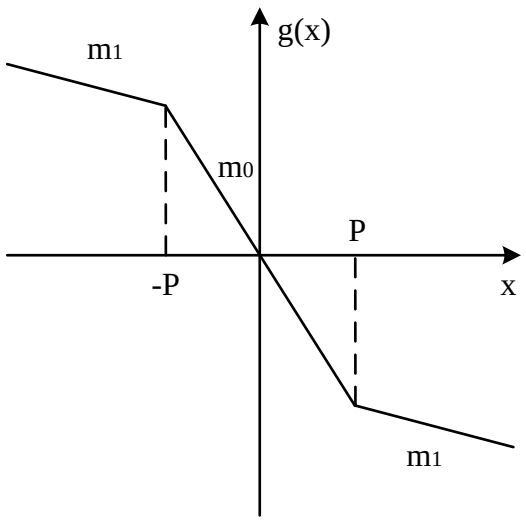

Figura 2.18. Función del diodo de Chua para dos enrollamientos.

Se muestra también su descripción mediante la funcion PWL la cual puede generar dos enrollamientos:

$$f(x) = \begin{cases} m_1 x + P(m_1 - m_0) & x < -P \\ m_0 x & -P \leq x \leq P \\ m_0 x + P(m_0 - m_1) & x > P \end{cases} \quad . \qquad (37)$$

2.5.1 Diseño de multi-enrollamientos

Tal y como se realiza con el oscilador caótico basado en SNLF, también se describe la extensión a multi-enrollamientos para el oscilador caótico basado en el circuito de Chua, en este caso depende del número de pendientes m_1 de la curva característica voltaje-corriente del diodo de Chua mostrada en la figura 2.18. En las ecuaciones (38)-(41) se muestran las descripciones PWL para la generación de 3, 4, 5 y 6 enrollamientos respectivamente, en las figuras 2.19-2.22 se muestran las curvas características del diodo de Chua descritas anteriormente mediante PWL.

El procedimiento realizado para describir mediante PWL las curvas del diodo de Chua es igual al seguido en el oscilador caótico basado en SNLF:

$$f(x) = \begin{cases} m_1 x + P(m_0 - m_1) + P_2(m_1 - m_0) & x < -P_2 \\ m_1 x + P(m_0 - m_1) & -P_2 \leq x_1 \leq -P \\ m_1 x & -P \leq x_1 \leq P \\ m_0 x + P(m_1 - m_0) & P \leq x_1 \leq P_2 \\ m_0 x + P(m_1 - m_0) + P_2(m_0 - m_1) & x > P_2 \end{cases} \qquad (38)$$

$$f(x)=\begin{cases} m_1x+P(m_1-m_0)+P_2(m_0-m_1)+P_3(m_1-m_0) & x<-P_3 \\ m_0x+P(m_1-m_0)+P_2(m_0-m_1) & -P_3\le x_1\le -P_2 \\ m_1x+P(m_1-m_0) & -P_2\le x_1\le -P \\ m_0x & -P\le x_1\le P \\ m_1x+P(m_0-m_1) & P\le x_1\le P_2 \\ m_0x+P(m_0-m_1)+P_2(m_1-m_0) & P_2\le x_1\le P_3 \\ m_1x+P(m_0-m_1)+P_2(m_1-m_0)+P_3(m_0-m_1) & x>P_3 \end{cases} \quad (39)$$

$$f(x)=\begin{cases} m_1x+P(m_0-m_1)+P_2(m_1-m_0)+P_3(m_0-m_1)+P_4(m_1-m_0) & x<-P_4 \\ m_0x+P(m_0-m_1)+P_2(m_1-m_0)+P_3(m_0-m_1) & -P_4\le x_1\le -P_3 \\ m_1x+P(m_0-m_1)+P_2(m_1-m_0) & -P_3\le x_1\le -P_2 \\ m_0x+P(m_0-m_1) & -P_2\le x_1\le -P \\ m_1x & -P\le x_1\le P \\ m_0x+P(m_1-m_0) & P\le x_1\le P_2 \\ m_1x+P(m_1-m_0)+P_2(m_0-m_1) & P_2\le x_1\le P_3 \\ m_0x+P(m_1-m_0)+P_2(m_0-m_1)+P_3(m_1-m_0) & P_3\le x_1\le P_4 \\ m_1x+P(m_1-m_0)+P_2(m_0-m_1)+P_3(m_1-m_0)+P_4(m_0-m_1) & x>P_4 \end{cases} \quad (40)$$

$$f(x)=\begin{cases} m_1x+P(m_1-m_0)+P_2(m_0-m_1)+P_3(m_1-m_0)+P_4(m_0-m_1)+P_5(m_1-m_0) & x<-P_5 \\ m_0x+P(m_1-m_0)+P_2(m_0-m_1)+P_3(m_1-m_0)+P_4(m_0-m_1) & -P_5\le x_1\le -P_4 \\ m_1x+P(m_1-m_0)+P_2(m_0-m_1)+P_3(m_1-m_0) & -P_4\le x_1\le -P_3 \\ m_0x+P(m_1-m_0)+P_2(m_0-m_1) & -P_3\le x_1\le -P_2 \\ m_1x+P(m_1-m_0) & -P_2\le x_1\le -P \\ m_0x & -P\le x_1\le P \\ m_1x+P(m_0-m_1) & P\le x_1\le P_2 \\ m_0x+P(m_0-m_1)+P_2(m_1-m_0) & P_2\le x_1\le P_3 \\ m_1x+P(m_0-m_1)+P_2(m_1-m_0)+P_3(m_0-m_1) & P_3\le x_1\le P_4 \\ m_0x+P(m_0-m_1)+P_2(m_1-m_0)+P_3(m_0-m_1)+P_4(m_1-m_0) & P_4\le x_1\le P_5 \\ m_1x+P(m_0-m_1)+P_2(m_1-m_0)+P_3(m_0-m_1)+P_4(m_1-m_0)+P_5(m_0-m_1) & x>P_5 \end{cases} \quad (41)$$

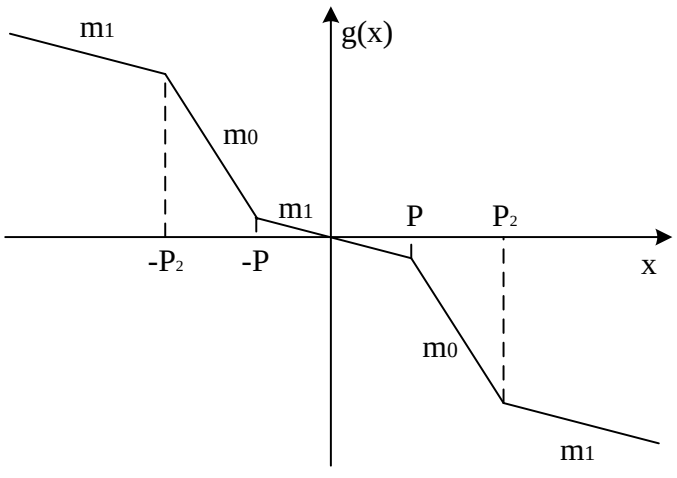

Figura 2.19. Curva característica del diodo de Chua para tres enrollamientos.

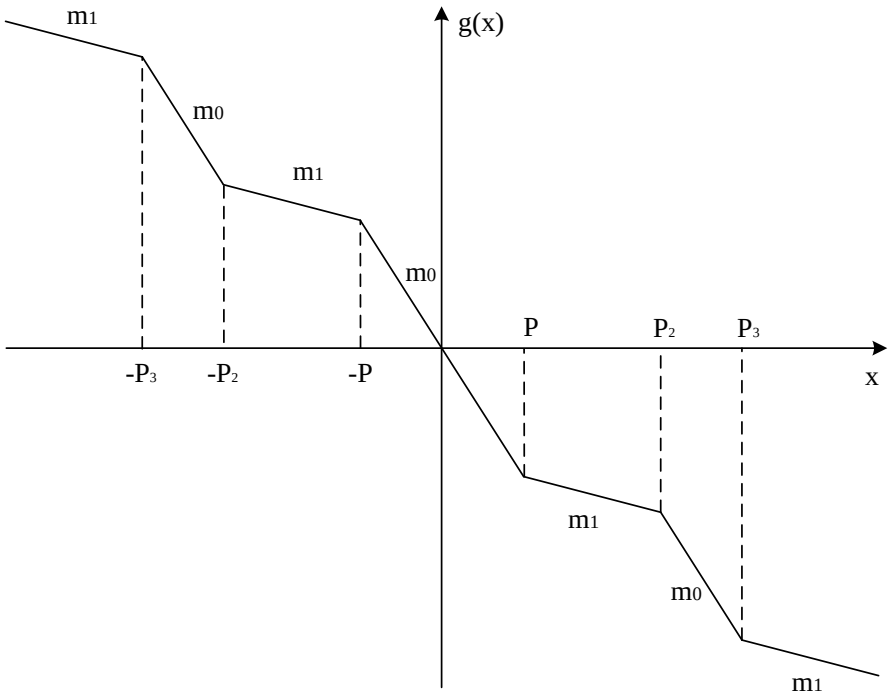

Figura 2.20. Curva característica del diodo de Chua para cuatro enrollamientos.

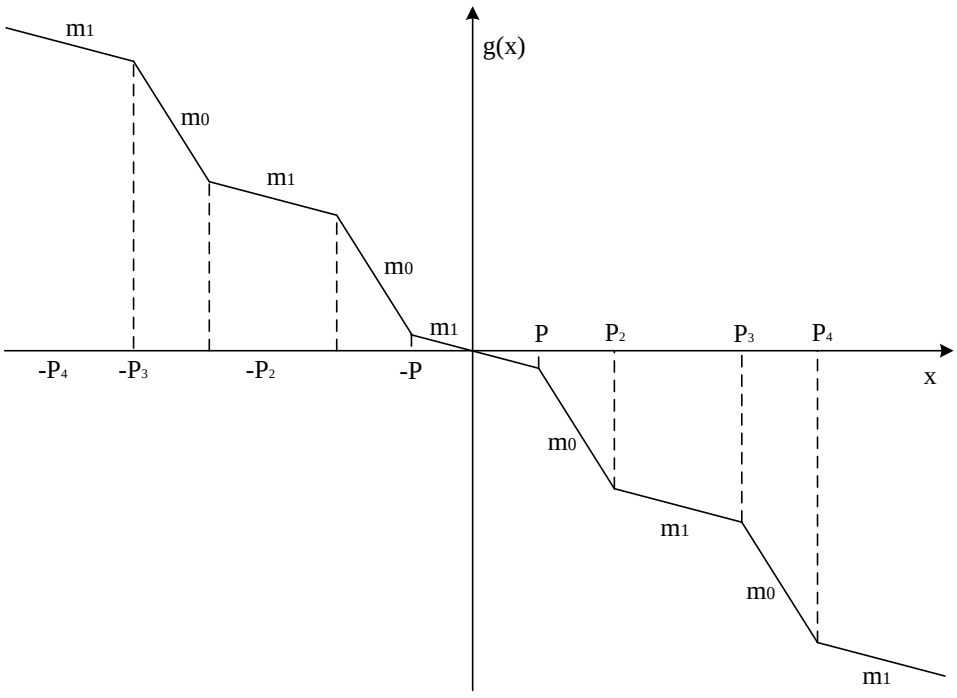

Figura 2.21. Curva característica del diodo de Chua para cinco enrollamientos.

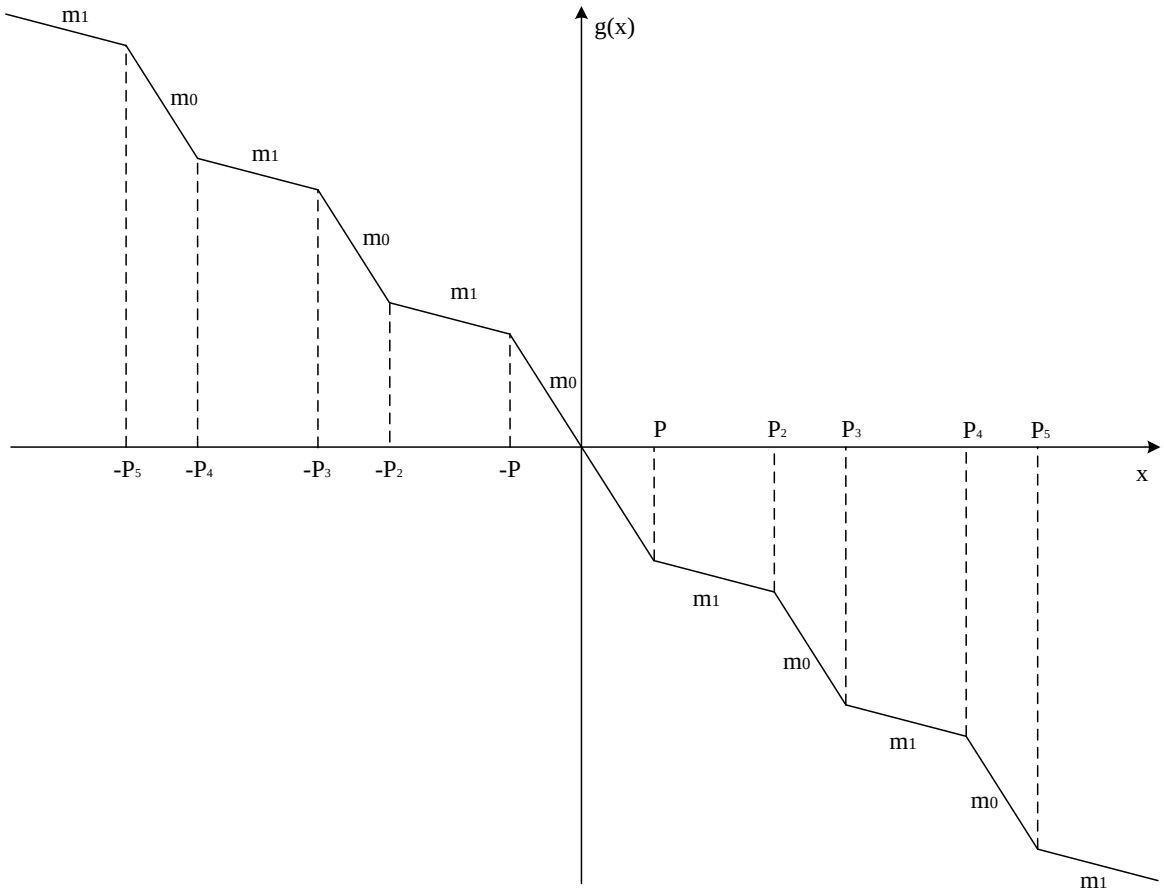

Figura 2.22. Curva característica del diodo de Chua para seis enrollamientos.

2.6 MÉTODOS NUMÉRICOS

Desde finales de la década de los años 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, por lo que muchos ingenieros seguían usando simples procedimientos analíticos en una buena parte de su trabajo. Vale la pena mencionar que la reciente evolución de computadoras personales de bajo costo ha permitido a mucha gente el acceso a los nuevos desarrollos computacionales. Además, existen diversas razones por las cuales se deben estudiar los métodos numéricos

Los métodos numéricos constituyen técnicas mediante las cuales es posible formular problemas matemáticos, de tal forma que puedan resolverse utilizando operaciones aritméticas. Aunque existen muchos tipos de métodos numéricos, éstos comparten una característica común: invariablemente requieren de un buen número de tediosos cálculos aritméticos. No es raro que con el desarrollo de computadoras digitales eficientes y rápidas, el papel de los métodos numéricos en la solución de problemas en ingeniería haya aumentado de forma considerable en los últimos años [23]:

2.6.1 Métodos numéricos en ecuaciones diferenciales

Las ecuaciones diferenciales son una de las herramientas matemáticas más importantes utilizadas en problemas de modelado en el ámbito físico-científico. Históricamente, las ecuaciones diferenciales (ED) se han originado en la química, la física y la ingeniería. Recientemente, también se han producido en la medicina, la biología, la antropología, y otras por el estilo. Las ecuaciones diferenciales ordinarias (EDO) surgen con frecuencia en el estudio de los sistemas físicos. Por desgracia muchos no se pueden resolver con exactitud. Esta es la razón por la cual la capacidad de aproximación numérica de los métodos numéricos es tan importante [24].

La solución numérica de EDOs es la técnica más importante que se ha desarrollado para sistemas dinámicos en tiempo continuo. Dado que la mayoría de las EDOs no tienen solución analítica, la integración numérica es la única forma de obtener información acerca de la trayectoria. Se han propuesto y utilizado muchos métodos diferentes en un intento de encontrar soluciones con precisión a varios tipos de EDOs. Todos estos, discretizan el sistema diferencial, para producir una ecuación diferencial o una ecuación que represente los estados del sistema [25].

Con la llegada de las computadoras, los métodos numéricos son ahora una forma cada vez más atractiva y eficiente de obtener soluciones aproximadas a las ecuaciones diferenciales que habían resultado hasta ahora difíciles, incluso imposible de resolver analíticamente.

En la actualidad, las computadoras y los métodos numéricos ofrecen una alternativa para los cálculos complicados. Al usar la potencia de la computadora se obtienen soluciones directamente, de esta manera se pueden aproximar los cálculos sin tener que recurrir a consideraciones de simplificación o a técnicas muy lentas. Aunque las soluciones analíticas aún son muy valiosas, tanto para resolver problemas como para brindar una mayor comprensión, los métodos numéricos representan opciones que aumentan, en forma considerable, la capacidad para enfrentar y resolver los problemas [26].

2.6.2 Método de Euler

El algoritmo de Euler hacia delante es el método más sencillo con el cual se pueden encontrar las soluciones de cada una de las ecuaciones diferenciales, discretizando al mismo tiempo el sistema ya que se obtienen datos por cada iteración.

La idea en la que se basa el método de Euler está basada en el significado geométrico de la derivada de una función en un punto dado. Supongamos que tuviéramos la curva solución de la ecuación diferencial y trazamos la recta tangente a la curva en el punto dado por la condición inicial x_0, véase la figura 2.23 [23].

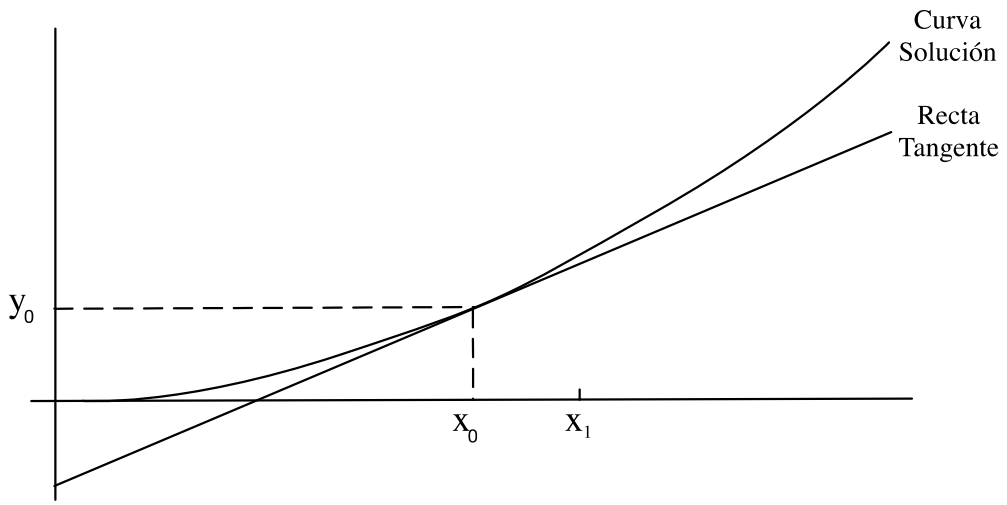

Figura 2.23. Recta tangente a la curva solución del punto x_0.

Debido a que la recta tangente se aproxima a la curva en valores cercanos al punto de tangencia, podemos tomar el valor de la recta tangente en el punto como una aproximación al valor deseado $y(x_1)$, así como se puede apreciar en la figura 2.16 [23].

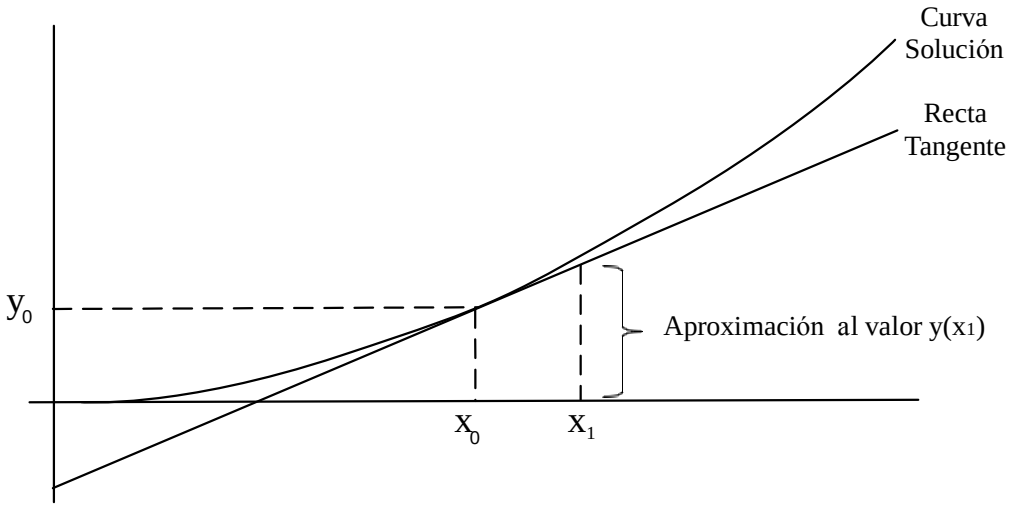

Figura 2.24. Recta tangente a la curva solución del punto x_1.

Así, se calcula la ecuación de la recta tangente a la curva solución de la ecuación diferencial dada en el punto (x_0, y_0). Se sabe que la ecuación de la recta es:

$$y = m \cdot (x - x_0) + y_0, \tag{42}$$

donde *m* es la pendiente. En este caso, sabemos que la pendiente de la recta tangente se calcula con la derivada:

$$m = y'|_{(x_0,y_0)} = f(x_0, y_0). \tag{43}$$

Por lo tanto, la ecuación de la recta tangente es:

$$y = f(x_0, y_0) \cdot (x - x_0) + y_0. \quad (44)$$

Ahora bien, supongamos que x_1 es un punto cercano a x_0, y por lo tanto estará dado como $x_1 = x_0 + h$. Sustituyendo x_1 en la ecuación (44), se obtiene:

$$y(x_1) = y(x_0 + h) \approx f(x_0, y_0) \cdot (x_0 + h - x_0) + y_0, \quad (45)$$

de esta manera se obtiene la siguiente formula de aproximación:

$$y(x_0 + h) \approx y_0 + h \cdot f(x_0, y_0). \quad (46)$$

Esta aproximación puede ser suficientemente buena, si el valor de h es realmente pequeño, de una décima ó menos. Pero si el valor de h es más grande, entonces se puede cometer un error mayor al aplicar dicha fórmula. Una forma de reducir el error y obtener así un método iterativo, es dividir la distancia $h = |x_1 - x_0|$ en n partes iguales y obtener entonces la aproximación en n pasos, por lo que h (paso) quedaría de la siguiente manera: $h = \frac{|x_1 - x_0|}{n}$.

Ahora bien, si se sabe que:

$$y_1 = y_0 + h \cdot f(x_0, y_0), \quad (47)$$

para obtener y_2 únicamente hay que pensar que ahora el papel de (x_0, y_0) lo toma el punto (x_1, y_1), por lo tanto si se sustituyen los datos adecuadamente, se obtiene que:

$$y_2 = y_1 + h \cdot f(x_1, y_1), \quad (48)$$

de la ecuación anterior ya se puede observar que la forma recursiva de este procedimiento está dada por la fórmula:

$$y_{n+1} = y_n + h \cdot f(x_n, y_n). \quad (49)$$

La expresión anterior resulta ser la conocida formula de Euler que se usa para aproximar el valor de $y(x_1)$ aplicándola sucesivamente desde x_0 hasta x_1 en pasos de longitud h. Expresando la ecuación (49) de forma general:

Nuevo valor = valor anterior + pendiente * tamaño de paso.

2.6.3 Método de Runge-Kutta

Los métodos de Runge-Kutta (RK) logran mayor exactitud que el procedimiento de Euler. Existen muchas variantes, pero todas tienen la forma generalizada de la ecuación:

$$y_{n+1} = y_n + h \cdot K_i(x_n, y_n; h), \quad (50)$$

donde $K_i(x_n, y_n; h)$.es la función incremento, la cual puede interpretarse como una pendiente representativa en el intervalo. La función incremento se escribe en forma general como:

$$K_i(x_n, y_n; h) = a_1 k_1 + a_2 k_2 + \cdots + a_i k_i, \quad (51)$$

donde las *a* son constantes y las *k* son:

$$k_1 = f(x_n, y_n) \quad (51.a)$$
$$k_2 = f(x_n + p_1 h, y_n + q_{11} k_1 h) \quad (51.b)$$
$$k_3 = f(x_n + p_2 h, y_n + q_{21} k_1 h + q_{22} k_2 h) \quad (51.c)$$
$$\cdot$$
$$\cdot$$
$$\cdot$$
$$k_i = f\left(x_n + p_{i-1} h, y_n + q_{i-1,1} k_1 h + q_{i-1,2} k_2 h + \cdots + q_{i-1,i-1} k_{i-1} h\right), \quad (51.d)$$

donde las *p* y las *q* son constantes. Se observa que las *k* son relaciones de recurrencia. Es decir, k_1 aparece en la ecuación k_2, la cual aparece en la ecuación k_3, etcétera. Como cada *k* es una evaluación funcional, esta recurrencia vuelve eficientes a los métodos RK para cálculos en computadora [23].
Es posible tener varios tipos de métodos de Runge-Kutta empleando diferentes números de términos en la función incremento especificada por *i*. Observe que el método de Runge-Kutta (RK) de primer orden con *n = 1* es, de hecho, el método de Euler.

La versión de cuarto orden de la ecuación (50) es:

$$y_{n+1} = y_n + h \cdot K_4(x_n, y_n; h), \quad (52)$$

donde $K_4(x_n, y_n; h)$ es la función de incremento truncada en orden 4, y se escribe como:

$$K_4(x_n, y_n; h) = \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4], \quad (53)$$

donde las *k* son:

$$k_1 \triangleq f(x_n, y_n) \quad (53a)$$
$$k_2 \triangleq f\left[y_n + \frac{h}{2} k_1, x_n + \frac{h}{2}\right] \quad (53b)$$
$$k_3 \triangleq f\left[y_n + \frac{h}{2} k_2, x_n + \frac{h}{2}\right] \quad (53c)$$
$$k_4 \triangleq f[y_n + h k_3, x_n + h]. \quad (53d)$$

2.7 EXPONENTE DE LYAPUNOV

El exponente de Lyapunov proporcionan información sobre qué tan divergente o convergente son las trayectorias de espacio en un sistema dinámico El número de exponentes de Lyapunov

equivalen al número de variables de estado en un sistema dinámico, y si al menos una es positiva, esta es una indicación de que el sistema es caótico [27]. El exponente de Lyapunov se define como:

$$\lambda(x_0; r) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} ln \left| \frac{\partial f(x; r)}{\partial x} \right|, \tag{54}$$

donde *f(x;r)* es una función en tiempo discreto que depende de los parámetros *x* y *r*.

Si se tiene que $\lambda(x_0; r) > 0$, entonces la trayectoria es caótica; si $\lambda(x_0; r) < 0$, la trayectoria tiene comportamiento regular. Los valores de r para los cuales $\lambda(x_0; r) = 0$ corresponen a puntos de bifurcación donde el comportamiento cambia de regular a caótico y viceversa.

Comportamientos:

- El comportamiento regular se presenta cuando un pequeño cambio de la condición inicial genera un cambio pequeño en el futuro distante; la trayectoria converge a una obrita periódica estable y el exponente de Lyapunov es menor o igual a cero, $\lambda \leq 0$.

- El comportamiento caótico se presenta cuando el sistema presenta sensibilidad a pequeños cambios a las condiciones iniciales, de tal manera que en el futuro distante la trayectoria tiene un comportamiento aperiódico; el exponente de Lyapunov es positivo ($\lambda \geq 0$), indicando asi una divergencia exponencial de trayectorias inicialmente vecinas,

El exponente de Lyapunov se anula ($\lambda(r_k) = 0$) para ciertos valores de r_k del parámetro *r*. En estos puntos se presenta una bifurcación, ya que el sistema cambia de comportamiento; por ejemplo de comportamiento regular a caótico o viceversa [28].

2.8 SINCRONIZACIÓN DE SISTEMAS CAÓTICOS

La sincronización es una de las dinámicas unidas más admirables que un sistema puede presentar. Un sistema está completamente sincronizado cuando todos sus elementos evolucionan idénticamente en el tiempo. Los procesos de sincronización son frecuentes en la naturaleza y están presentes en diferentes contextos.

El estudio de la sincronización entre sistemas acoplados es conocido desde el trabajo de Huygens en 1673 que trataba sobre péndulos interconectados. Posteriormente estos conceptos fueron aplicados a diversos sistemas, y fue a principios del siglo XX cuando se descubrió que también pueden observarse en sistemas eléctricos y electromecánicos [29-32]. Recientemente, a partir de los estudios de oscilaciones caóticas, se ha verificado la posibilidad de sincronización de sistemas que exhiben tal comportamiento. Han sido Pecora y Carroll los que han mostrado experimentalmente que comportamientos caóticos aparentemente aleatorios e imprevisibles pueden fundirse en una única trayectoria; con estos resultados surgieron nuevas expectativas en torno a la teoría del caos [33-35].

La idea que subyace bajo el fenómeno de sincronización es que dos sistemas caóticos, que inicialmente evolucionan sobre atractores diferentes, al acoplarse de algún modo, finalmente siguen una trayectoria común.
La sincronización entre dos sistemas se consigue cuando uno de los dos sistemas cambia su trayectoria a la seguida por el otro sistema o bien a una nueva trayectoria común a ambos sistemas, esto significa que, la trayectoria de estados de la señal caótica del receptor usada para desenmascarar el mensaje original, rastree asintóticamente la señal del transmisor. Para que esto suceda, el receptor debe de tener una señal de entrada externa o bien una variable de estado en particular, en general, una función de salida del transmisor al que se requiere sincronizar. El problema de sincronización es de una naturaleza bastante similar a la de diseñar un observador no lineal para un sistema de transmisor, como ya se ha observado en el trabajo [36], en el trabajo [37] también se presenta la sincronización con un enfoque de observadores de los sistemas caóticos Lorenz, Rössler y Sport, utilizando específicamente el observador de Lumberger.

Un sistema caótico en específico se puede sincronizar con otro totalmente diferente, tal y como se muestra en el trabajo [38] donde se establecen condiciones para las cuales el sistema caótico de Rössler de 4-dimensiones se sincroniza con diferentes sistemas (señales) por medio de un control aditivo.

Existe una gran diferencia en los procesos a la consecución de estados sincronizados, dependiendo del tipo de acoplamiento [39].

- Acoplamiento unidireccional: Un sistema se subdivide en dos subsistemas, uno de ellos envuelve y conduce al otro, siendo la respuesta del sistema esclavizado seguir la dinámica del sistema conductor. Dicho de otro modo, cuando la evolución de uno de los dos sistemas no es alterada por el acoplamiento la configuración resultante es un acoplamiento unidireccional. Este tipo de configuración es conocida como maestro-esclavo. Una aplicación típica es en la seguridad de las comunicaciones.

- Acoplamiento bidireccional: Aquí ambos subsistemas son acoplados con otro, ó cuando los dos subsistemas son conectados de tal forma que sus trayectorias están mutuamente influenciadas por el comportamiento del otro. Esta situación se da en láseres con retroalimentación.

2.8.1 Tipos de sincronización.

Existen varios tipos de sincronización de sistemas, entre las más conocidas se encuentras:

- Sincronización Completa (CS): La sincronización completa fue el primer tipo descubierto. Consiste en una perfecta unión de las trayectorias caóticas de dos sistemas conseguidos por medio de una señal de acoplamiento. Con este mecanismo se demuestra que dos sistemas caóticos son acoplados unidireccionalmente sólo si todos los exponentes de Lyapunov del subsistema a sincronizar son negativos.

- Sincronización Generalizada (GS): En la sincronización generalizada se usan sistemas completamente diferentes y se asocia la salida de uno de ellos como una función dada de la entrada del otro.

- Sincronización de fase (PS): La sincronización de fase se utiliza para sistemas de osciladores no idénticos o sistemas rotatorios, que pueden alcanzar un régimen intermedio donde se produce una unión de las fases, mientras la correlación entre las amplitudes permanece débil (amplitudes descorrelacionadas, caóticas).

- Sincronización de retardo (LS): La sincronización de retardo es un paso entre la sincronización de fase y la completa. La cuestión radica en que existe un límite asintótico entre el tiempo *t*, de la salida de un sistema, y la salida del otro τ_{lag}, esto hace que las fases y las amplitudes vayan unidas, pero con la presencia de un tiempo de retardo.

- Sincronización de retardo intermitente (ILS): En la sincronización de retardo intermitente, los sistemas la mayor parte del tiempo verifican la sincronización de fase, pero existen estallidos intermitentes de comportamientos no sincronizados, debido a que la trayectoria pasa por una región del atractor donde el exponente local de Lyapunov se acorta en alguna dirección y se vuelve positivo.

- Sincronización de fase imperfecta (IPS): La sincronización de fase imperfecta ocurre cuando estando fuera del régimen de sincronización de fase (PS), las fases se ajustan.

- Casi sincronización (AS): La "casi" sincronización es debida a la existencia del límite asintótico entre un subconjunto de variables de un sistema y el correspondiente subconjunto de variables del otro sistema [37].

2.8.2 Importancia de la sincronización.

La sincronización entre dos de sistemas caóticos ha sido intensivamente estudiada en años recientes [40] en campos como óptica [41] con potenciales aplicaciones tecnológicas, por ejemplo, comunicaciones seguras [42], [43]. Los circuitos electrónicos no lineales, en particular, han impulsado el estudio y entendimiento de la sincronización del caos. Debido a su simplicidad y al hecho de que todas las variables del circuito son accesibles y medibles, los circuitos fueron usados en estudios pioneros de comunicaciones caóticas [44].

La sincronía dentro de un sistema de comunicaciones se vuelve crítica conforme la información a transmitir aumenta, de ahí viene el interés de emplear los recursos que nos brinda la teoría del caos para poder manipularla y obtener un dispositivo confiable para ser usado dentro de las comunicaciones seguras.

Al existir sincronización se obtiene precisión de la información y la incertidumbre de tener datos de calidad se reduce, ya que se asegura que los datos intercambiados entre el transmisor y el receptor sean exactos. Se genera la capacidad de detección de conflictos, por ejemplo, cuando hay algún archivo que no está sincronizado correctamente debido a diferentes versiones de ambos lados.

El proceso de sincronización puede hacerse manualmente o automáticamente utilizando alguna herramienta de software, lo que permite mayor confiabilidad. Y aunque en este trabajo de tesis la sincronización solo es de transmisor a receptor, existe posibilidad de compresión de datos, si es que la sincronización se hace a través de una red.

2.9 SISTEMAS CAÓTICOS APLICADOS EN CRIPTOGRAFÍA

En el campo de la ingeniería eléctrica y electrónica, al igual que en otras muchas disciplinas, existen múltiples áreas en las que el caos presenta numerosas aplicaciones potenciales: el diseño de nuevos dispositivos electrónicos con mejores características (menor consumo, mayor rapidez de funcionamiento y/o ancho de banda, compatibilidad electromagnética mejorada, etc.), el análisis y modelado de comportamientos anómalos y/o transitorios de dispositivos electrónicos convencionales, el análisis de señales biomédicas, la generación de números aleatorios, la criptografía, etc. [44].

Todo sistema caótico es muy sensible a las condiciones iníciales y genera un comportamiento aparentemente aleatorio pero a la vez completamente determinista. Estas propiedades del caos proporcionan un potencial para aplicaciones en criptografía ya que las predicciones a largo plazo de los sistemas caóticos son muy difíciles. Los sistemas caóticos tienen la particularidad de que pueden ser sincronizados. Este hecho puede ser utilizado para enmascarar mensajes sobre una portadora caótica. Un receptor autorizado puede reproducir la portadora caótica que enmascara el mensaje, utilizando un sistema caótico sincronizado con el emisor, y desencriptarlo.

Para incrementar la complejidad y seguridad de un sistema caótico se recomienda usar atractrores caóticos de multi-enrollamientos en lugar de optar por los sistemas caóticos diseñados a partir de la topología general simple que no permiten más de dos enrollamientos [45].

El caos en relación con la generación de números aleatorios permite repetir la misma cadena de números siempre que se utilice la misma función de correspondencia caótica (o atractor) y valor inicial. La apariencia aleatoria del sistema hace prácticamente imposible los ataques del tipo codebook (un codebook contiene todas las posibles transformaciones entre texto llano y texto cifrado bajo cada clave) [46]. Puesto que las funciones caóticas son muy sensibles a las condiciones iníciales, cualquier ligera diferencia en el valor inicial utilizado, significará que el texto cifrado producido utilizando caos será muy diferente. Esto significa que el sistema será robusto contra ataques por fuerza bruta, ya que el número de posibles claves es impresionantemente grande, dependiendo de la precisión de los valores iníciales que estarán en función del hardware utilizado, y que puede ser más o menos elevado según el dominio sea analógico o digital.

También existen dos clases de cifradores caóticos:

1. Criptosistemas basados en técnicas de sincronización de caos de circuitos analógicos.
2. Cifradores basados en caos realizados sobre circuitos digitales y computadoras.

2.10 SINCRONIZACIÓN MEDIANTE FORMAS HAMILTONIANAS GENERALIZADAS

Los atractores con multi-enrollamientos pueden ser sincronizados al aplicar aproximación Hamiltoniana. Esta técnica es bien descrita en [47]. Se optó por esta técnica debido a su condición de ser autómata [48], [49].

Considerando el sistema dinámico:

$$\dot{x} = f(x), \tag{55}$$

donde $\dot{x} \in \mathbb{R}^n$ es el vector de estado y $f: \mathbb{R}^n \to \mathbb{R}^n$ es una función no lineal. En [47] se reporta como el sistema (5) puede ser escrito en la *forma canónica Hamiltoniana Generalizada*:

$$\dot{x} = \mathcal{J}(x)\frac{\partial H}{\partial x} + \mathcal{S}(x)\frac{\partial H}{\partial x} + \mathcal{F}(x), \quad \dot{x} \in \mathbb{R}^n, \tag{56}$$

donde *H(x)* denota una función de energía suave la cual es globalmente positiva definida en $\mathbb{R}^n$. El vector gradiente de H, denotado por $\frac{\partial H}{\partial x}$, se asume que existe en todas partes. Se usa una función cuadrática de energía $H(x) = \frac{1}{2}x^T\mathcal{M}x$ siendo M una matriz constante, simétrica definida positiva. En este caso, $\frac{\partial H}{\partial x} = \mathcal{M}x$. Las matrices $\mathcal{J}(x)$ y $\mathcal{S}(x)$ satisfacen, para toda $\dot{x} \in \mathbb{R}^n$, las siguientes propiedades: $\mathcal{J}(x) + \mathcal{J}^T(x) = 0$ y $\mathcal{S}(x) = \mathcal{S}^T(x)$. El vector campo $\mathcal{J}(x)\frac{\partial H}{\partial x}$ exhibe la parte conservativa del sistema y se refiere también a la parte sin trabajo, y $\mathcal{J}(x)$ denota la parte trabajadora o no-conservativa del sistema.
Para ciertos sistemas $\mathcal{S}(x)$ es definida negativa o semidefinida negativa. Por lo tanto, el vector campo se refiere a la parte disipativa del sistema. Si por otra parte, $\mathcal{S}(x)$ es definida positiva, semidefinida positiva, o indefinida, esta representa claramente la parte global, semi-global, o desestabilizadora local del sistema, respectivamente. En el último caso, se puede (aunque no únicamente) descomponer una matriz simétrica indefinida en la suma de una matriz semidefinida negativa $\mathcal{R}(x)$ y una matriz simétrica semidefinida positiva $\mathcal{N}(x)$. Finalmente, $\mathcal{F}(x)$ representa un vector campo localmente desestabilizante. En el contexto del diseño de observador, se considera una clase especial de la forma Generalizada Hamiltoniana con salida $y(t)$, dada por:

$$\dot{x} = \mathcal{J}(y)\frac{\partial H}{\partial x} + (\mathcal{I} + \mathcal{S})\frac{\partial H}{\partial x} + \mathcal{F}(y), \quad \dot{x} \in \mathbb{R}^n, \tag{57}$$

$$y = \mathcal{C}\frac{\partial H}{\partial x}, \quad y \in \mathbb{R}^m, \tag{58}$$

donde $\mathcal{S}$ es una matriz simétrica constante, no necesariamente de un signo definido. $\mathcal{I}$ es una matriz simétrica diagonal constante y $\mathcal{C}$ es una matriz constante.
Se denota la estimación del estado $x(t)$ por $\xi(t)$, y se considera la función de energía Hamiltoniana $H(\xi)$ como la particularización de H en términos de $\xi(t)$. Similarmente, la salida estimada se

denota por $\eta(t)$, computarizada en términos de $\xi(t)$. El vector gradiente $\frac{\partial H(\xi)}{\partial \xi}$ es, naturalmente, de la forma $\mathcal{M}\xi$ siendo $\mathcal{M}$ una matriz constante, simétrica definida positiva. Un observador de estado no lineal para (58) está dado por:

$$\xi = \mathcal{J}(y)\frac{\partial H}{\partial \xi} + (\mathcal{I} + \mathcal{S})\frac{\partial H}{\partial \xi} + \mathcal{F}(y) + \mathcal{K}(y - \eta), \quad \xi \in \mathbb{R}^n, \tag{59}$$

$$\eta = \mathcal{C}\frac{\partial H}{\partial \xi}, \quad \eta \in \mathbb{R}^m, \tag{60}$$

donde $\mathcal{K}$ es la ganancia del observador. El error de estimación, definido como $e(t) = x(t) - \xi(t)$, y el error de estimación de la salida, definido como $e_y(t) = y(t) - \eta(t)$, están gobernados por:

$$\dot{e} = \mathcal{J}(y)\frac{\partial H}{\partial e} + (\mathcal{I} + \mathcal{S} - \mathcal{K}\mathcal{C})\frac{\partial H}{\partial e}, \quad e \in \mathbb{R}^n, \tag{61}$$

$$e_y = \mathcal{C}\frac{\partial H}{\partial e}, \quad e_y \in \mathbb{R}^m, \tag{62}$$

donde $\frac{\partial H}{\partial e}$ realmente se destaca, con un cierto abuso de notación, para el vector gradiente de la función de energía modificada, $\frac{\partial H(e)}{\partial e} = \frac{\partial H}{\partial x} - \frac{\partial H}{\partial \xi} = \mathcal{M}(x - \xi) = \mathcal{M}e$. Se establece cuando sea necesario, $\mathcal{I} + \mathcal{S} = \mathcal{W}$.

2.10.1 Condición de sincronización.

El sistema esclavo (observador de estado no lineal) (59) se sincroniza con el sistema caótico maestro en forma Hamiltoniana Generalizada (58), siempre y cuando cumpla la siguiente condición:

$$\lim_{t\to\infty} \|x(t) - \xi(t)\| = 0. \tag{63}$$

No importa qué condiciones iniciales tengan $x(0)$ y $\xi(0)$, los sistemas tienden a sincronizarse. El error de estimación $e(t) = x(t) - \xi(t)$ corresponde al error de sincronización.

2.10.2 Sincronización de osciladores caóticos con multi-enrollamientos.

A partir de las ecuaciones (56)-(58) se puede definirá matemáticamente la sincronización de dos sistemas caóticos basados en SNLF mediante formas hamiltonianas generalizadas y observadores en un sistema maestro-esclavo, el modelo maestro del sistema de ecuaciones (22), está dado por [50]:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2b} & \frac{1}{2} \\ -\frac{1}{2b} & 0 & 1 \\ -\frac{1}{2} & -1 & 0 \end{bmatrix}\frac{\partial H}{\partial x} + \begin{bmatrix} 0 & \frac{1}{2b} & -\frac{1}{2} \\ \frac{1}{2b} & 0 & 0 \\ -\frac{1}{2} & 0 & -c \end{bmatrix}\frac{\partial H}{\partial x} + \begin{bmatrix} 0 \\ 0 \\ d_1 f(x) \end{bmatrix}, \tag{64}$$

se toma como función de energía Hamiltoniana:

$$H(x) = \frac{1}{2}[a{x_1}^2 + a{x_2}^2 + {x_3}^2], \tag{65}$$

y como vector gradiente:

$$\frac{\partial H}{\partial x} = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} ax_1 \\ ax_2 \\ x_3 \end{bmatrix}. \tag{66}$$

El vector campo llama a las señales x_1 y x_2 para ser usadas como salidas, del modelo maestro (64). En (64) se usa $y = x_1$. Las matrices $\mathcal{C}$, $\mathcal{S}$ e $\mathcal{I}$ están dadas por:

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ a & & \end{bmatrix},$$

$$\mathcal{S} = \begin{bmatrix} 0 & \frac{1}{2b} & -\frac{1}{2} \\ \frac{1}{2b} & 0 & 0 \\ -\frac{1}{2} & 0 & -c \end{bmatrix},$$

$$\mathcal{I} = \begin{bmatrix} 0 & \frac{1}{2b} & \frac{1}{2} \\ -\frac{1}{2b} & 0 & 1 \\ -\frac{1}{2} & -1 & 0 \end{bmatrix},$$

La pareja $(\mathcal{C}, \mathcal{S})$ es observable. Por lo tanto, el observador no lineal en (64), para ser utilizado como el modelo esclavo, está diseñado a partir de (59) como:

$$\begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2b} & \frac{1}{2} \\ -\frac{1}{2b} & 0 & 1 \\ -\frac{1}{2} & -1 & 0 \end{bmatrix} \frac{\partial H}{\partial \xi} + \begin{bmatrix} 0 & \frac{1}{2b} & -\frac{1}{2} \\ \frac{1}{2b} & 0 & 0 \\ -\frac{1}{2} & 0 & -c \end{bmatrix} \frac{\partial H}{\partial \xi} + \begin{bmatrix} 0 \\ 0 \\ d_1 f(\xi) \end{bmatrix} + \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} e_y. \tag{67}$$

Con el fin de garantizar que la estabilidad exponencial asintótica del error de reconstrucción de las trayectorias de estado (es decir, el error de sincronización $e(t)$) sea cero, se utilizan ganancias $k_i, i = 1, 2, 3$. De (64) y (65) se tiene que la dinámica del error de sincronización está gobernada por [48]:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2b} & \frac{1}{2} \\ -\frac{1}{2b} & 0 & 1 \\ -\frac{1}{2} & -1 & 0 \end{bmatrix} \frac{\partial H}{\partial e} + \begin{bmatrix} 0 & \frac{1}{2b} & -\frac{1}{2} \\ \frac{1}{2b} & 0 & 0 \\ -\frac{1}{2} & 0 & -c \end{bmatrix} \frac{\partial H}{\partial e} + \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} e_y. \tag{68}$$

Al realizar todas las operaciones en las ecuaciones (64) y (65), se obtiene un sistema reducido dado por (69) y (70) para el sistema maestro y el sistema esclavo, respectivamente.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -ax_1 - bx_2 - cx_3 + d_1 f(x_1; k, \alpha, p, q) \end{aligned} \tag{69}$$

$$\begin{aligned} \dot{y}_1 &= y_2 + k_1(x_1 - y_1) \\ \dot{y}_2 &= y_3 + k_2(x_1 - y_1) \\ \dot{y}_3 &= -ay_1 - by_2 - cy_3 + d_1 f(x_1; k, \alpha, p, q) + k_3(x_1 - y_1). \end{aligned} \tag{70}$$

2.11 CONCLUSIÓN

En este capítulo se abordó el estado del arte sobre osciladores caóticos que se usaran en este marco de tesis para su implementación y análisis, se presentaron de manera general una introducción a los métodos numéricos, y una breve explicación de los métodos de Euler y Runge-Kutta que fueron los utilizados para resolver los sistemas caóticos en esta tesis, también se mostraron algunas aplicaciones de los métodos numéricos en general, enfocadas a la solución de ecuaciones diferenciales. Se hizo referencia a trabajos importantes donde se utiliza la sincronización de sistemas caóticos para aplicaciones específicas, toda esta teoría fue la base para el desarrollo de esta tesis, y sus principales aportaciones.

Capítulo 3

Simulación de osciladores caóticos

3.1 INTRODUCCIÓN

En este capítulo se presentan las simulaciones correspondientes a los dos tipos de osciladores caóticos utilizados, específicamente el de SNLF y el del circuito de Chua, las simulaciones se realizan describiendo ambos sistemas mediante código en el software matemático MATLAB, obteniendo las soluciones de los sistemas mediante los métodos numéricos de Euler y Runge-Kutta orden 4 (RK4). Además se muestran sus extensiones a multi-enrollamientos para ambos tipos de osciladores. Además se muestra una manera alternativa de simular el oscilador caótico basado en el circuito de Chua utilizando la herramienta Simulink.

3.2 SIMULACIÓN DE OSCILADOR CAÓTICO BASADO EN SNLF

En esta sección se presenta la simulación de un oscilador caótico basado en SNLF, con ayuda del software MATLAB, se utilizaron los métodos de Euler hacia delante y el método de Runge-Kutta de orden 4 (RK4) para la integración numérica del sistema de ecuaciones (22), se implementa cada uno de los casos de las SNLF para la generación de multi-enrollamientos mediante código en MATLAB.

Se consideraron los valores de los parámetros usados en [27]: $a = b = c = d_1 = 0.7$, la pendiente $s = 0.0165$, los niveles de saturación $k = 1$, se tomaron como condiciones iniciales $x_0 = y_0 = z_0 = 0.01$ y se utiliza un tamaño de paso de integración de *step* = 0.01.

3.2.1 Solución mediante el método de Euler

Los casos de SNLF para multi-enrollamientos se resuelven primeramente mediante el método de Euler hacia delante, cada caso se simula mediante MATLAB a través de archivos .m, en los cuales se describe la solución de los sistemas de ecuaciones (22), los cuales generan 2, 3, 4, 5 y 6 enrollamientos respectivamente. Los resultados de la simulación de cada caso se muestran en las figuras (3.1-3.5), en cada caso se muestran las señales *x*, *y*, *z* y el atractor *x*-*y*.

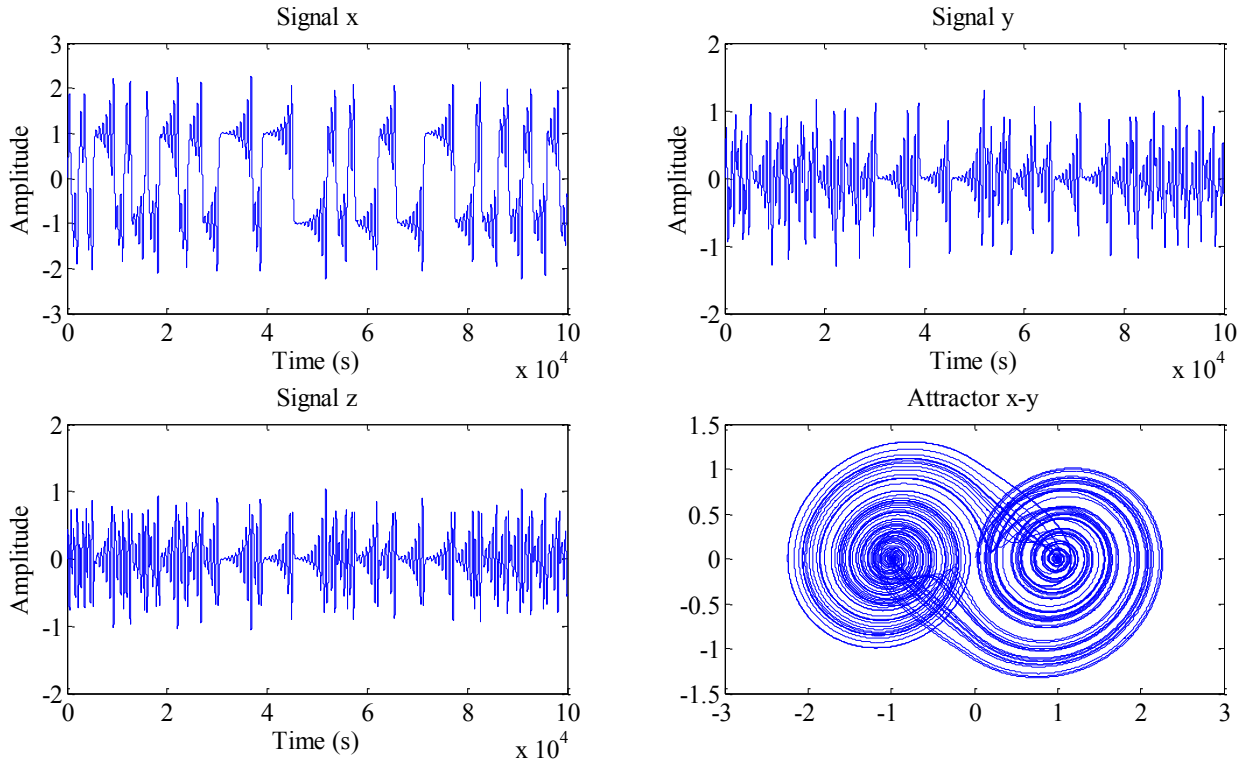

Figura 3.1. Señales SNLF con dos enrollamientos resuelto por método de Euler.

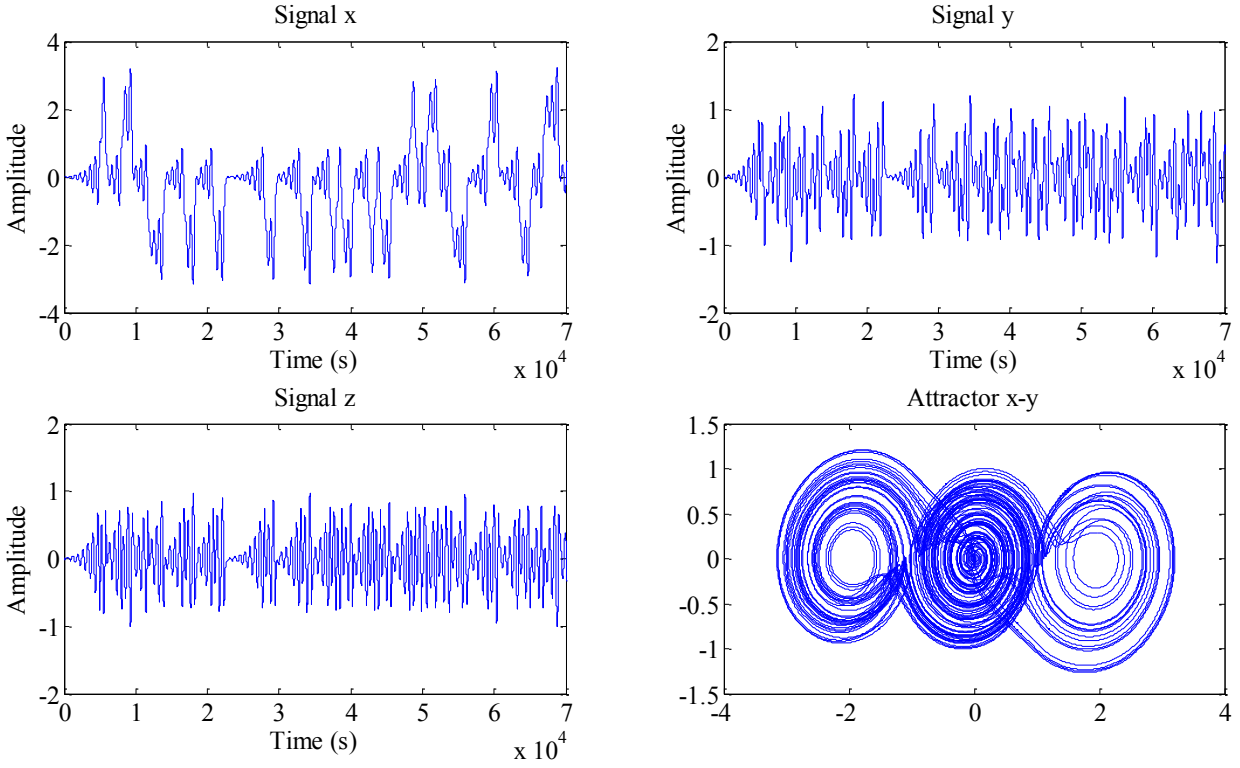

Figura 3.2. Señales SNLF con tres enrollamientos resuelto por método de Euler.

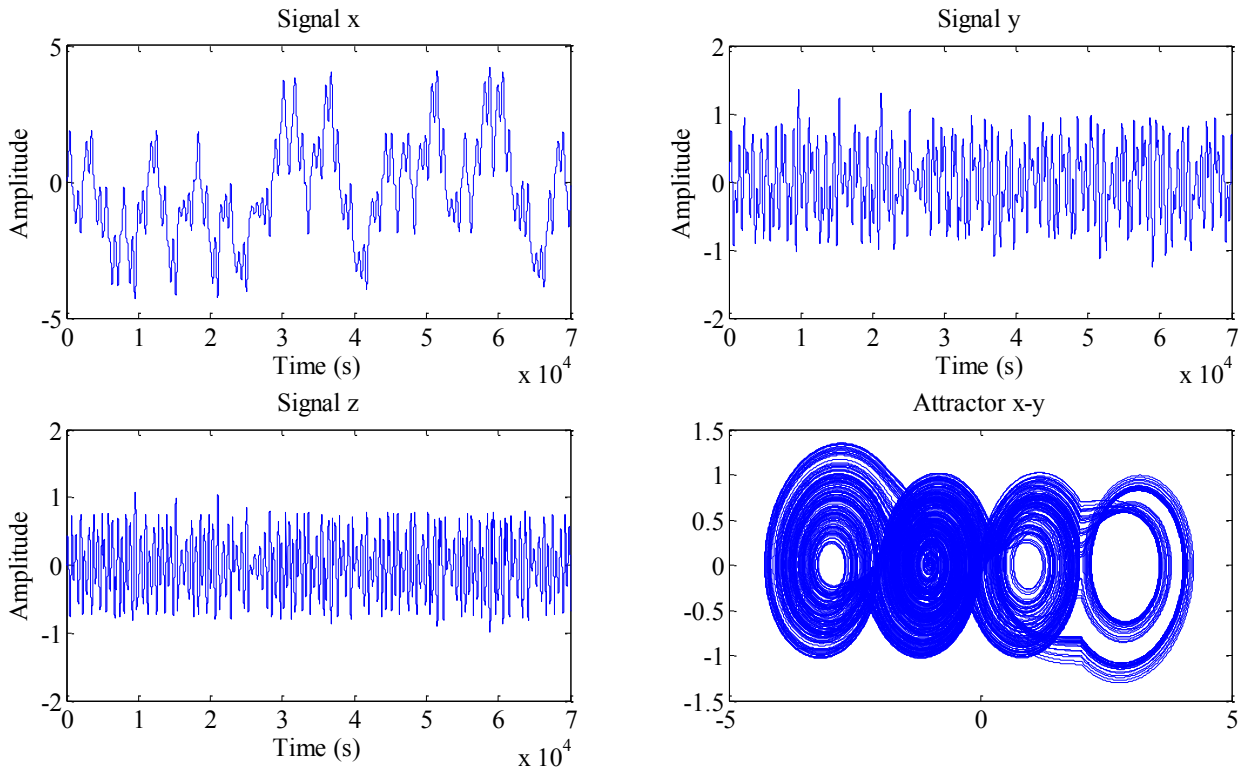

Figura 3.3. Señales SNLF con cuatro enrollamientos resuelto por método de Euler.

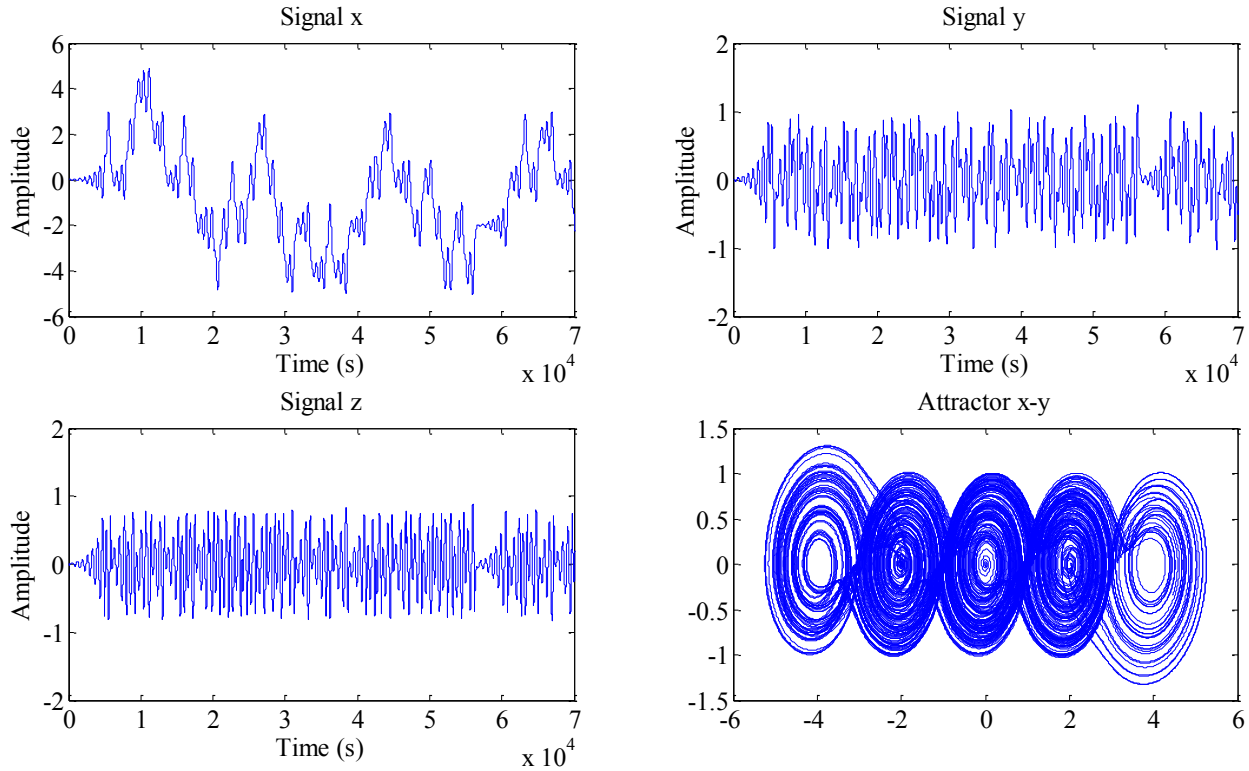

Figura 3.4. Señales SNLF con cinco enrollamientos resuelto por método de Euler.

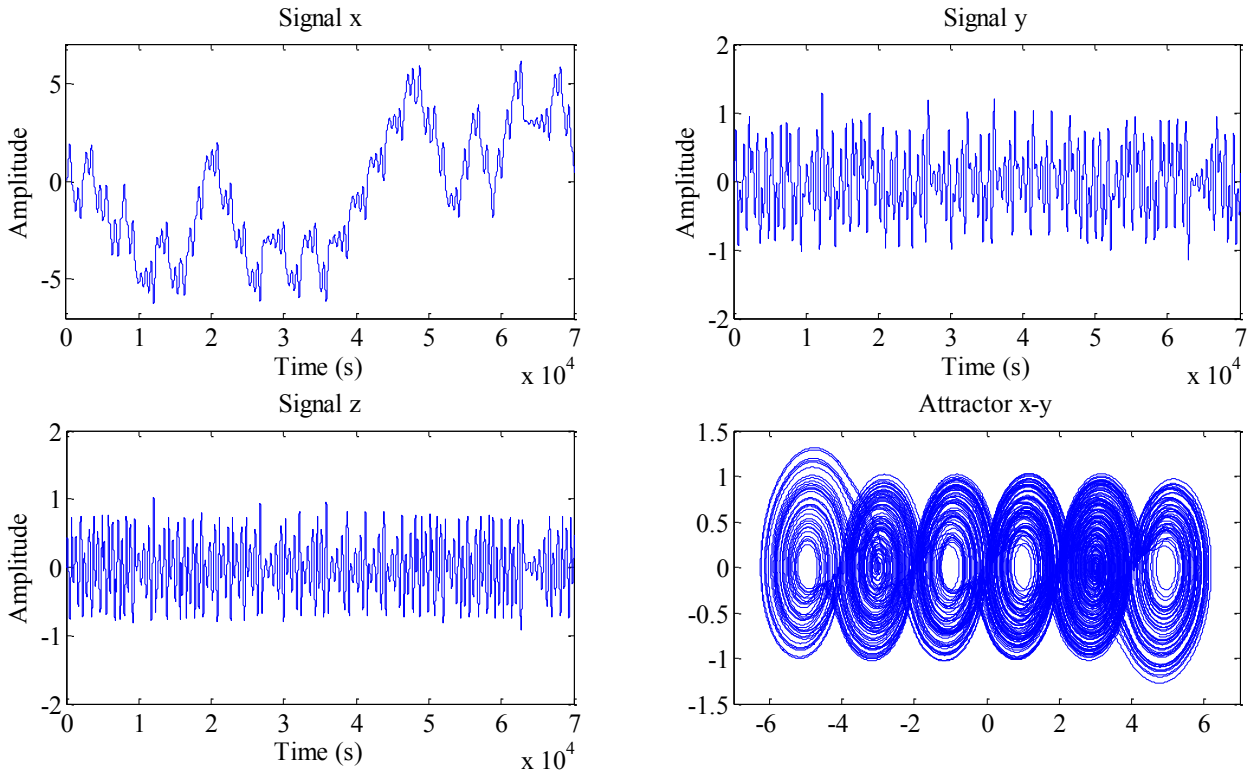

Figura 3.5. Señales SNLF con seis enrollamientos resuelto por método de Euler.

3.2.2 Solución mediante el método de Runge-Kutta orden 4

En esta subsección se presenta los resultados de las simulaciones en MATLAB de los casos de multi-enrollamientos del oscilador caótico basado en SNLF resueltos mediante el método de Runge-Kutta de orden 4 (RK4), se utilizan los mismos valores de las constantes de [27], los resultados son obtenidos mediante código en MATLAB los cuales generan 2, 3, 4, 5 y 6 enrollamientos respectivamente. Los resultados de la simulación de cada caso se muestran en las figuras (3.6-3.10), en cada caso se muestran las señales *x*, *y*, *z* y el atractor *x*-*y*.

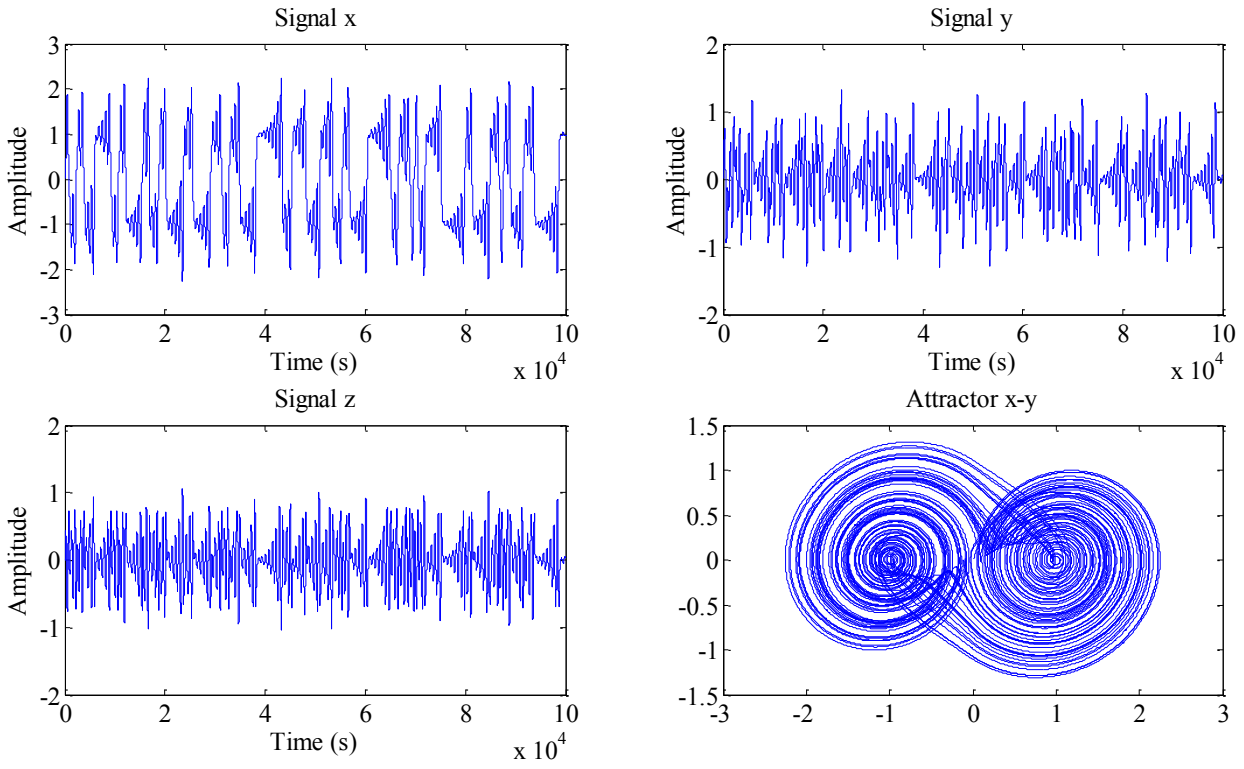

Figura 3.6. Señales SNLF con dos enrollamientos resuelto por método de RK4.

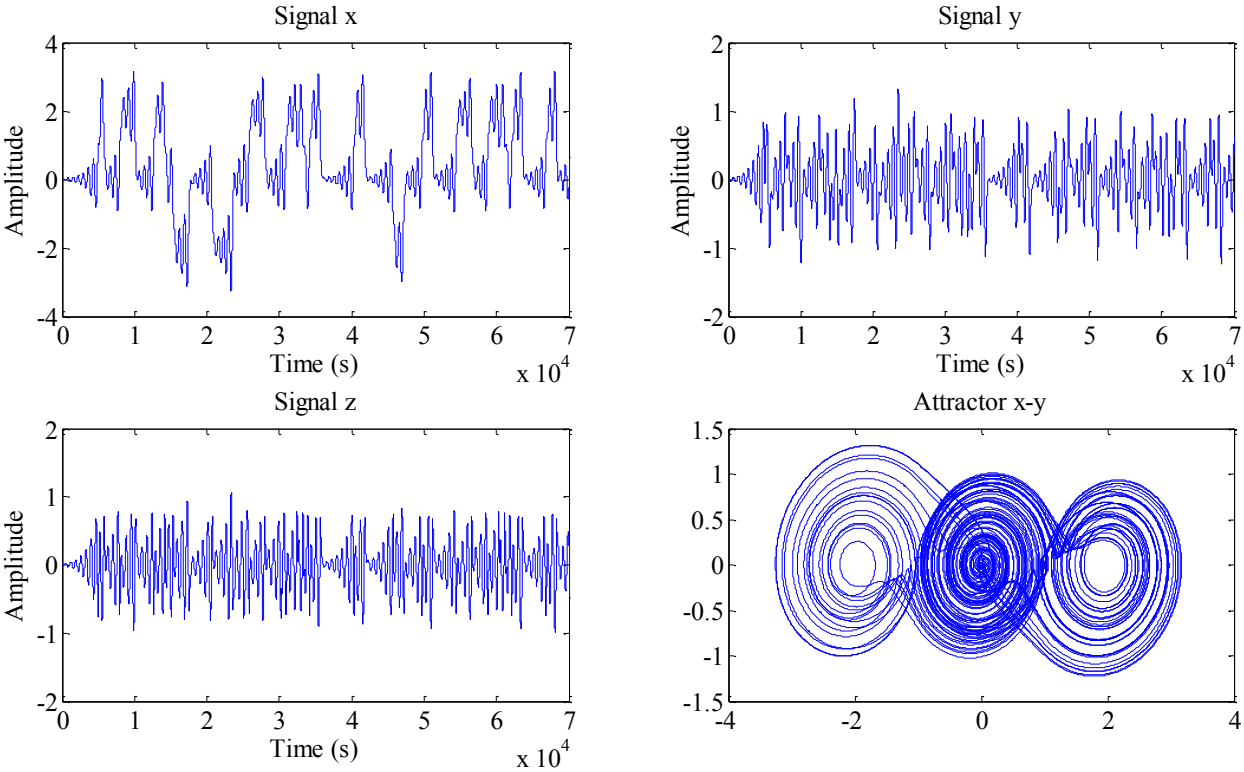

Figura 3.7. Señales SNLF con tres enrollamientos resuelto por método de RK4.

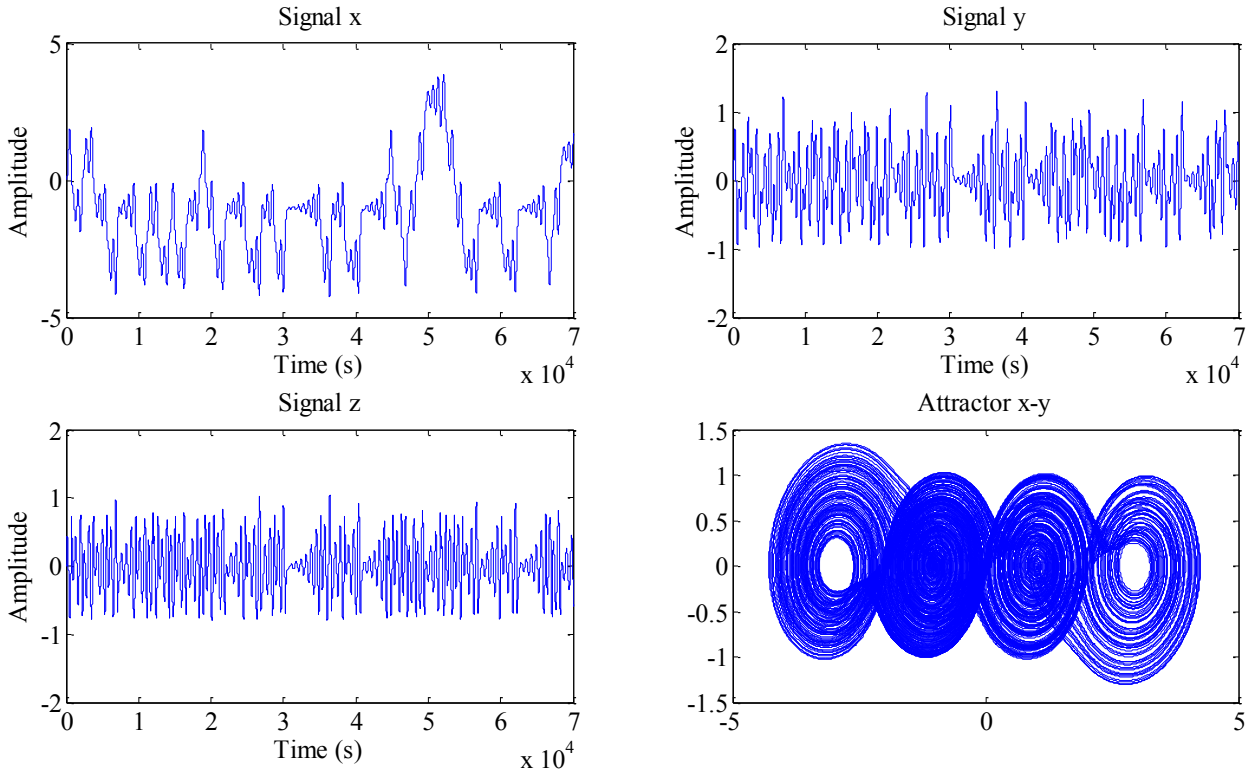

Figura 3.8. Señales SNLF con cuatro enrollamientos resuelto por método de RK4.

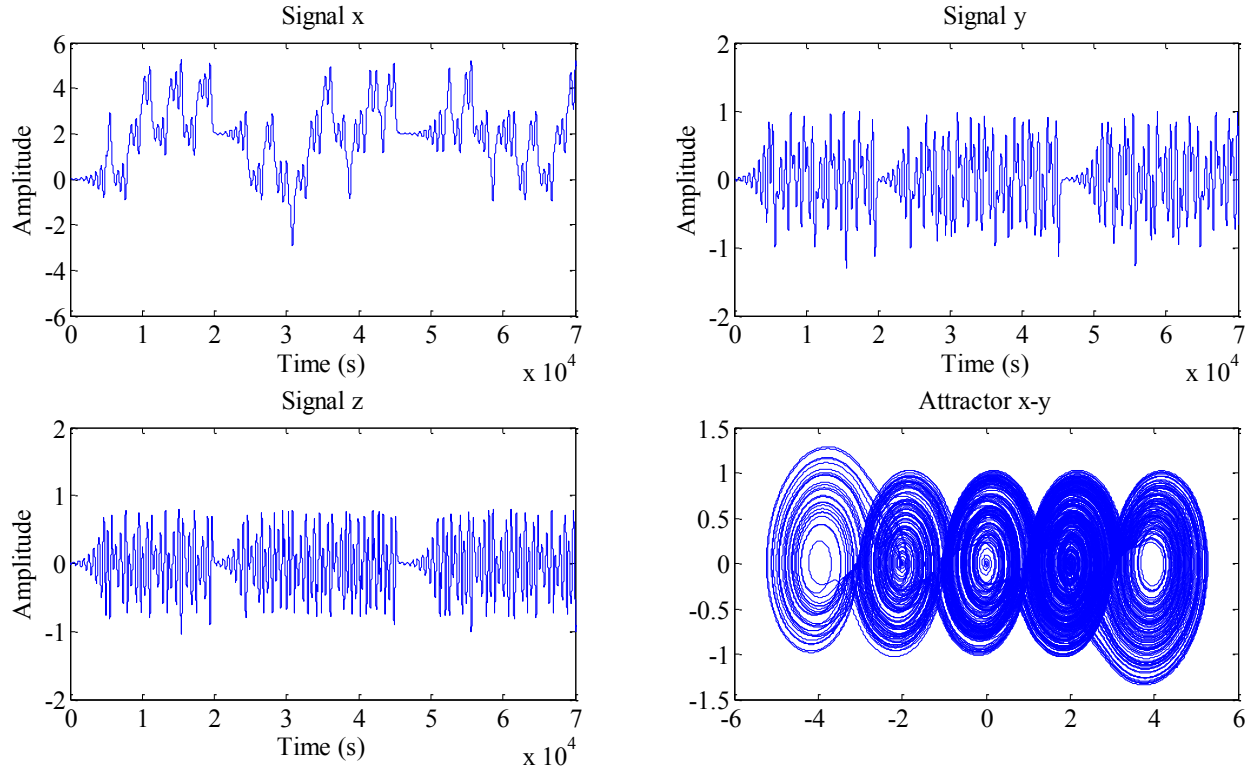

Figura 3.9. Señales SNLF con cinco enrollamientos resuelto por método de RK4.

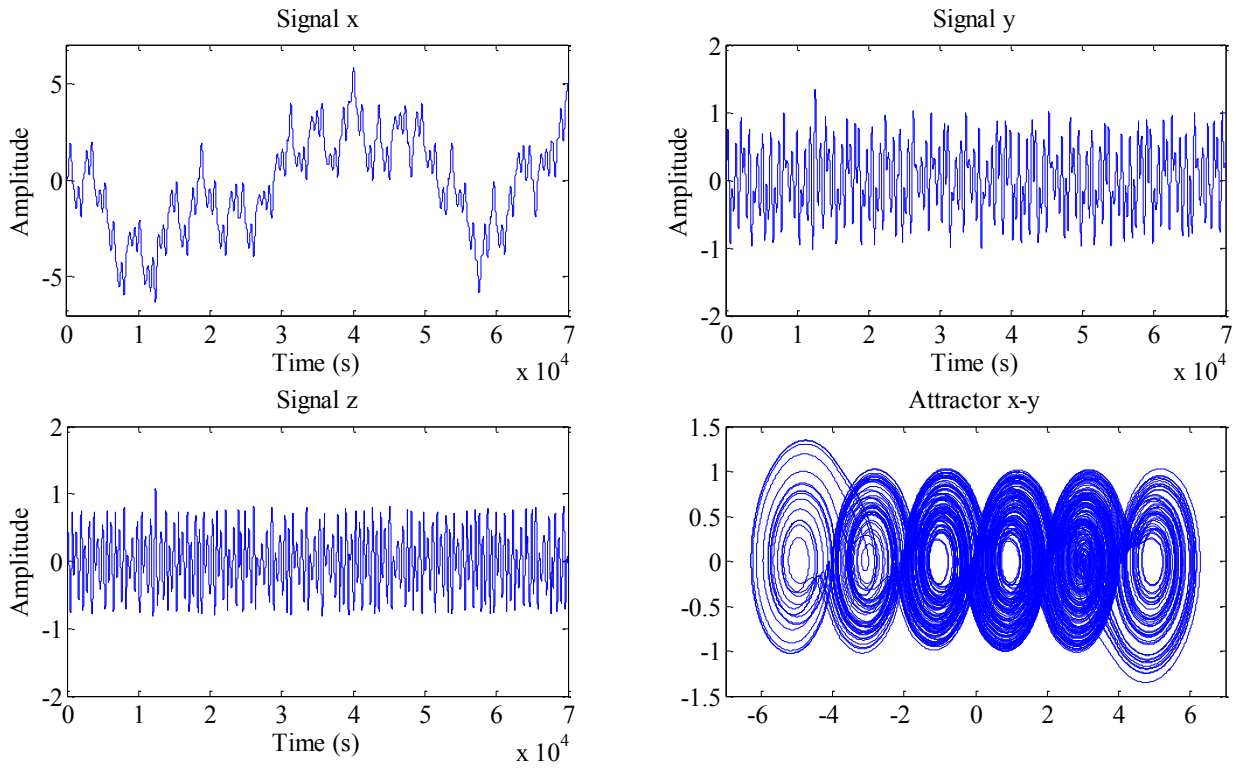

Figura 3.10. Señales SNLF con seis enrollamientos resuelto por método de RK4.

3.3 SIMULACIÓN DE OSCILADOR CAÓTICO BASADO EN EL CIRCUITO DE CHUA

En esta sección se presenta la simulación de un oscilador caótico basado en el circuito de Chua, con ayuda del software Matlab, se utilizan los métodos de Euler hacia delante y el método de RK4 para la integración numérica de las ecuaciones (33)-(36), se implementa cada uno de los casos de la curva característica del diodo de Chua para la generación de multi-enrollamientos mediante código. A diferencia del oscilador caótico basado en SNLF, donde para todos los casos de enrollamientos los valores de las constantes eran los mismos, en este oscilador se tienen que tomar valores específicamente para el numero de enrollamientos que se desean realizar, se consideran los valores de los parámetros de la tabla 3.1, estos parámetros son escogidos con la finalidad de optimizar el exponente de Lyapunov para cada uno de los casos de la curva característica del diodo de Chua y son tomados de [28], como paso de integración se utiliza un valor de *step* = 0.001 y condiciones iniciales de *x(0)=0.1*, y*(0)=0* y *z(0)=0*.

Tabla 3.1. Valores optimizados para el oscilador caótico basado en el circuito de Chua.

Enrollamientos	**Valores de las constantes**		
	Pendientes m_0, m_1	Puntos de quiebre P	Coeficientes (α, β)
2	-1.131, -0.395	2.618	11.991, 14.997
3	-0.385, -0.234	0.234, 2.394	10.297, 12.565
4	-3.981, -0.397	0.101, 1.206, 1.467	11.489, 14.674
5	-2.813, -0.356	0.893, 1.559, 3.432, 4.114	10.227, 11.786
6	-1.464, -0.381	0.19, 0.70, 1.58, 2.21, 3.31	10.556, 11.415

3.3.1 Solución mediante el método de Euler

En esta subsección se presenta los resultados de las simulaciones del oscilador caótico de Chua para cada uno de los casos de enrollamientos utilizando el método de Euler, se utilizaron los valores de la tabla 3.1, las simulaciones son obtenidas a través de archivos *.m* que se realizaron para la solución del sistemas de ecuaciones (33)-(36), estos generan 2, 3, 4, 5 y 6 enrollamientos respectivamente. Los resultados de la simulación de cada caso se muestran en las figuras (3.11-3.15), en cada caso se muestran las señales *x*, *y*, *z* y el atractor *x*-*y*.

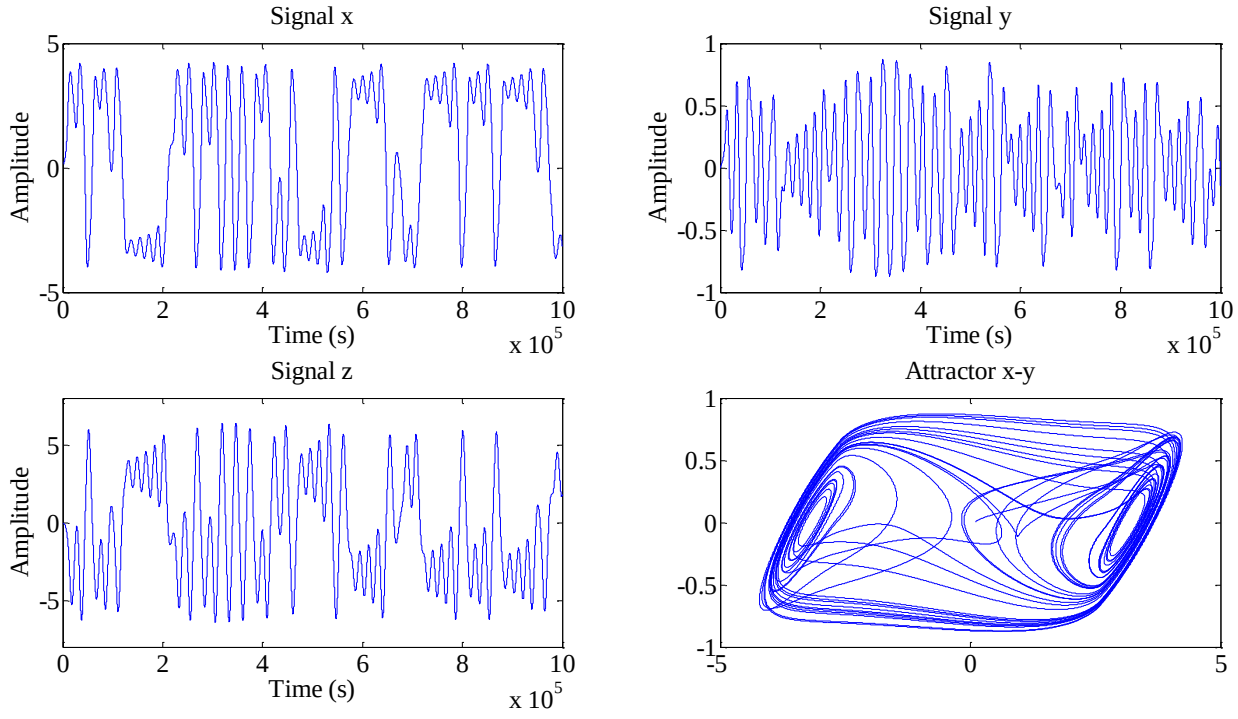

Figura 3.11. Señales Chua con dos enrollamientos resuelto por método de Euler.

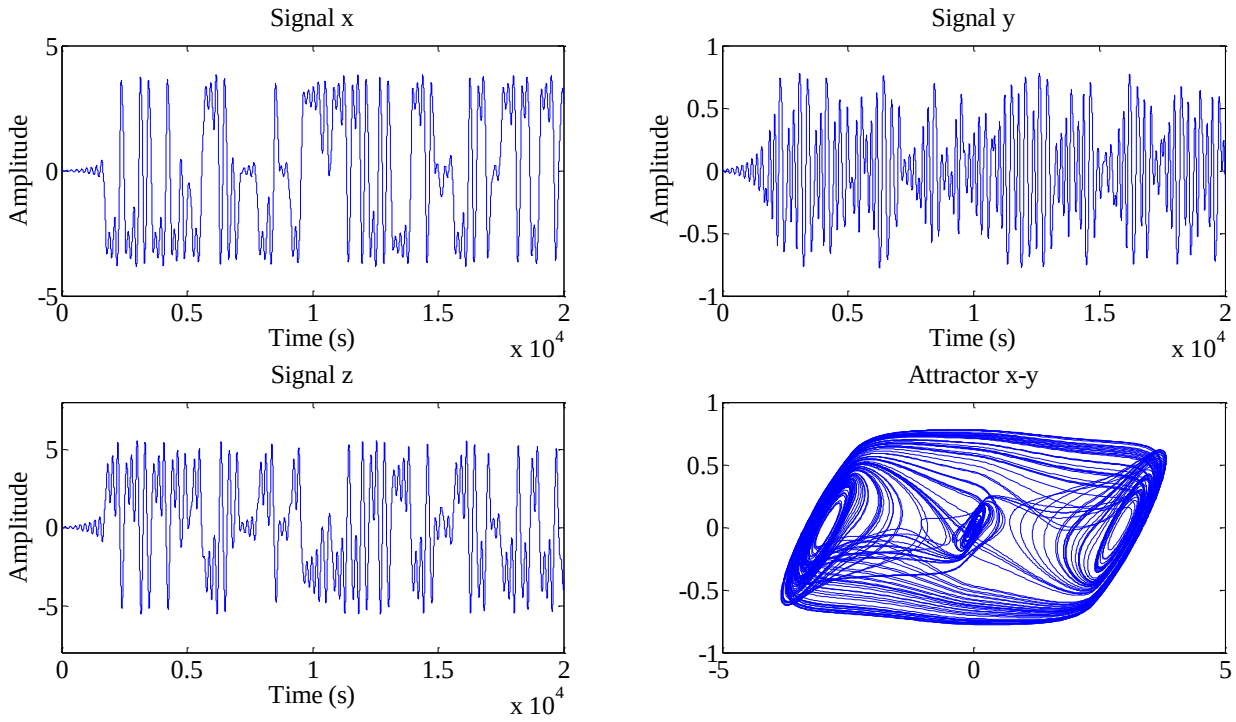

Figura 3.12. Señales Chua con tres enrollamientos resuelto por método de Euler.

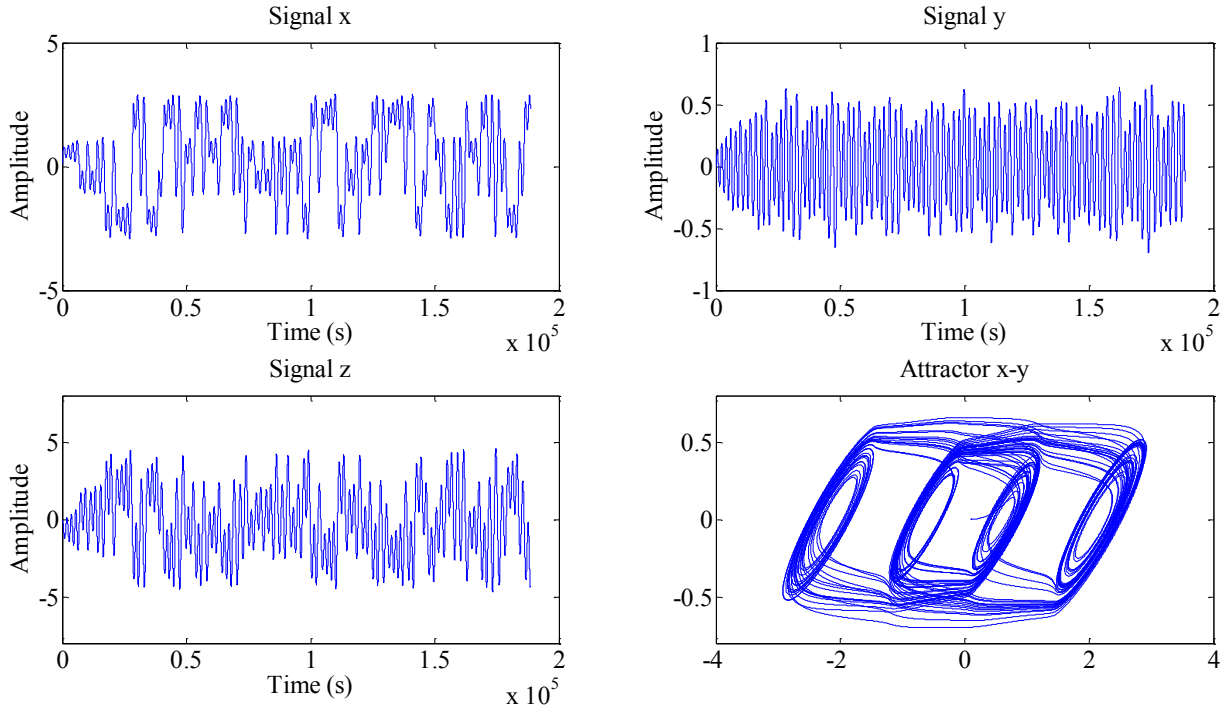

Figura 3.13. Señales Chua con cuatro enrollamientos resuelto por método de Euler.

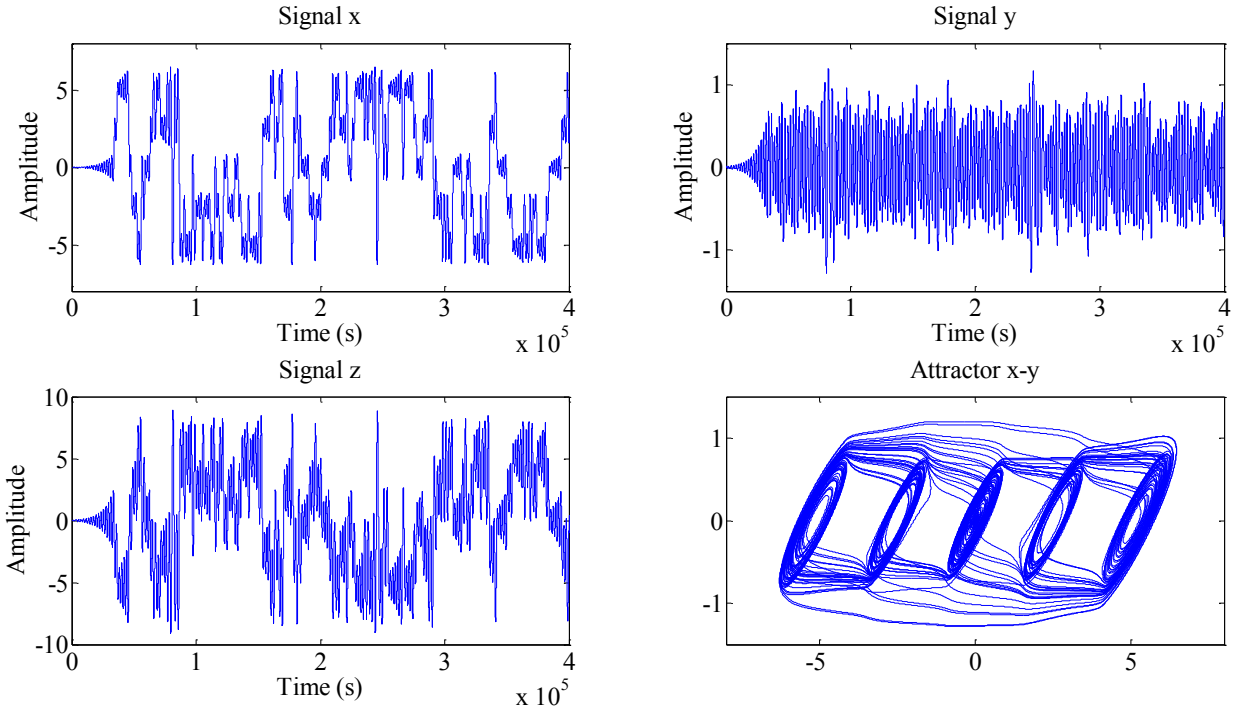

Figura 3.14. Señales Chua con cinco enrollamientos resuelto por método de Euler.

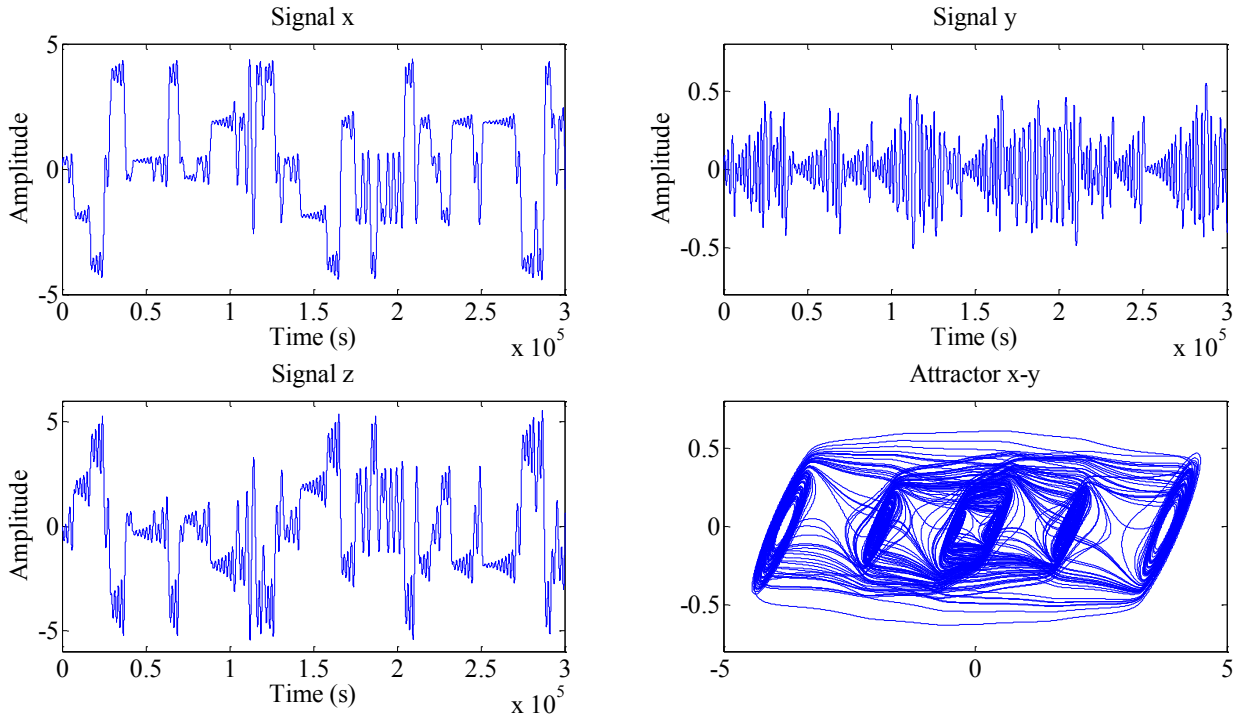

Figura 3.15. Señales Chua con seis enrollamientos resuelto por método de Euler.

3.3.2 Solución mediante el método de Runge-Kutta orden 4

En esta subsección se presenta los resultados de las simulaciones en MATLAB de los casos de multi-enrollamientos del oscilador caótico basado en el circuito de Chua resueltos mediante el método de Runge-Kutta de orden 4, se utilizan los mismos valores de las constantes de la tabla 3.1, se realizaron los archivos *.m* para cada caso de multi-enrollamientos, los cuales generan 2, 3, 4, 5 y 6 enrollamientos respectivamente. Los resultados de la simulación de cada caso se muestran en las figuras (3.16-3.20), en cada caso se muestran las señales *x*, *y*, *z* y el atractor *x-y*.

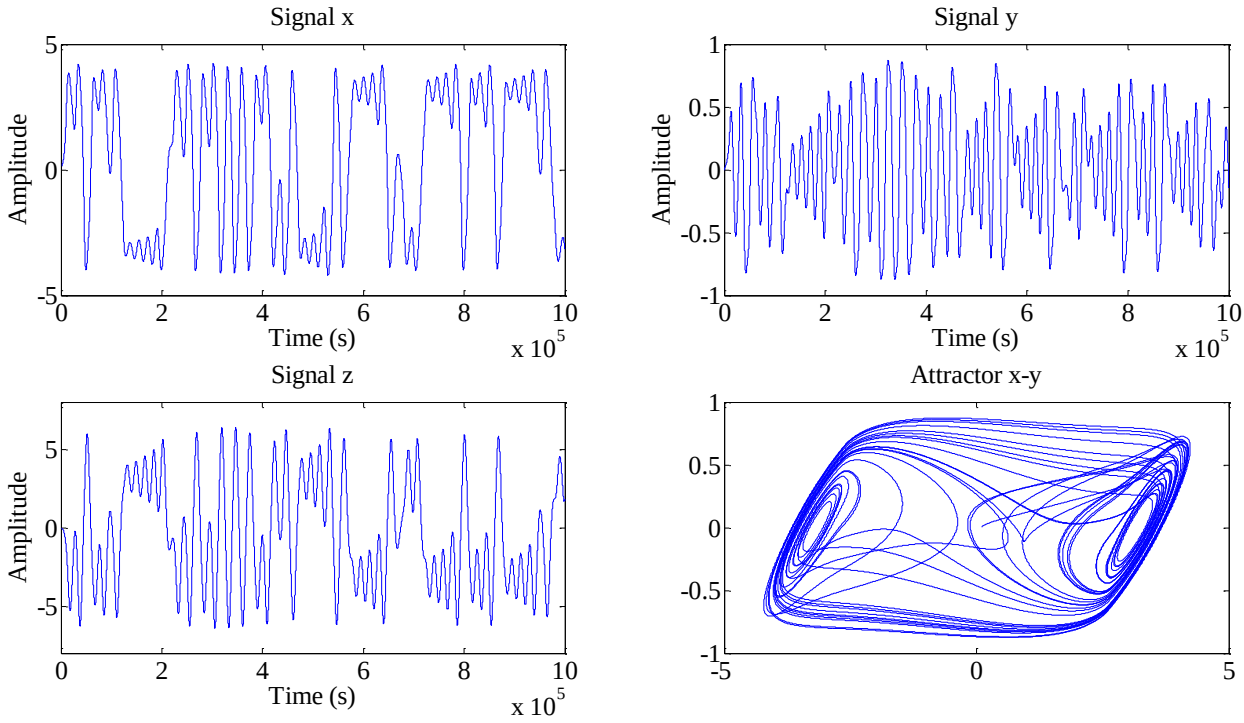

Figura 3.16. Señales Chua con dos enrollamientos resuelto por método de RK4.

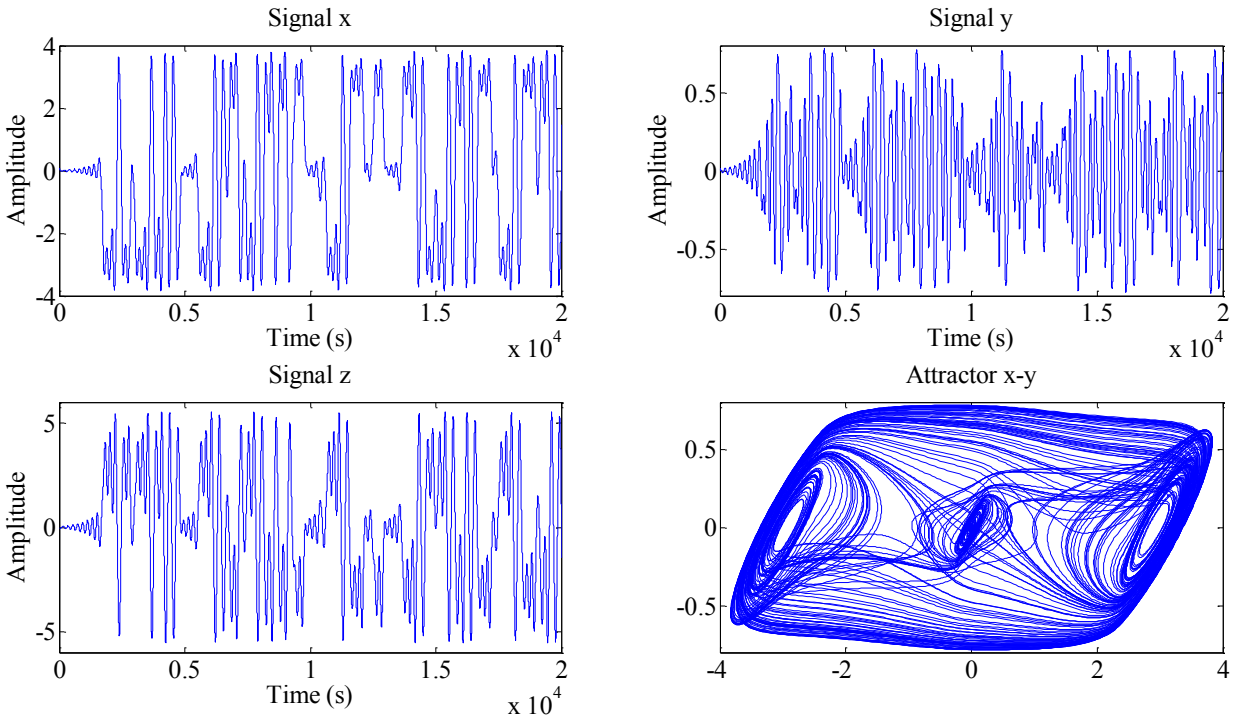

Figura 3.17. Señales Chua con tres enrollamientos resuelto por método de RK4.

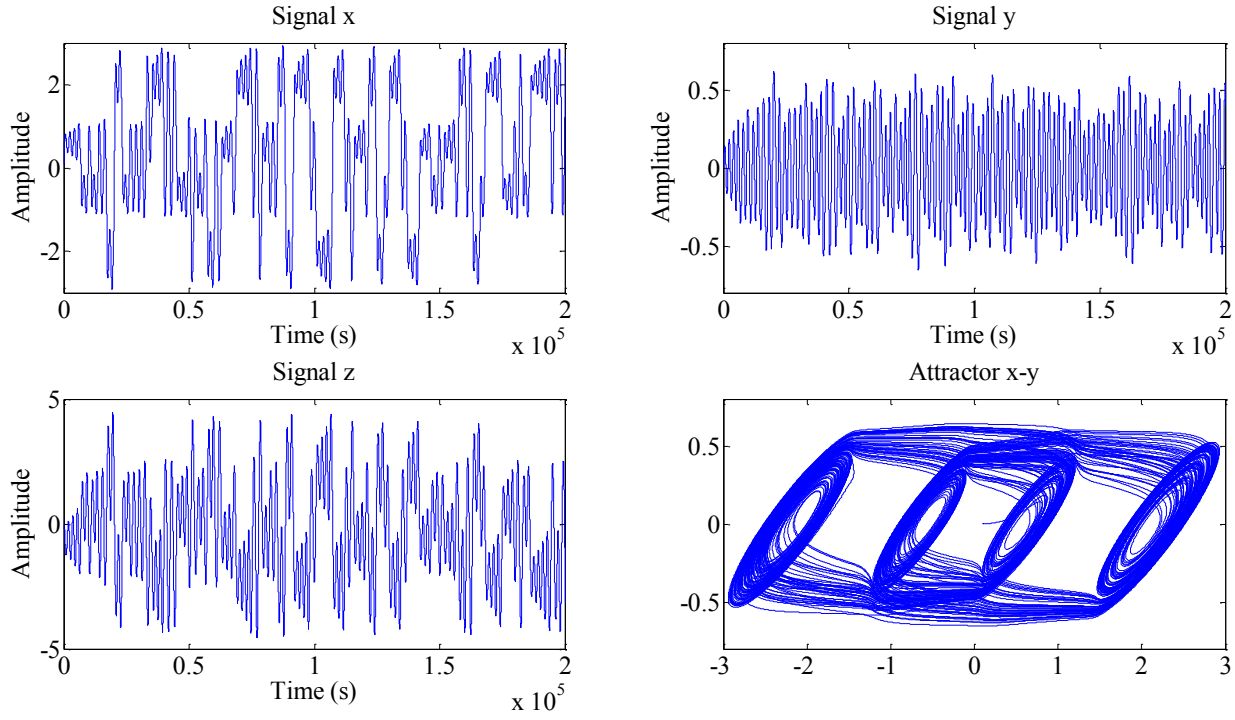

Figura 3.18. Señales Chua con cuatro enrollamientos resuelto por método de RK4.

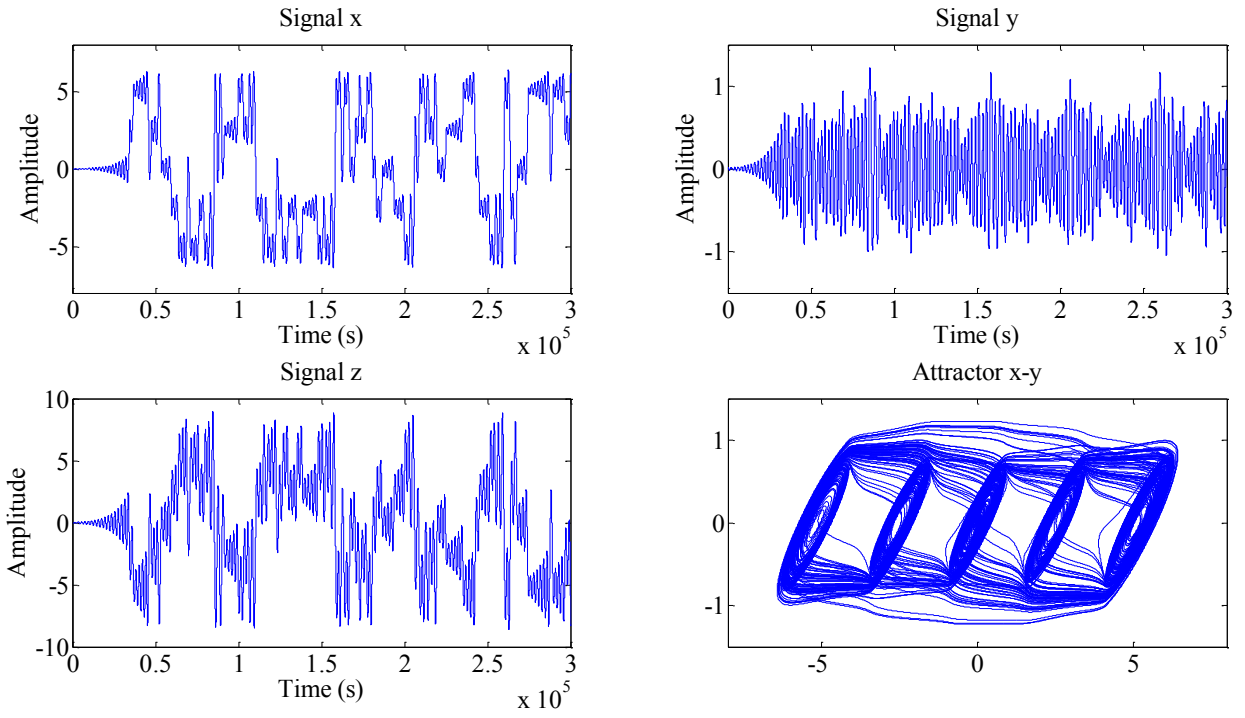

Figura 3.19. Señales Chua con cinco enrollamientos resuelto por método de RK4.

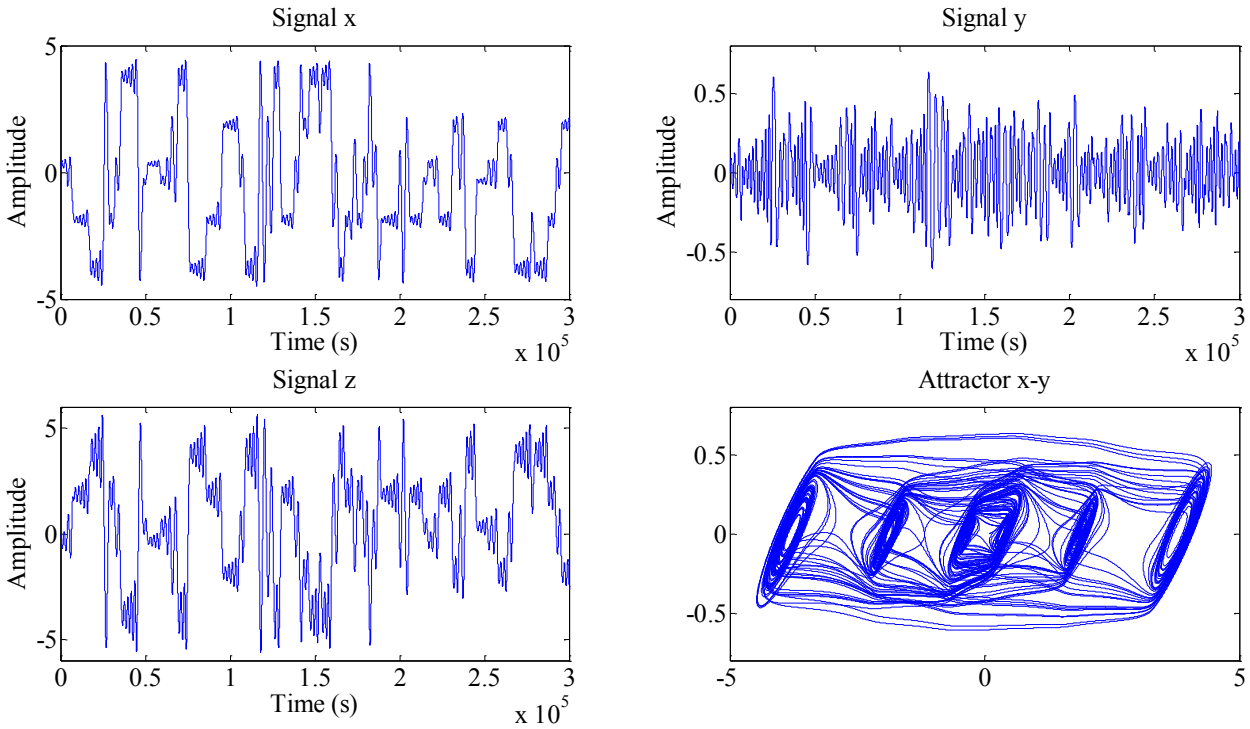

Figura 3.20. Señales Chua con seis enrollamientos resuelto por método de RK4.

3.4 SIMULACIÓN DE OSCILADOR CAÓTICO BASADO EN EL CIRCUITO DE CHUA EN SIMULINK

Como una forma alternativa de simular un sistema caótico en Matlab, se utilizó la herramienta Simulink para la realización del oscilador caótico basado en el circuito de Chua. Se consideran los valores de los parámetros de [51], $\alpha = 10$, $\beta = 100/7$, $m_0 = -8/7$, $m_1 = -5/7$, los valores iniciales $x_0 = 0.7$, $y_0 = 0$ y $z_0 = 0$.

La figura 3.21 muestra la implementación del oscilador caótico, donde se observa la retroalimentación entre los estados *x*, *y*, *z*, creando así las ecuaciones obtenidas y normalizadas (33)-(35) con ayuda de los bloques de suma, resta y los bloques de ganancias para la representación de las constantes. También se muestra la inserción de la curva característica del diodo de Chua "función del diodo" con ayuda del bloque "función de Matlab" donde se utiliza un archivo .m que describe mediante código a la ecuación (36). Cabe recalcar que se utiliza el método de solución numérica ode45 con una tolerancia relativa de $1x10^{-8}$. En cada iteración del método se genera un dato de las soluciones de las variables *x*, *y*, *z*, a partir de las condiciones iniciales. Dado que se trata de un sistema retroalimentado, después de cada iteración se va obteniendo como salida del sistema la solución para cada variable, posteriormente estas salidas son tomadas como las nuevas condiciones iniciales del algoritmo en la siguiente iteración. En este caso se generaron 4441 datos para cada solución, en un tiempo de simulación de 300 segundos. Cada solución de las variables de estado es enviada a la salida del modelo implementado para su visualización y se almacena igualmente en el espacio de trabajo de Matlab.

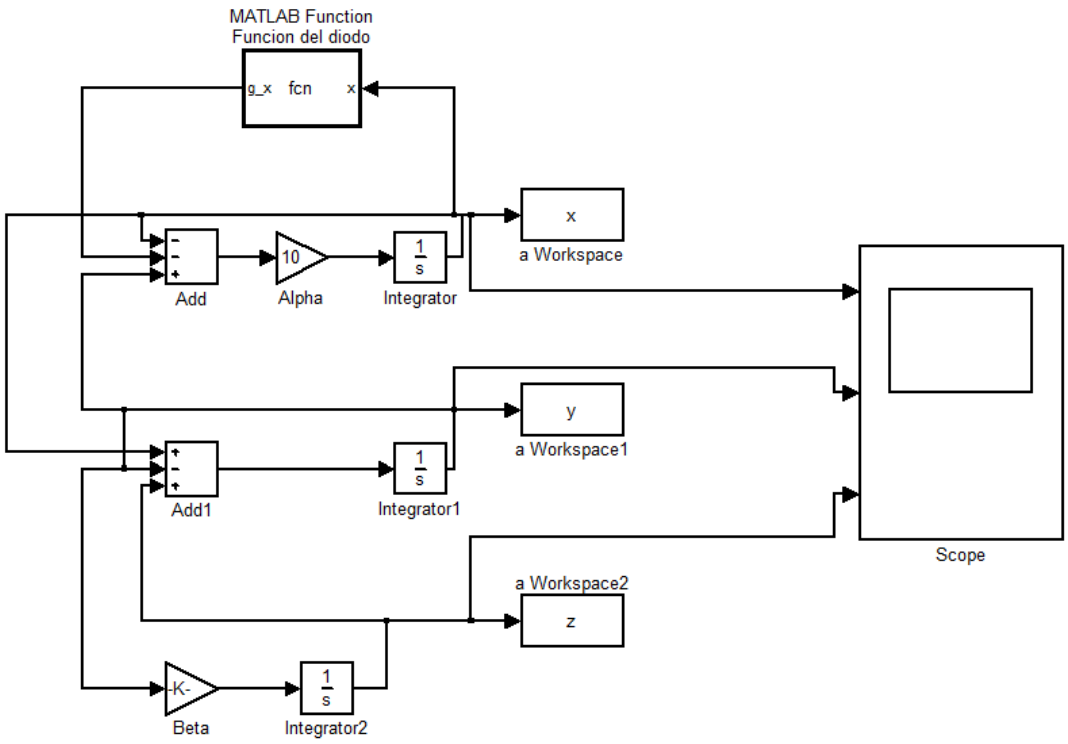

Figura 3.21. Implementación de oscilador caótico en Simulink.

El sistema genera tres tipos de planos para graficar las secuencias de los comportamientos caóticos. La figura 3.22 presenta la gráfica representativa de cada variable de estado. Las figuras 3.23, 3.24 y 3.25 representan gráficamente las combinaciones de dos variables de estado en dos dimensiones (2D). La figura 3.26 muestra la gráfica las tres variables de estado en tres dimensiones (3D). Las representaciones graficas del sistema muestran como los estados de cada variable describen un patrón complejo y no repetitivo, lo cual representa la aleatoriedad del sistema y el resultado esperado. Al graficar en par el comportamiento caótico que se presenta en las variables de estado se obtienen los estados de fase, admirando así los atractores de doble enrollamiento. Se puede notar que los atractores están localizados en los valores dados como puntos de quiebre $\pm P$.

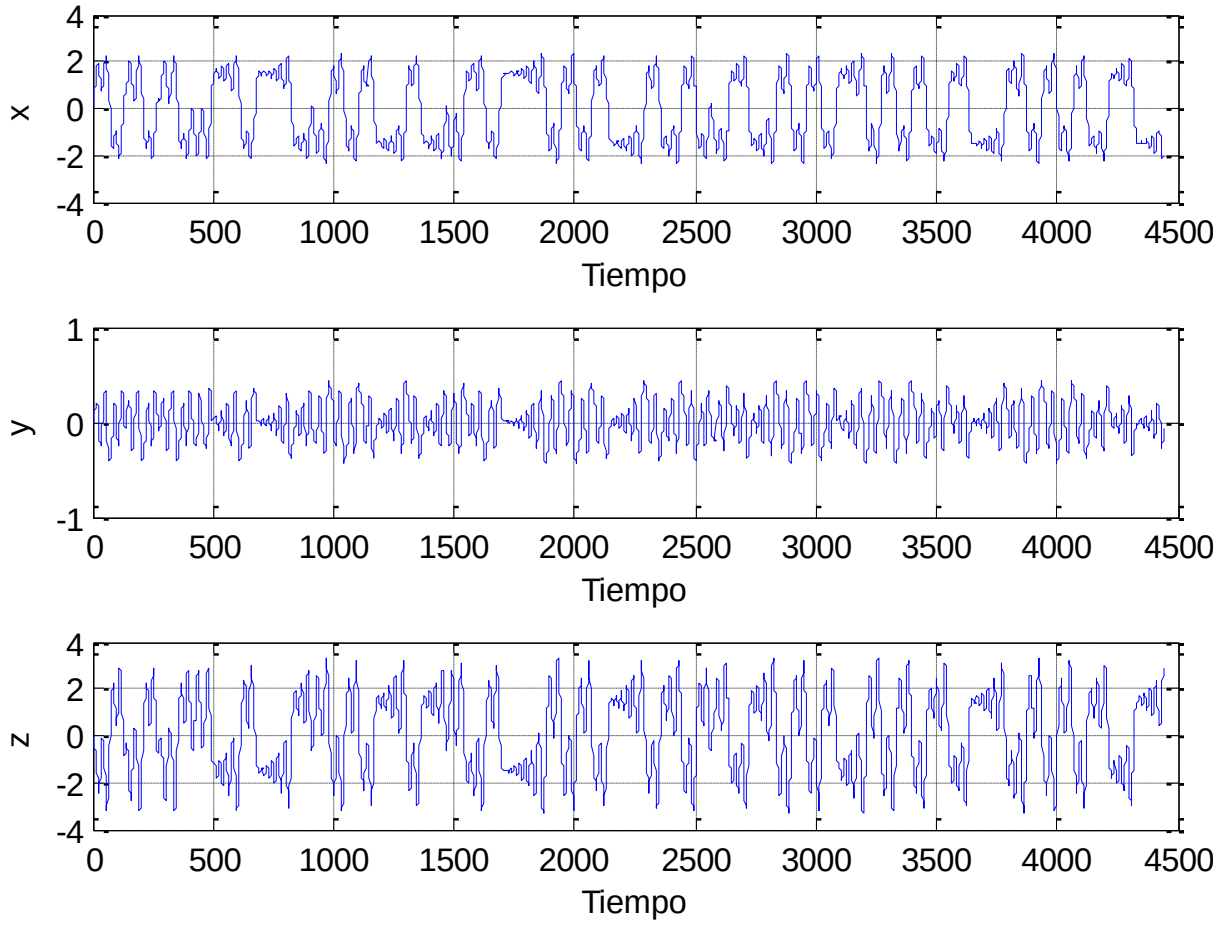

Figura 3.22. Señales *x*, *y*, *z* generadas por el modelo en Simulink.

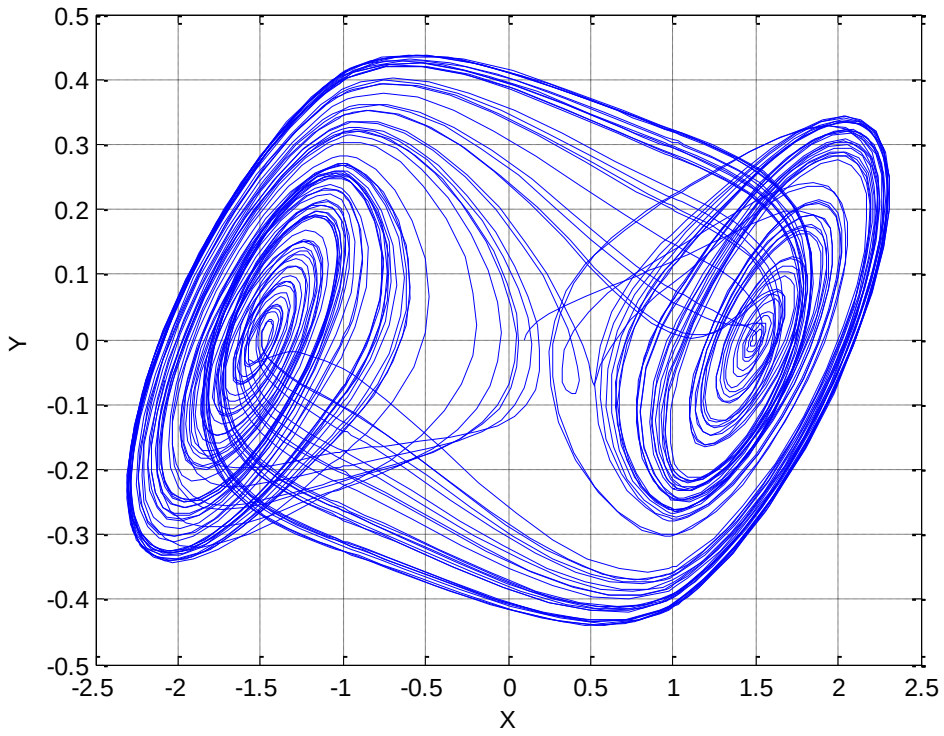

Figura 3.23. Atractor *x-y* en 2D en Simulink.

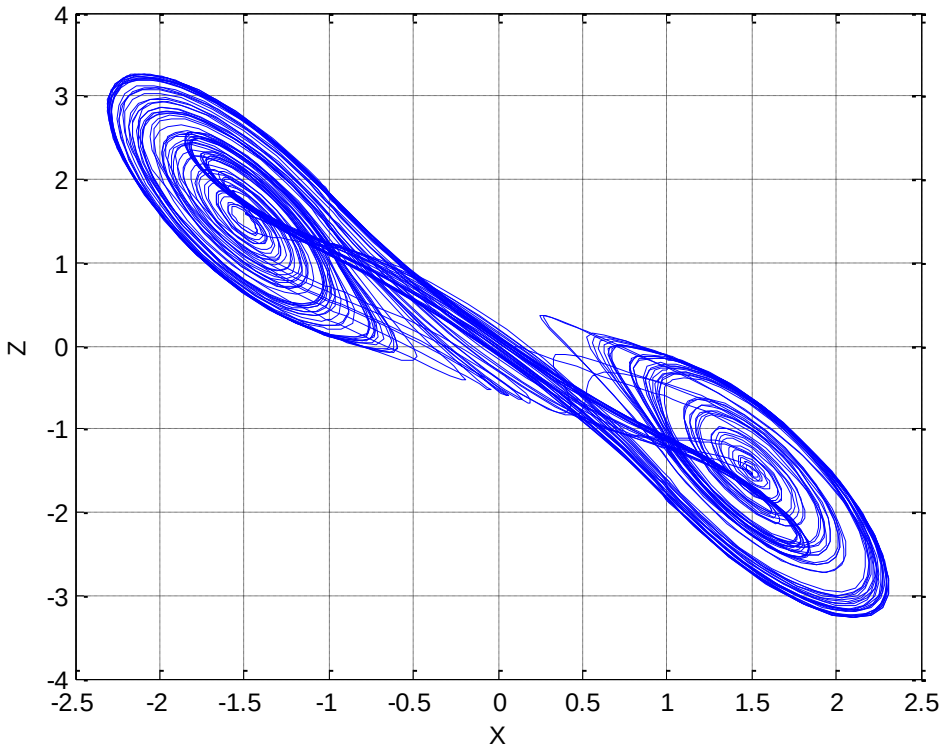

Figura 3.24. Atractor *x-z* en 2D en Simulink.

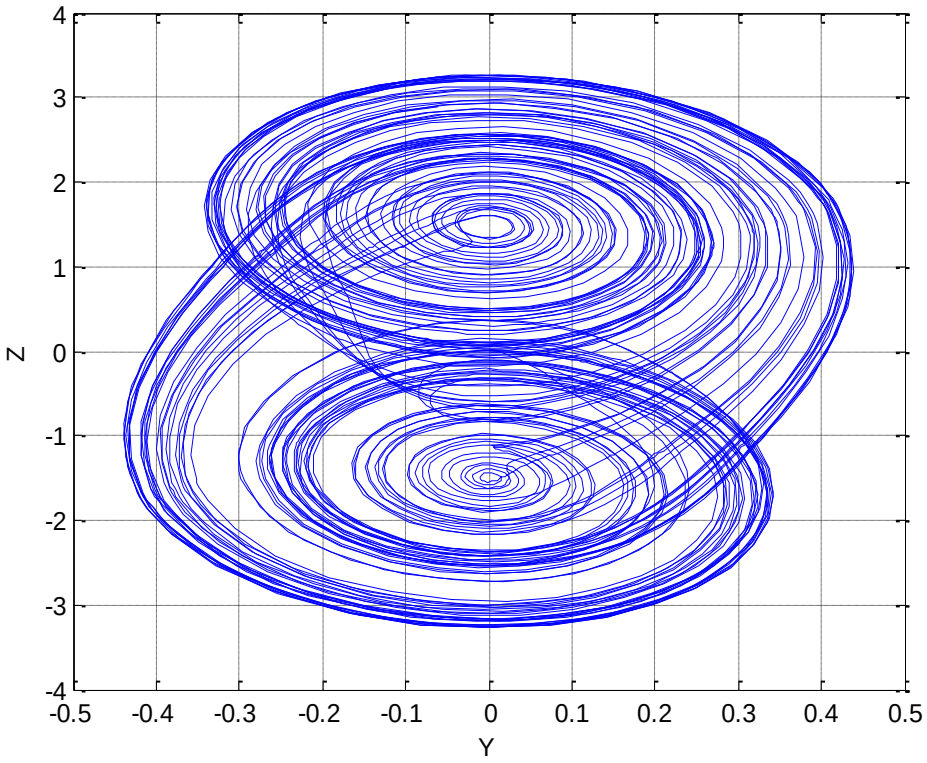

Figura 3.25. Atractor *y-z* en 2D en Simulink.

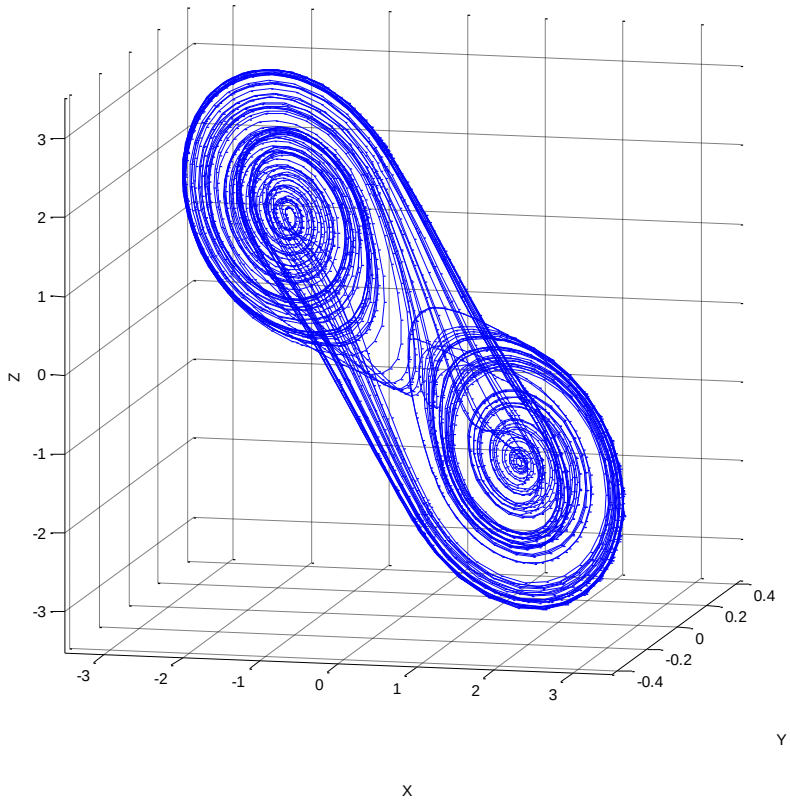

Figura 3.26. Atractor *x-y-z* en 3D en Simulink.

3.5 CONCLUSIÓN

En este capítulo se mostró el diseño de multi-enrollamientos de los osciladores caóticos basados en SNLF y el circuito de Chua, se muestran los resultados de las simulaciones en el software matemático MATLAB, resolviendo ambos sistemas mediante el método de Euler y Runge-Kutta de orden 4, se puede notar que a simple vista no se ve la diferencia de utilizar el método de Euler y Runge-Kutta, sin embargo se sabe que el método de Runge-Kutta es más preciso que el método de Euler. Además se mostró una forma alternativa de simular el oscilador caótico basado en el circuito de Chua utilizando la herramienta Simulink.

Capítulo 4

Implementación de osciladores caóticos en FPGA

4.1 INTRODUCCIÓN

En este capítulo se lleva cabo las descripción de la metodología empleada para realizar la implementación en el dispositivo FPGA-DSP Cyclone III Edition de Altera del oscilador caótico de Chua mediante la herramienta DSP-Builder y Simulink. Además se muestra otra metodología para la implementación en FPGA para los osciladores caóticos basados en SNLF y el circuito de Chua a través del diseño de arquitecturas mediante código VHDL, ambos tipos de osciladores resueltos mediante el método de Euler y Runge-Kutta, se muestran los resultados en co-simulación y experimentales.

4.2 PLATAFORMA DSP-FPGA

En este marco de tesis, se usa una de las tecnologías más eficientes para el diseño de sistemas digitales, el kit de desarrollo DSP- FPGA Cyclone III Edition de Altera, la cual está orientada a aplicaciones que requieren procesamiento digital de señales y además incluye la herramienta DSP Builder. A través de un diseño de bloques permite implementar un oscilador caótico en Matlab/Simulink, genera un código complejo en VHDL, el cual es usado posteriormente para configurar la tarjeta DSP-FPGA, que permite un ambiente de diseño basado en modelos de Simulik, con la ventaja de que no es necesario contar con experiencia previa con FPGAs de Quartus, ya sea una metodología de diseño VHDL o RTL. Esta herramienta realiza la generación del código VHDL necesario para la implementación en la FPGA con bloques específicos de Altera. El ejemplo del oscilador caótico tomado en este trabajo se basa en el comportamiento de un Circuito de Chua.

El kit de desarrollo Cyclone III también cuenta con una tarjeta de adquisición de datos HSMC.
La interfaz HSMC proporciona un mecanismo para extender el conjunto de periféricos de una tarjeta FPGA por medio de tarjetas intermedias que funcionan como conectores o llamado también puerto HSCM en el dispositivo FPGA.

En la figura 4.1 se muestra el montaje de la tarjeta de adquisición HSMC que está conectada al puerto A de la tarjeta de desarrollo FPGA Cyclone III.

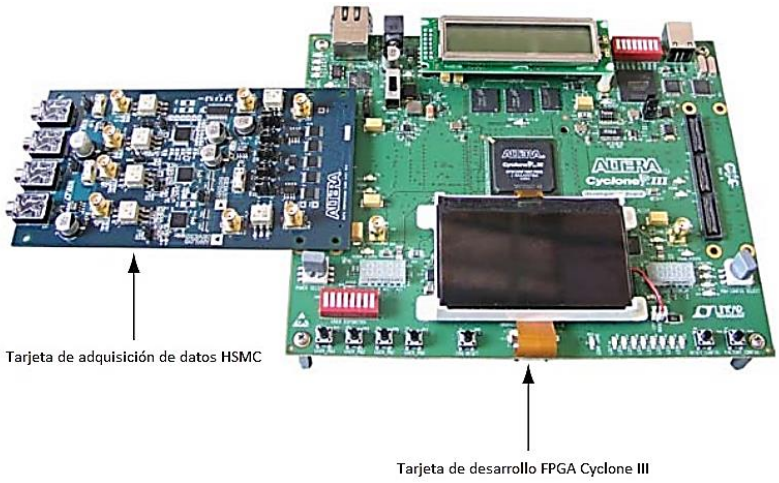

Figura 4.1. Unión de tarjetas DSP-FPGA

4.3 METODOLOGÍA DE DISEÑOS DIGITALES EN FPGAS.

La metodología de diseño es similar a la cualquier sistema digital, salvo que al final se obtiene un archivo ejecutable que se descarga al FPGA para que se reconfigure, implementando así el diseño esperado. Primer hay que tener una descripción del circuito a realizar. Tradicionalmente en las ingenierías se realizan planos o esquemas para esta descripción, de forma similar a como un arquitecto diseña un edificio. Sin embargo es posible realizar una descripción del hardware utilizando algún lenguaje de descripción de hardware, como VHDL o Verilog Con esta descripción se pueden realizar simulaciones del circuito, para comprobar que lo diseñado trabaja correctamente de lo contrario se volverá a modificar la descripción (esquemas o programa) hasta que la simulación sea satisfactoria. Hasta aquí sólo se ha utilizado el computador y no se ha tocado hardware. Sin embargo en el caso del software, la propia simulación es la ejecución del programa. Se observa directamente el resultado del programa y se modifican el código fuente hasta que se eliminen los errores.
En el caso del hardware hay que construir el circuito. Y aquí es donde vienen los FPGA para hacerlo. A partir de la especificación hardware y utilizando un compilador especial, se obtiene un archivo binario, llamado bitstream que contiene toda la información necesaria para configurar el FPGA. Este archivo, que es el equivalente a un programa ejecutable en el caso del software, es el que hay que cargar en el FPGA. Se carga este archivo en el FPGA y listo. Ya se tiene el hardware que se quería situado en el interior de un chip. No se ha tenido que soldar, ni comprar componentes, ni perder tiempo haciendo un prototipo. Ahora los cambios en el diseño se pueden hacer igual de rápidos que en el caso de software. Sólo hay que cambiar la especificación del diseño, volver a compilar y reconfigurar el FPGA con el nuevo bitstream generado.

4.4 DSP BUILDER.

DSP Builder es una herramienta de desarrollo de procesamiento digital de señales (DSP) que hace de interfaz entre el software Quartus II y las herramientas de Matlab/Simulink. Para ello integra

estas características combinando el desarrollo de algoritmos, simulación, y la capacidad de verificación de las herramientas de diseño a nivel sistema de Matlab/Simulink con síntesis VHDL, simulación y herramientas de desarrollo de Altera. DSP Builder acorta el tiempo dedicado al diseño, ya que ayuda a crear la representación de hardware de un diseño de DSP en un ambiente de desarrollo para crear los algoritmos.
La ventaja de esta herramienta es que simplifica la implementación de hardware de funciones DSP, proporcionando una herramienta de verificación a nivel sistema para el usuario que no esté necesariamente familiarizado con el flujo de diseño en HDL, permitiendo así implementar funciones DSP en un FPGA sin la necesidad de aprender HDL [52].

La metodología de configuración del dispositivo DSP-FPGA utilizada con ayuda de la herramienta DSP Builder se aprecia en el diagrama de flujo mostrado en la figura 4.2 y se explica con más detalle a continuación.

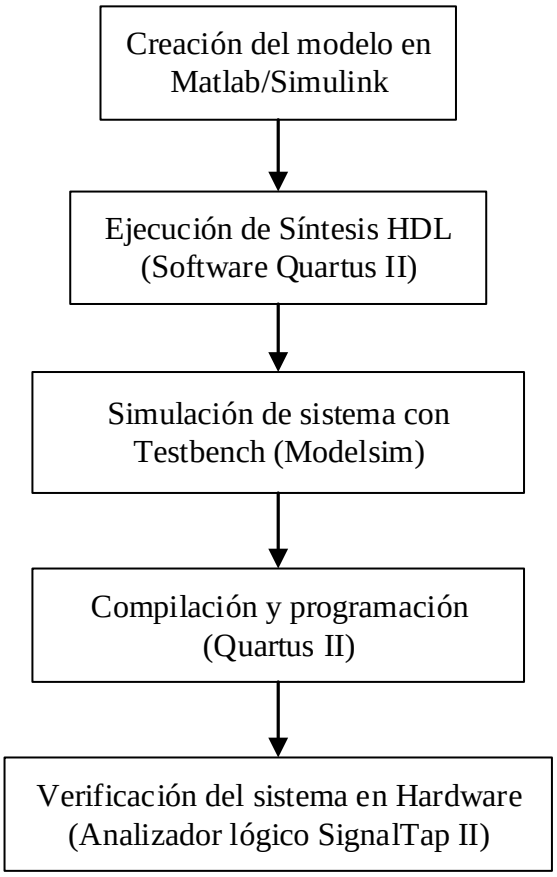

Figura 4.2. Diagrama de flujo del diseño a nivel sistema de DSP Builder

Los bloques son descritos a continuación:

- Creación del diseño del modelo en Simulink: Se el modelo en Simulink con ayuda de los elementos y librerías incluidas en la herramienta DSP Builder.

- Síntesis: Se recoge el diseño (.mdl) para transformarlo a código VHDL como un diseño netlist de alguna arquitectura especifica.
- Generar archivos: Permite crear los archivos necesarios para la simulación y programación de la tarjeta DSP-FPGA a partir de la compilación del proyecto generado con la síntesis en el software Quartus II.
- Simulación: Se verifica la funcionalidad del diseño en varios puntos durante el flujo de diseño utilizando una herramienta de simulación. Dentro del entorno de Quartus de Altera, se utiliza ModelSim pero alternativamente se puede simular el diseño usando cualquier otro simulador compatible.
- Programador: Se programa el diseño utilizando los archivos generados de la compilación en Quartus II en el dispositivo DSP-FPGA.
- Análisis de la implementación. Después de la implementación se puede analizar el diseño con respecto a los resultados obtenidos en Matlab/Simulink. Se observan la utilización de recursos del dispositivo, el rendimiento del tiempo, y utilización de la energía.

Una de las grandes ventajas es que a partir del análisis de los resultados del diseño, se pueden hacer cambios para diseñar nuevas señales, propiedades del proceso, cambiar parámetros de simulación, y a continuación, volver a sintetizar e implementar si así se le requiere.

4.5 IMPLEMENTACIÓN DE OSCILADOR CAÓTICO BASADO EN EL CIRCUITO DE CHUA CON DSP BUILDER.

En esta sección se realiza el proceso de acondicionamiento de las señales de salida del modelo implementado en MATLAB/Simulink mostrado en la figura 3.29, para lo cual se utiliza la herramienta DSP Builder en Simulink para la generación del código HDL y los proyectos para los programas informáticos Quartus II, Timequest, QSYS y ModelSim, todos pertenecientes a Altera.

La implementación con DSP Builder se muestra en la figura 4.3, debido a que el dispositivo FPGA solo cuenta con dos conectores SMA para salidas del convertidor D/A (Convertidor Digital a Analógico), solo fue posible visualizar dos señales del sistema caótico a la vez. A través del uso del conjunto de bloques de la herramienta DSP Builder en Simulinkse retomaron el par de señales (x, y) que estaban almacenadas en el “workspace” (espacio de trabajo) de MATLAB obtenidas del modelo mostrado en la figura 3.29, para su conversión a señales digitales, repitiendo este procedimiento para los demás pares: (x, z) y (y, z). Las señales fueron muestreadas en palabras de 12 bits, obteniendo un total de 4096 datos, añadiéndole además 2 bits para el signo, cada palabra fue almacenada en una tabla de memoria (LUT), las cuales son posteriormente la salida de los convertidores D/A del dispositivo, el diagrama de bloques en Simulink del procedimiento anterior se muestra en la figura 4.3.

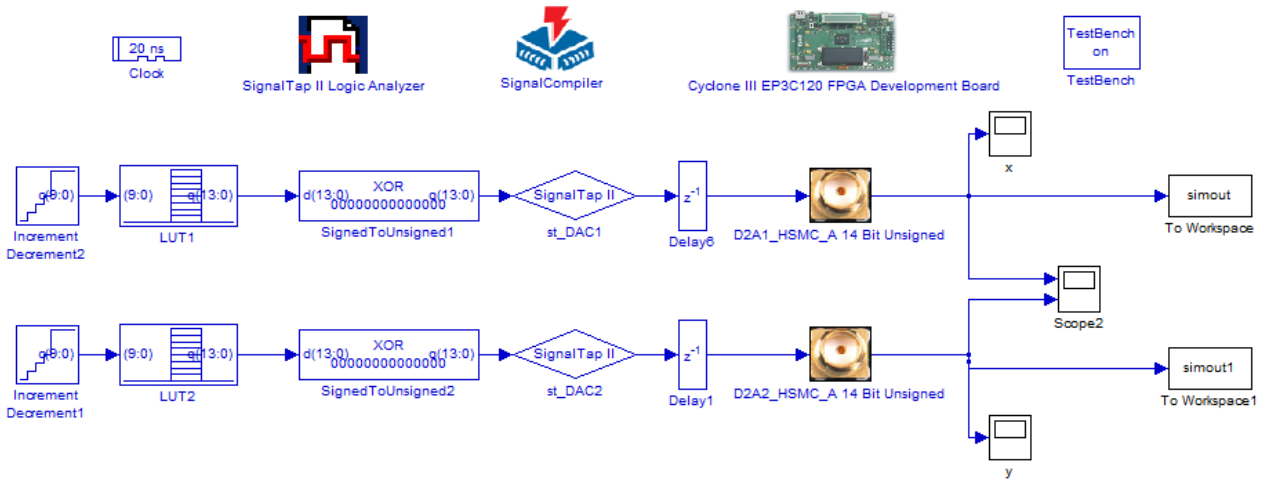

Figura 4.3. Implementación del oscilador caótico de Chua con DSP Builder.

Las frecuencias de las señales obtenidas son de 1.12 MHz. En las figuras 4.4, 4.5y 4.6 se presentan las señales individuales en tiempo real de *x*, *y* y *z*, respectivamente. Se puede visualizar que las señales de salida de la tarjeta DSP-FPGA cuentan con las mismas características que las señales generadas en Matlab/Simulink mostradas en la figura 3.30 del capítulo anterior.

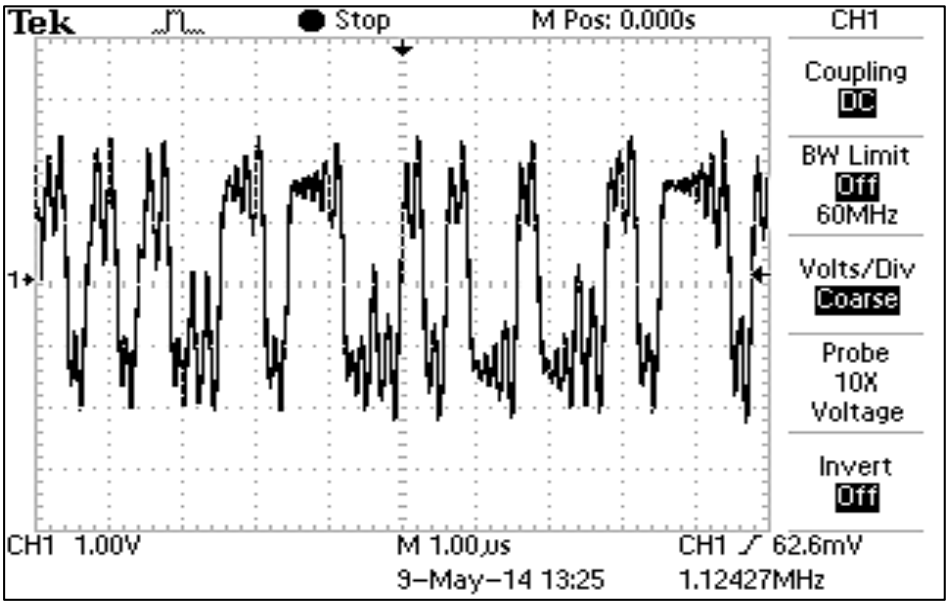

Figura 4.4. Señal *x* del oscilador caótico de Chua en tiempo real implementado con DSP Builder.

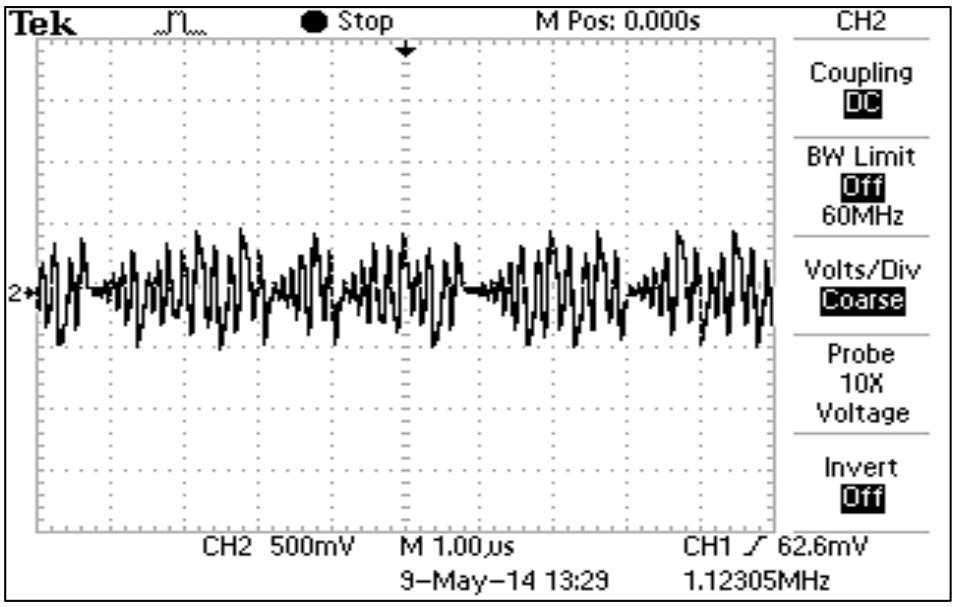

Figura 4.5. Señal *y* del oscilador caótico de Chua en tiempo real implementado con DSP Builder.

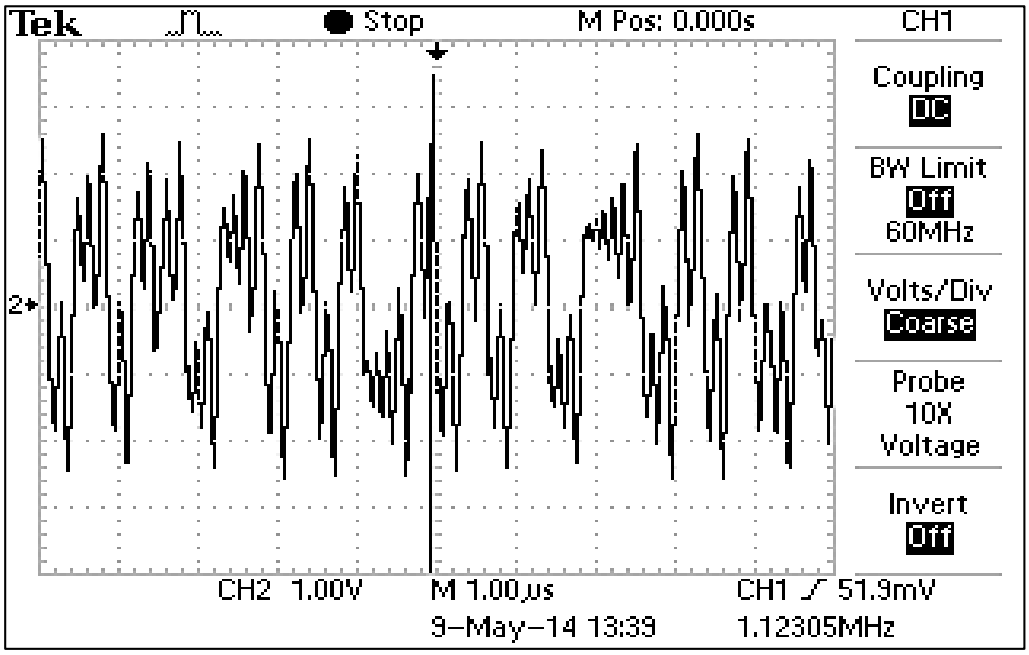

Figura 4.6. Señal *z* del oscilador caótico de Chua en tiempo real implementado con DSP Builder.

Para visualizar los enrollamientos de una señal con respecto a la otra es necesario establecer el formato en modo X-Y en el osciloscopio, de esta manera se pueden visualizar las posibles combinaciones de dos variables de estado en dos dimensiones 2D. Las señales de salida, medidas en el osciloscopio, para los atractores (*x*, *y*), (*x*, *z*) y (*y*, *z*) son mostradas en las figuras 4.7, 4.8 y 4.9, respectivamente.

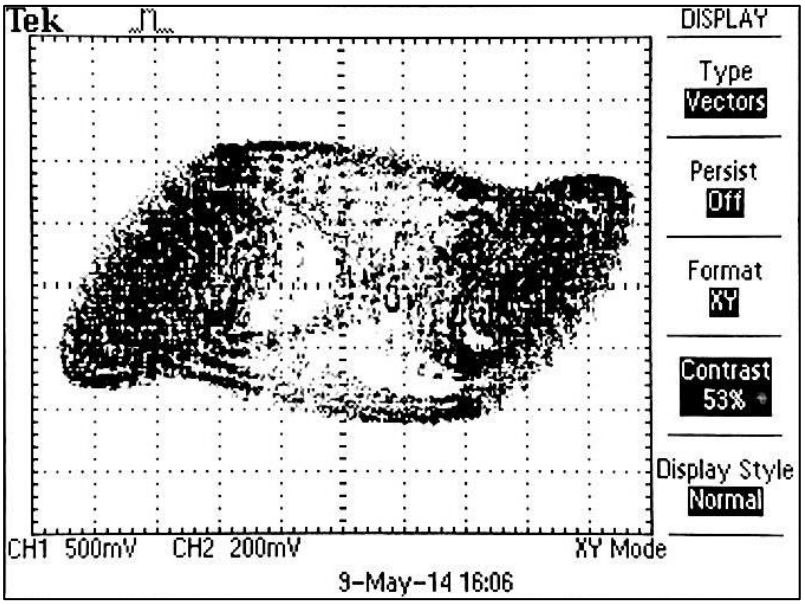

Figura 4.7. Atractor *x-y* del oscilador caótico de Chua en tiempo real implementado con DSP Builder.

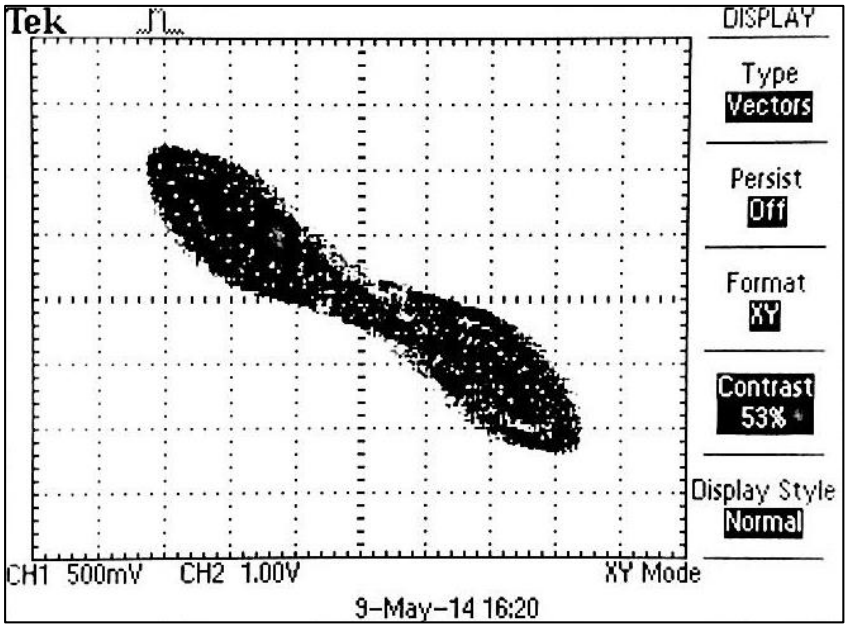

Figura 4.8. Atractor *x-z* del oscilador caótico de Chua en tiempo real implementado con DSP Builder.

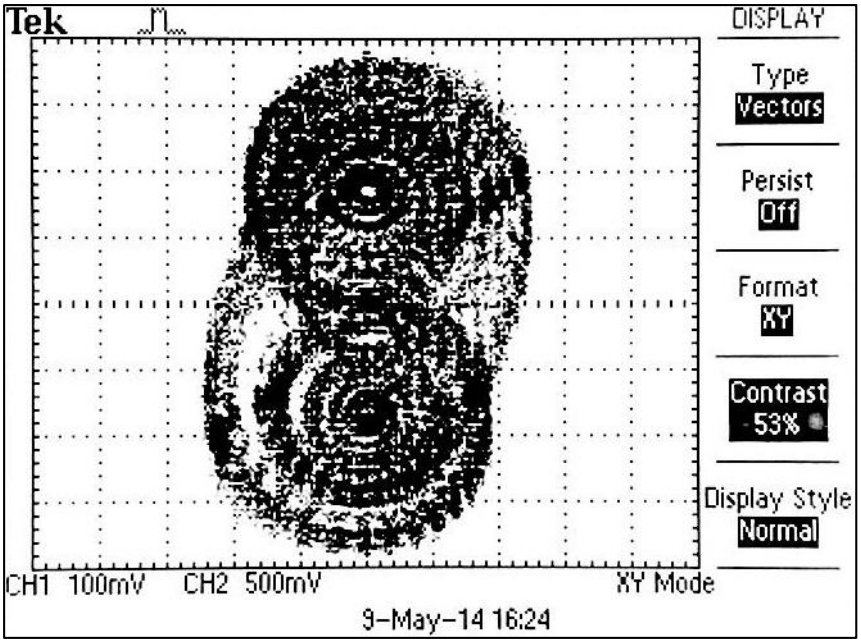

Figura 4.9. Atractor *y-z* del oscilador caótico de Chua en tiempo real implementado con DSP Builder.

Se puede observar que los atractores obtenidos experimentalmente son similares a los atractores generados en Matlab/Simulink mostradas en las figuras 3.31, 3.32 y 3.33 del capítulo anterior.

4.6 IMPLEMENTACIÓN DE OSCILADOR CAÓTICO BASADO EN SNLF MEDIANTE CÓDIGO VHDL.

En esta sección se presenta la metodología que se siguió para la implementación del oscilador caótico basado en SNLF, diseñando una arquitectura a partir de los métodos numéricos de Euler y Runge-Kutta, esta arquitectura fue diseñada mediante código VHDL para su co-simulación y su posterior implementación en FPGA.

4.6.1 Arquitectura en base al algoritmo de Euler.

Con el propósito de discretizar el sistema, se aplicó el método de Euler al sistema de ecuaciones (22). La mayoría opta por discretizar sistemas continuos usando el método de Runge-Kutta, de tercer o cuatro orden, debido a que es más exacto al hacer el cómputo repetidas veces sobre los resultados obtenidos. Sin embargo, para la explicación de la metodología del diseño de la arquitectura que resuelva el sistema caótico se eligió el método de Euler hacia delante por ser muy fácil de utilizar, principalmente para personas que inician con procesos de discretización, y porque permite obtener la solución deseada si necesidad de gran exactitud. Así mismo, al ser un algoritmo de fácil aplicación, permite al dispositivo FPGA tener una mayor velocidad de procesamiento, mientras que con Runge-Kutta puede existir una mayor complejidad de cómputo, debido a las excesivas operaciones que esta contiene.

A partir de una condición inicial, la solución a un sistema de ecuaciones diferenciales ordinarias se puede encontrar aplicando la formula iterativa de Euler descrita en (49), donde *step* sustituye a *h* que es el tamaño de paso de integración numérica.

Aplicando entonces la formula iterativa de Euler (49) al sistema de ecuaciones (22), se obtiene el siguiente conjunto de ecuaciones discretas:

$$\begin{aligned} x(i+1) &= x(i) + step * y(i) \\ y(i+1) &= y(i) + step * z(i) \\ z(i+1) &= z(i) + step * \left(-a * x(i) - b * y(i) - c * z(i) + d_1 f(x(i); k, \alpha, p, q)\right), \end{aligned} \tag{71}$$

aquí $x(i)$, $y(i)$ y $z(i)$ conforman el estado del sistema en tiempo discreto, donde los parámetros utilizados en el sistema de ecuaciones anterior son las mismas a excepción del parámetro *step*, que es el tamaño de paso de integración numérica.

En la tercera ecuación de (71) $z(i+1)$, se encuentra implícita la función no lineal saturada (SNLF) *f(x(i); k, h, p, q)*, esta ecuación depende de dicha función, por lo que dependiendo del valor que tenga $x(i)$ será el valor que tendrá $f(x(i))$. A fin de abordar con mayor simplicidad la explicación, se tomará la SNLF para dos enrollamientos mostrada en la figura 2.11, y su descripción PWL (24). Esta función se describe por tres condiciones, por lo cual la ecuación $z(i+1)$ del sistema de ecuaciones (71) tendrá tres variantes, como se muestra en la tabla 4.1

Tabla 4.1. Condiciones para ecuación $\boldsymbol{z(i+1)}$.

Condición	$\boldsymbol{z(i+1)}$
$\boldsymbol{x(i) > \alpha}$	$z(i) + step * (-a * x(i) - b * y(i) - c * z(i) + d_1 k)$
$\boldsymbol{-\alpha \leq x \leq \alpha}$	$z(i) + step * \left(-a * x(i) - b * y(i) - c * z(i) + d_1 sx(i)\right)$
$\boldsymbol{x(i) < -\alpha}$	$z(i) + step * (-a * x(i) - b * y(i) - c * z(i) - d_1 k)$

Para la implementación del sistema de ecuaciones (71) es necesario crear componentes mediante código VHDL que realicen las operaciones matemáticas necesarias (sumador, restador y multiplicador) utilizando palabras de 19 bits con punto fijo, para eso se toman 1 bit para el signo, 4 bits para la parte entera y 14 bits para la parte fraccionaria, definiendo cada palabra tal y como se muestra en la tabla 4.2.

Tabla 4.2. Distribución de palabra de punto fijo.

1 bit	**4 bits**	**14 bits**
0	*0000*	*.00000000000000*
Signo	Parte Entera	Parte Fraccionaria

Las entidades de los componentes de sumador, restador y multiplicador se muestran en la figura 4.13, donde *N1* y *N2* representan las señales de entrada de cada entidad y *N*3 corresponde la salida.

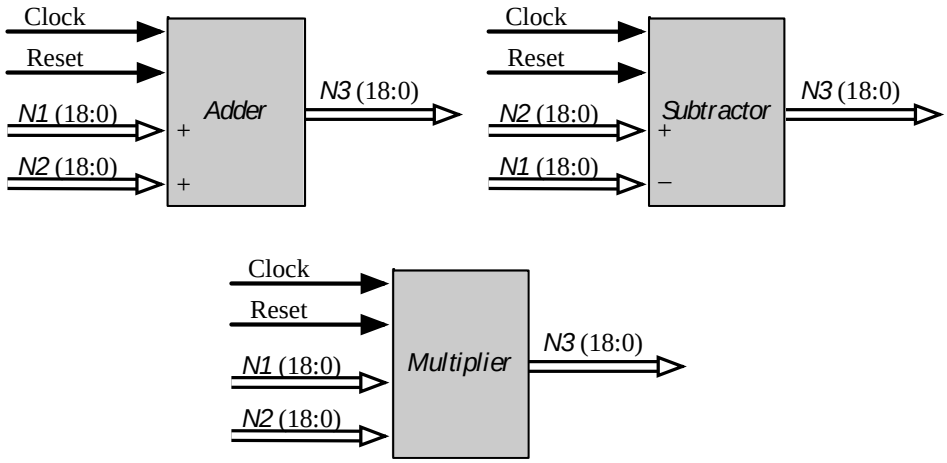

Figura 4.10. Componentes sumador, restador y multiplicador.

Una vez que los componentes sean declarados mediante código VHDL, estos se pueden llamar en el diseño cuando se necesiten mediante el nombre de la entidad y unirse en forma de sistema mediante señales, de esta manera se pueden implementar el sistema de ecuaciones (71). Para el diseño se utilizan los mismos valores usados anteriormente en las simulaciones de MATLAB en la sección 3.2. La implementación del sistema de ecuaciones (71) se muestra en el diagrama de bloques de la figura 4.14.

En la figura 4.11 se puede observar que $x(i)$, $y(i)$ y $z(i)$ representan los estados presentes del sistema y, $x(i+1)$, $y(i+1)$ y $z(i+1)$ los estados futuros. Se muestra también que se hace uso de un comparador, el cual es uno de los componentes más importantes del sistema ya que en él se define el comportamiento de la SNLF, este se realiza a partir de condiciones, para efectos prácticos solo se muestra en diagrama de bloques la implementación para una SNLF de dos enrollamientos debido a su simplicidad, la SNLF para dos enrollamientos fue mostrada anteriormente en la figura 2.11 y fue descrita mediante la ecuación (24), para la implementación de las demás SNLF descritas mediante las ecuaciones (25)-(28) y así obtener más enrollamientos, solamente fue necesario añadir más condiciones al comparador.

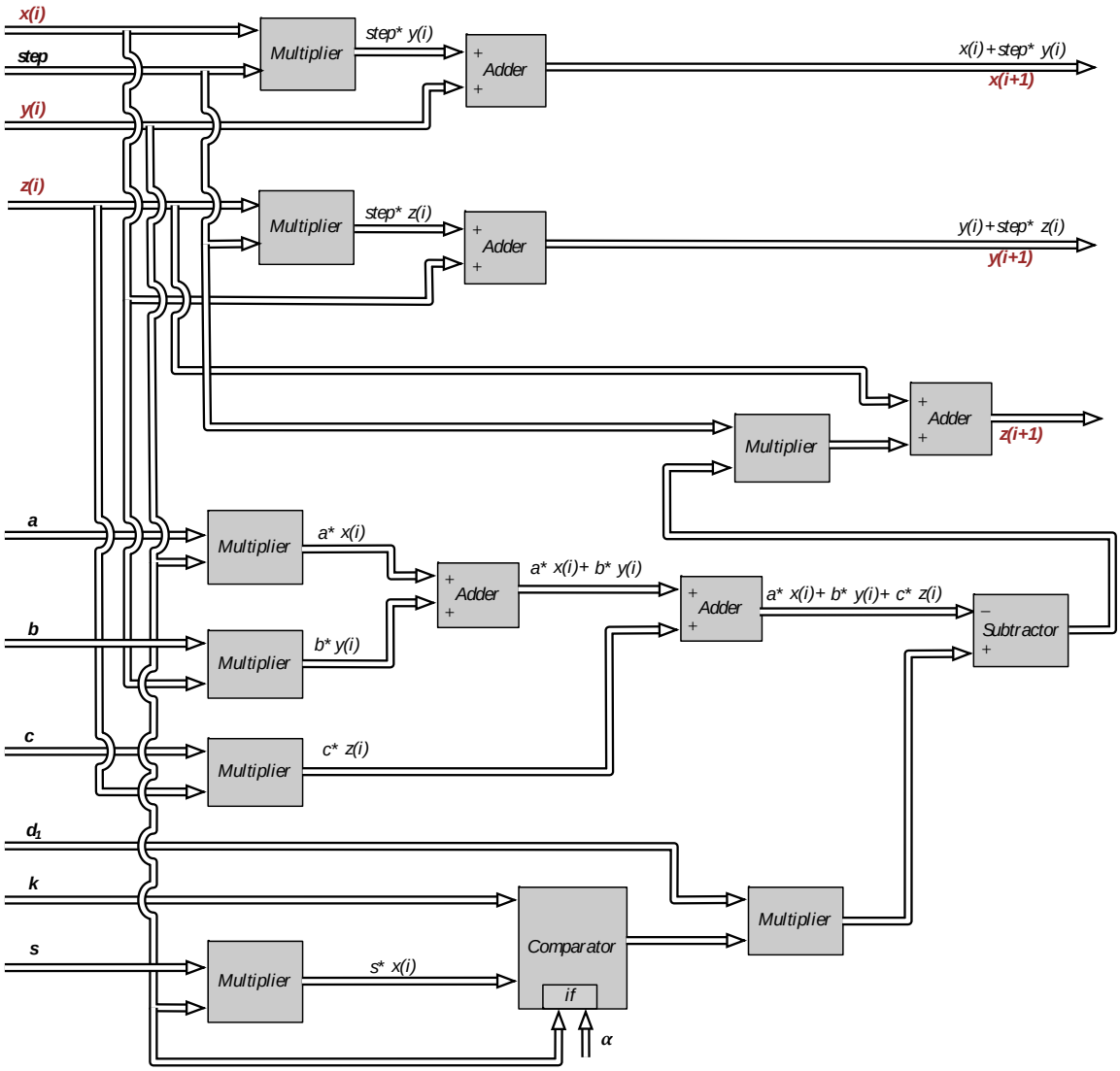

Figura 4.11. Diagrama de bloques del sistema de ecuaciones (48).

El diagrama de bloques de la figura 4.11 muestra así la arquitectura del sistema implementado a partir de las ecuaciones (71), sin embargo este todavía está en su tercer nivel de diseño, al cual se le dio el nombre de Unidad de Oscilador Caótico de SNLF representada en la figura 4.12. En la figura 4.12 *Xini*, *Yini* y *Zini* son las señales de las condiciones iniciales de 18 bits correspondientes a las variables de estado $x(i)$, $y(i)$ y $z(i)$ rescpectivamente, también se observan las señales de entrada *Clock*, *Reset* de 1 bit, las cuales se han utilizado para garantizar la sincronización de los componentes dentro de la unidad.

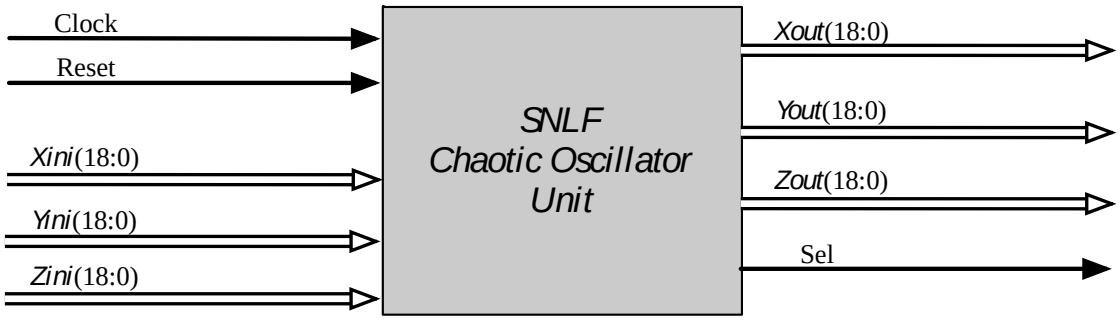

Figura 4.12. Unidad Oscilador Caótico de SNLF.

Si el sistema se implementa así como tal solo se obtendrá la primera iteración de las soluciones, ya que aún no se cuenta con la retroalimentación, para ello se necesita de un Multiplexor. La unidad de Multiplexor se encarga de mantener el sistema en un bucle lo cual permite las iteraciones, esta unidad le proporciona las condiciones iniciales del sistema a la Unidad Oscilador Caótico de SNLF. Estas condiciones iniciales permiten obtener la primera iteración, la cual necesita de 8 ciclos de reloj, después de eso el sistema necesita retroalimentarse, aquí entra la función del Contador que después de 8 ciclos de reloj, se encarga de mandarle la señal *Sel* de un bit al Multiplexor, esta señal sirve para indicar que se empiece la retroalimentación del sistema y que de ahí en adelante las condiciones iniciales serán las mismas salidas *Xout*, *Yout* y *Zout* del sistema caótico.

En la figura 4.13 se muestra el diagrama de bloques del oscilador caótico en su segundo nivel del diseño, en él se observan tres unidades que componen el modelo: Unidad Oscilador Caótico de SNLF, Contador, Multiplexor y Registros.

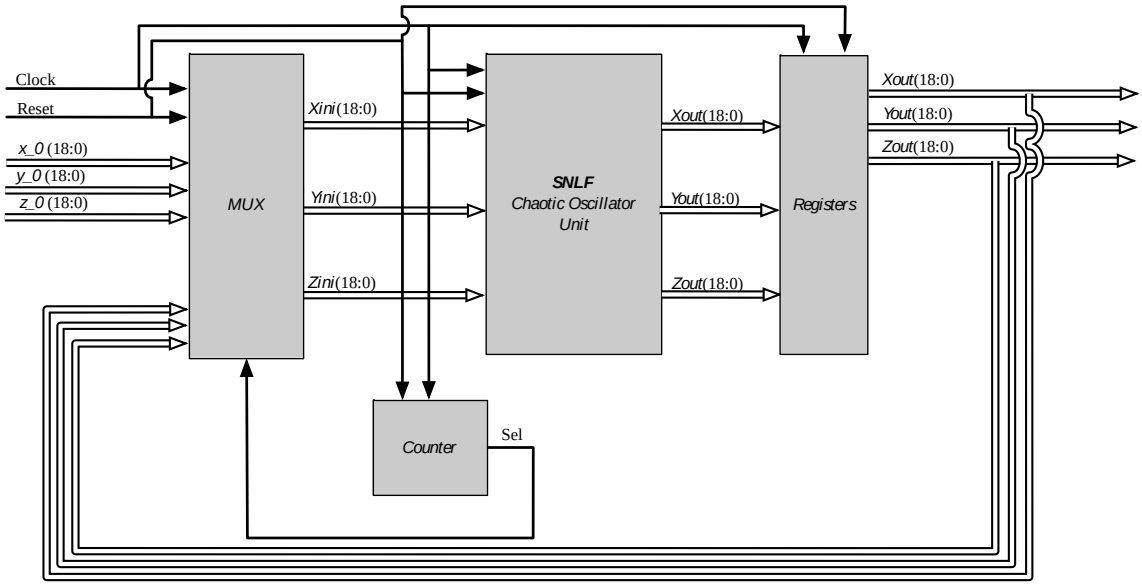

Figura 4.13. Diagrama de bloques en su segundo nivel de diseño del oscilador caótico basado en SNLF.

La unidad de Registros se encarga de capturar las señales *Xout*, *Yout*, y dar el tiempo necesario para que la señal *Zout* esté lista, debido a que esta señal necesita de más operaciones lo cual conlleva a mas ciclos de reloj.

En la figura 4.14 se muestra la caja negra del sistema completo, el cual indica el Top-level del diseño, o sea el primer nivel.

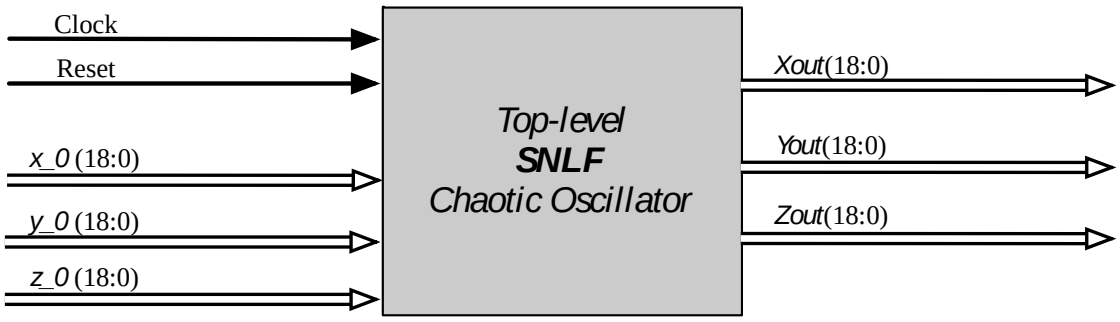

Figura 4.14. Top-level de oscilador caótico basado en SNLF.

4.6.2 Compilación de arquitectura y co-simulación Active-MATLAB para Euler.

Para la compilación del sistema descrito anteriormente, se utiliza el software Active-HDL el cual, además proporciona una interfaz para el entorno de simulación de MATLAB y Simulink, que permite la Co-simulación de bloques funcionales que pueden ser descritos mediante el uso de fórmulas matemáticas, modelos de comportamiento y lenguajes de descripción de hardware. La co-simulación facilita notablemente el análisis de los resultados antes de realizar la implementación físicamente, y ver así detalladamente cada señal, incluso la capacidad de almacenar los datos obtenidos como vectores en Matlab para usos posteriores.

A través de la compilación en el software Active, es posible generar un archivo *.m* de cualquier módulo o componente descrito mediante código VHDL, en este caso se genera el archivo a partir del Top-level del sistema mostrado en la figura 4.14 con el nombre de *osc*. En el entorno de Simulink mediante los bloques específicos de Active-HDL se puede realizar la integración Active con Simulink, esto mediante el bloque HDL Black-Box el cual será descrito por el archivo *.m* ya generado y el bloque Active-HDL Co-sim el cual permite la co-simulación. La integración de estos bloques en Simulink es presentada en la figura 4.15.

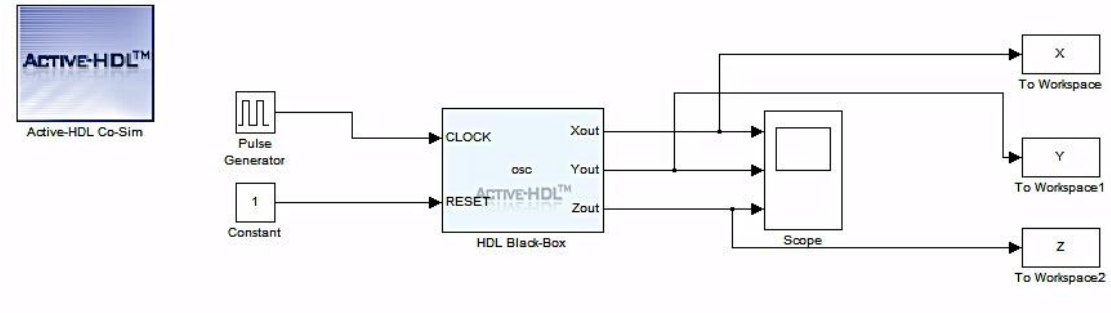

Figura 4.15. Oscilador caótico SNLF en co-simulación con MATLAB.

Para la señal de reloj "*CLOCK"* se utiliza un generador de pulsos con una frecuencia de 10 kHz, y para la señal de reinicio "*RESET"* simplemente una constante con el valor lógico de "1", también se utiliza un Scope (encargado de graficar las señales de entrada con respecto al tiempo de simulación), de esta manera visualizar las señales resultantes, y los bloques de Simulink "To workspace" los cuales capturan cada uno de los datos obtenidos y los almacena en arreglos de vectores (*arrays*), para visualizar los datos de forma numérica o simplemente graficarlos.

Se muestran los resultados de co-simulación del modelo diseñado en Active mediante código VHDL, en donde para simplicidad solo se muestra la co-simulación del caso de la SNLF para seis enrollamientos, utilizando un tiempo de simulación de $3x10^6$ segundos. En la figura 4.16 se visualizan claramente las señales *Xout*, *Yout*, *Zout* y el atractor *X-Y* que es donde mejor se aprecian los multi-enrollamientos.

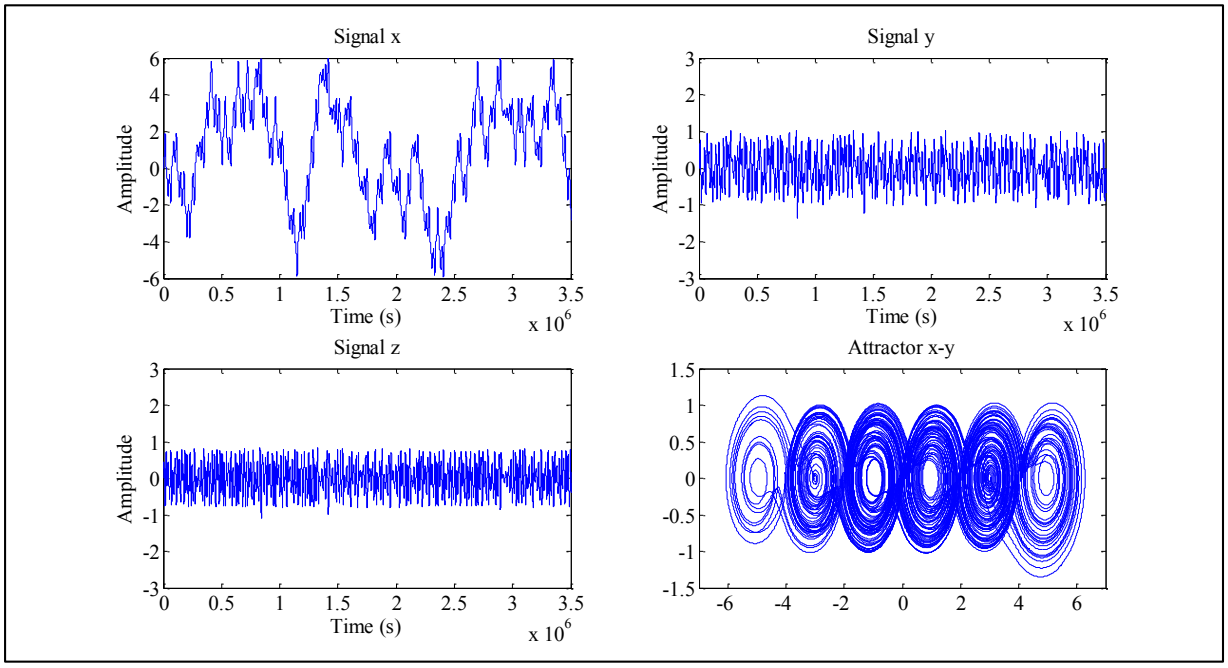

Figura 4.16. Oscilador caótico resuelto por Euler en co-simulación con una SNLF para seis enrollamientos.

Se observa que las señales obtenidas son satisfactorias ya que se visualizan claramente los enrollamientos, mostrando también la importancia que tiene la co-simulación en cuanto al análisis de los resultados, no solo para este tipo de sistemas en específico sino para cualquier tipo de diseño en VHDL, Verilog o RTL, ya que permite una visualización más exacta en base al comportamiento ideal del circuito antes de la implementación en hardware.

4.6.3 Compilación de arquitectura y co-simulación Active-MATLAB para Runge-Kutta.

De manera similar se realiza el diseño de la arquitectura del oscilador caótico resuelto mediante el algoritmo de Runge-Kutta, la arquitectura se realiza igualmente re-utilizando cada bloque (sumador, restador, multiplicador, multiplexor, contador, registros) ya realizado anteriormente mediante

código VHDL, debido a que la arquitectura obtenida es muy grande solo se muestran los resultados obtenidos. Sin embargo la metodología seguida es la misma que con el método de Euler de la sección 4.6. Utilizando el software Active-HDL para la compilación, se puede utilizar nuevamente el modelo implementado en Simulink de la figura 4.18 para la co-simulación. Obteniendo así las señales de la figura 4.17 donde se muestra la co-simulación del sistema diseñado con una SNLF para seis enrollamientos.

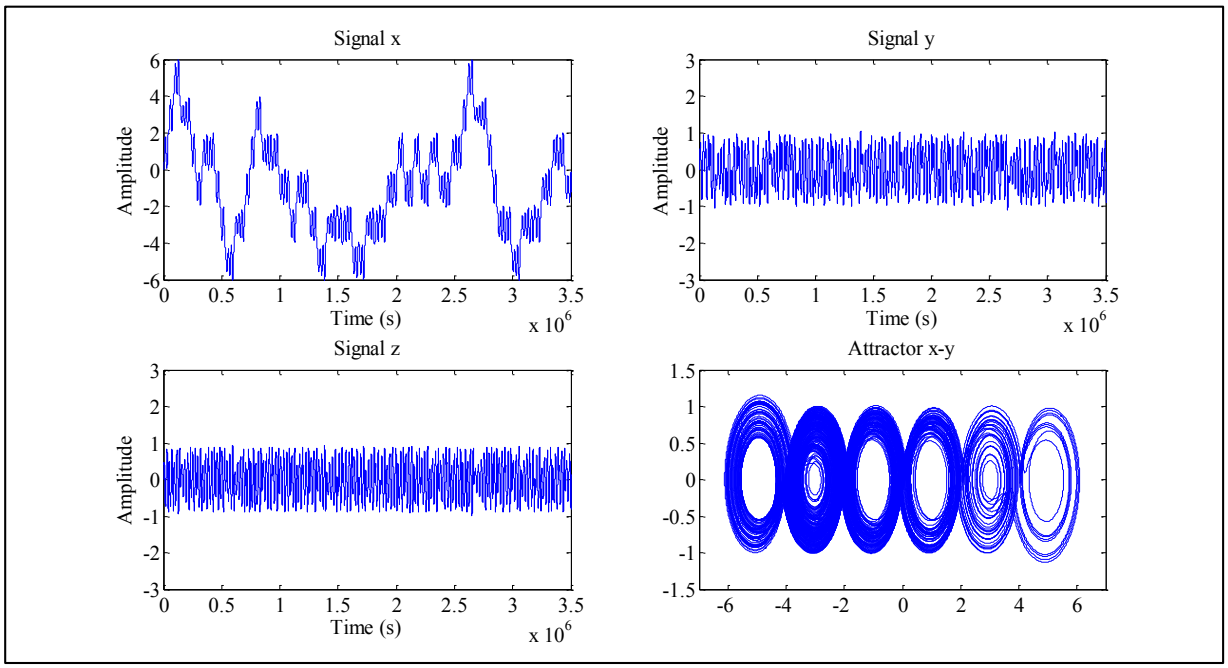

Figura 4.17. Oscilador caótico resuelto por Runge-Kutta en co-simulación con una SNLF para seis enrollamientos.

4.6.4 Resultados experimentales.

Se utilizó nuevamente el dispositivo FPGA Cyclone III Edition de Altera mostrado en la figura 4.1. A partir del modelo matemático discretizado mediante los algoritmos de Euler y Runge Kutta, se realizó la compilación y programación de las arquitecturas diseñadas mediante código VHDL, con ayuda del programa informático Quartus II de Altera.

Los valores de las condiciones iniciales que se utilizan en este trabajo son: *x(0)=0.1*, *y(0)=0.1* y *z(0)=0*, señalados en la sección XYZ, las cuales permiten obtener la primera iteración. Para reducir el consumo de recursos de la tarjeta FPGA todos estos valores son embebidos dentro del diseño.
A continuación se presentan las imágenes de los multi-enrollamientos en tiempo real, adquiridas a través de un osciloscopio donde en todas las figuras se muestran las señales *x*, *y*, *z* y el atractor *x-y*. En la figura 4.18 se muestran las señales obtenidas utilizando el método de Euler con un SNLF para dos enrollamientos, en la figura 4.19 para una SNLF de tres enrollamientos, en la figura 4.20 para una SNLF de cuatro enrollamientos, en la figura 4.21 para un SNLF de cinco enrollamientos y por último en la figura 4.22 para un SNLF de seis enrollamientos.

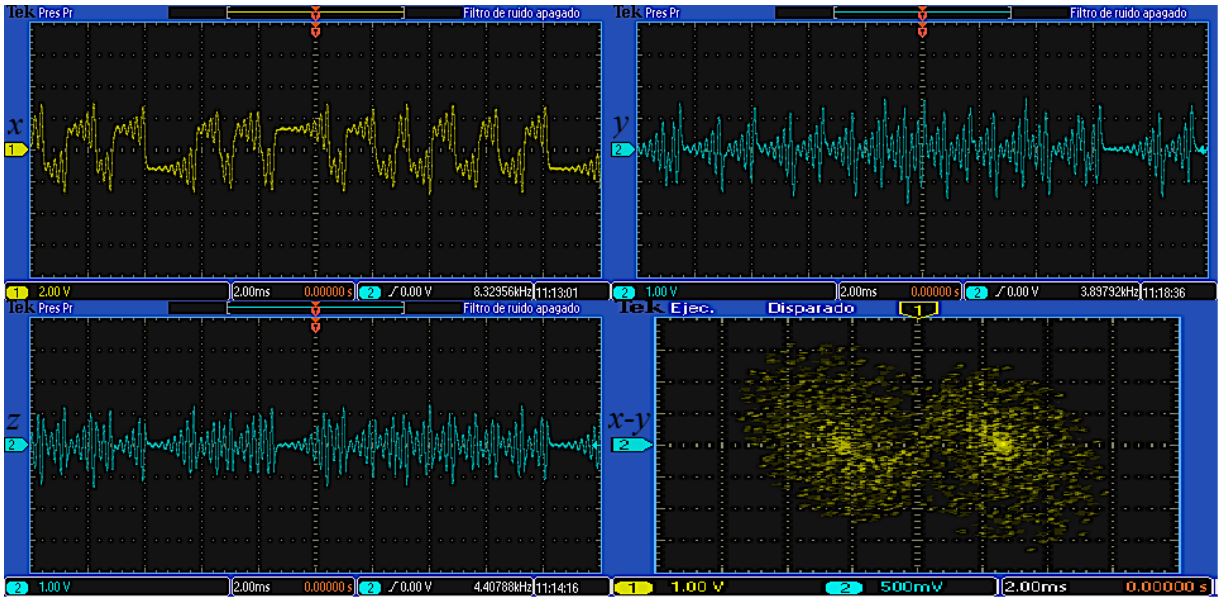

Figura 4.18. Señales utilizando Euler en tiempo real con una SNLF para dos enrollamientos.

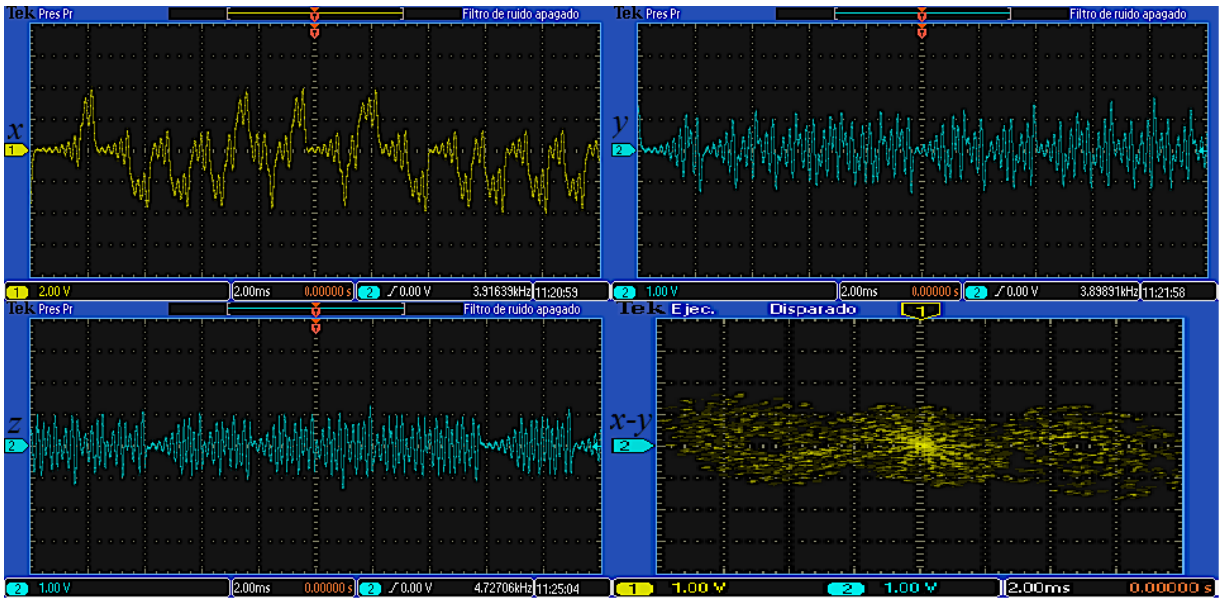

Figura 4.19. Señales utilizando Euler en tiempo real con una SNLF para tres enrollamientos.

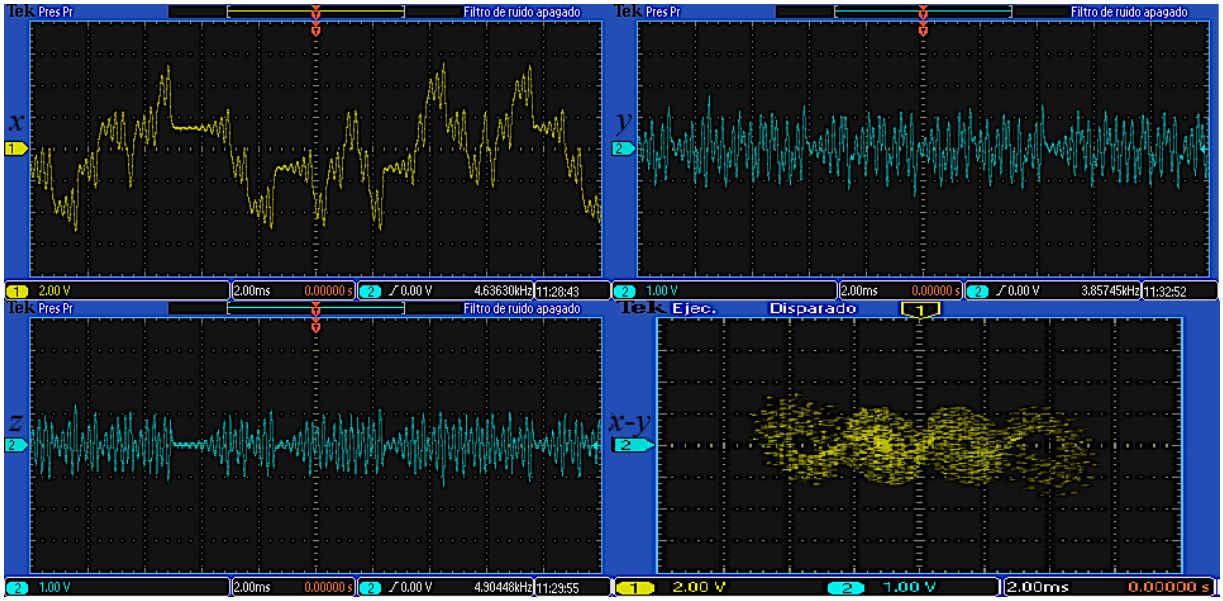

Figura 4.20. Señales utilizando Euler en tiempo real con una SNLF para cuatro enrollamientos.

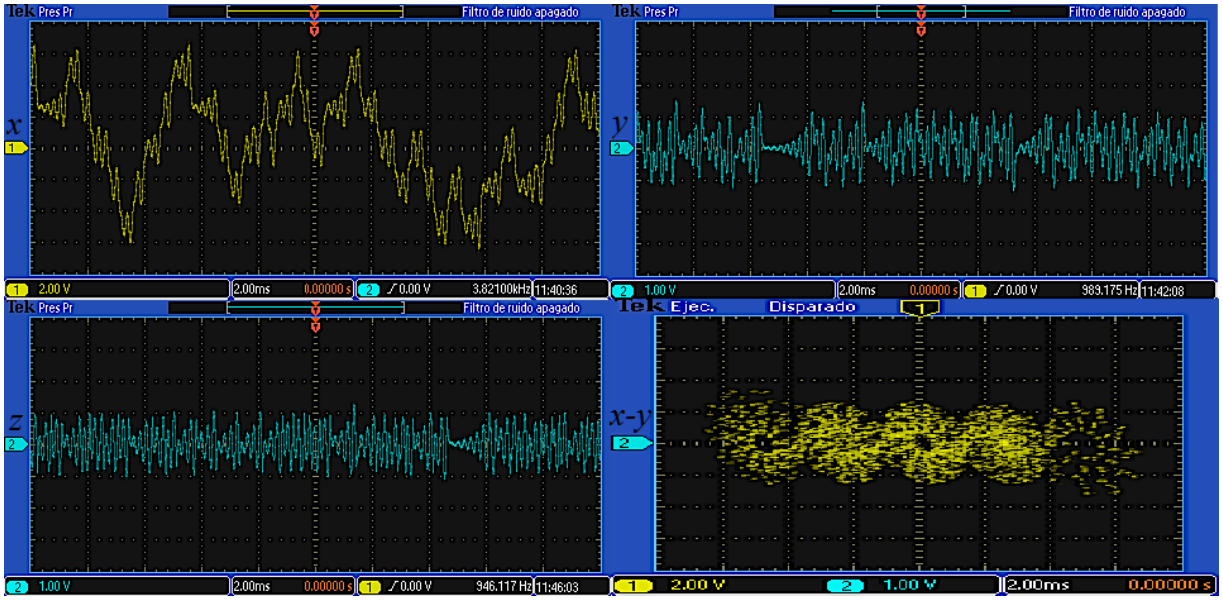

Figura 4.21. Señales utilizando Euler en tiempo real con una SNLF para cinco enrollamientos.

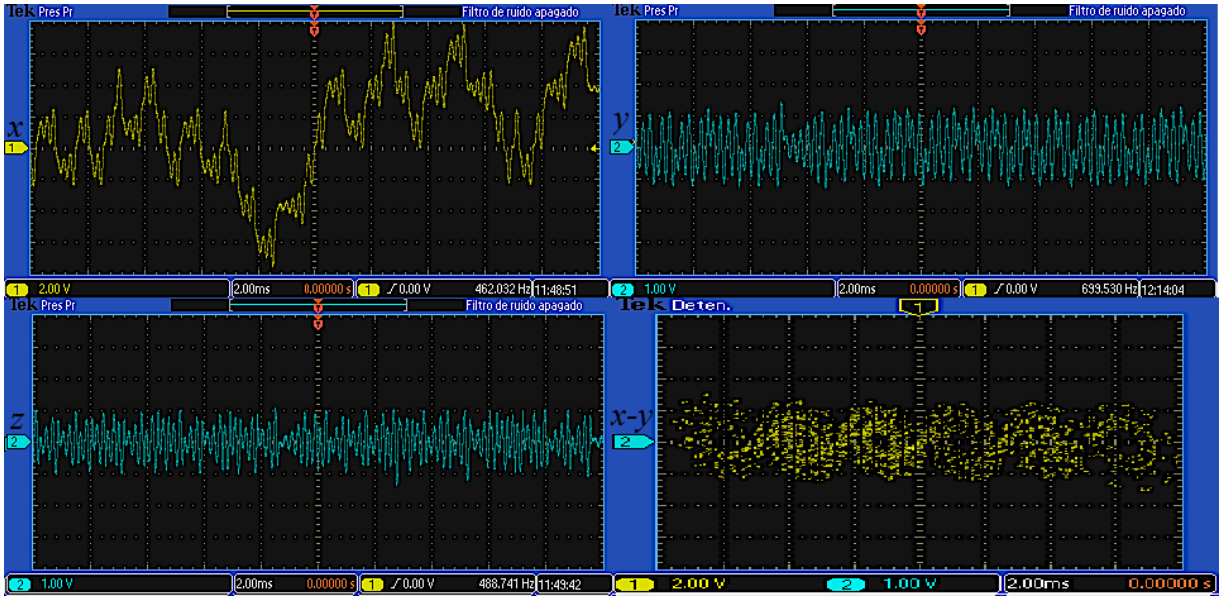

Figura 4.22. Señales utilizando Euler en tiempo real con una SNLF para seis enrollamientos.

Similarmente se capturaron las señales de la implementación mediante el algoritmo de Runge-Kutta de orden 4 (RK4), en la figura 4.23 se muestra el atractor *x-y* utilizando Runge-Kutta con un SNLF para dos enrollamientos, en la figura 4.24 para una SNLF de tres enrollamientos, en la figura 4.25 para una SNLF de cuatro enrollamientos, en la figura 4.26 para un SNLF de cinco enrollamientos y por último en la figura 4.27 para un SNLF de seis enrollamientos

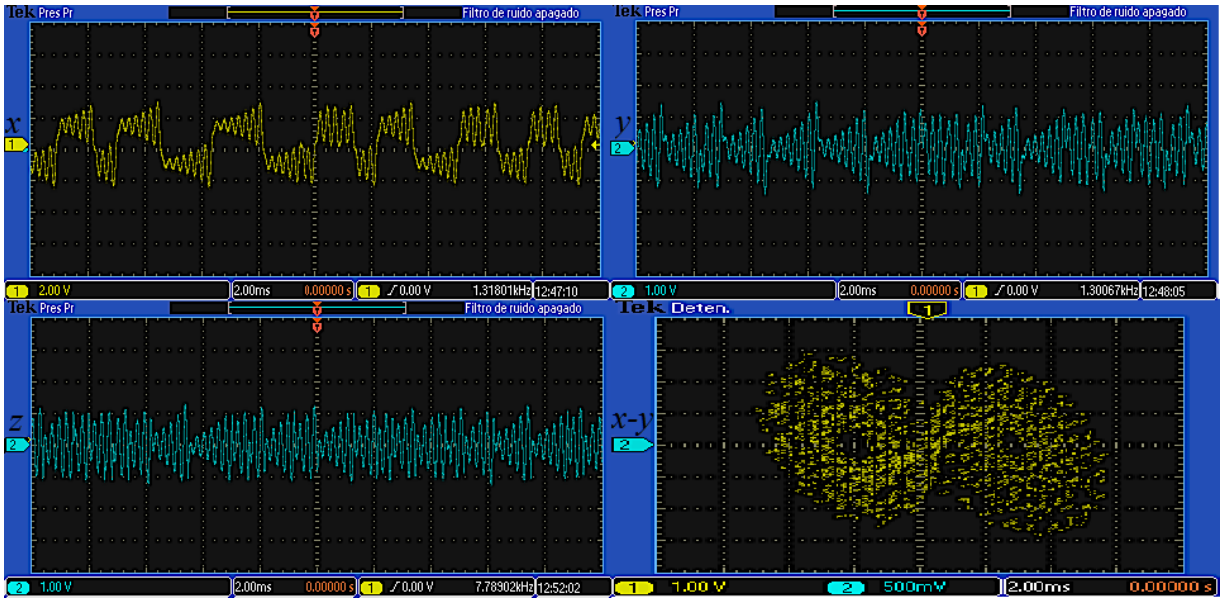

Figura 4.23. Señales utilizando RK4 en tiempo real con una SNLF para dos enrollamientos.

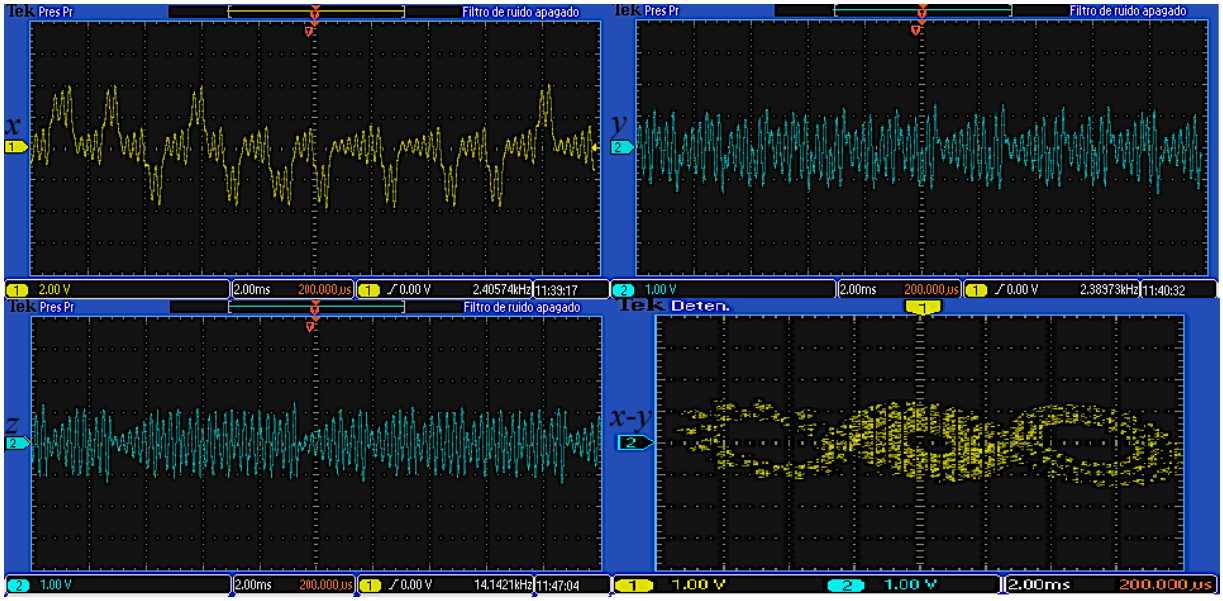

Figura 4.24. Señales utilizando RK4 en tiempo real con una SNLF para tres enrollamientos.

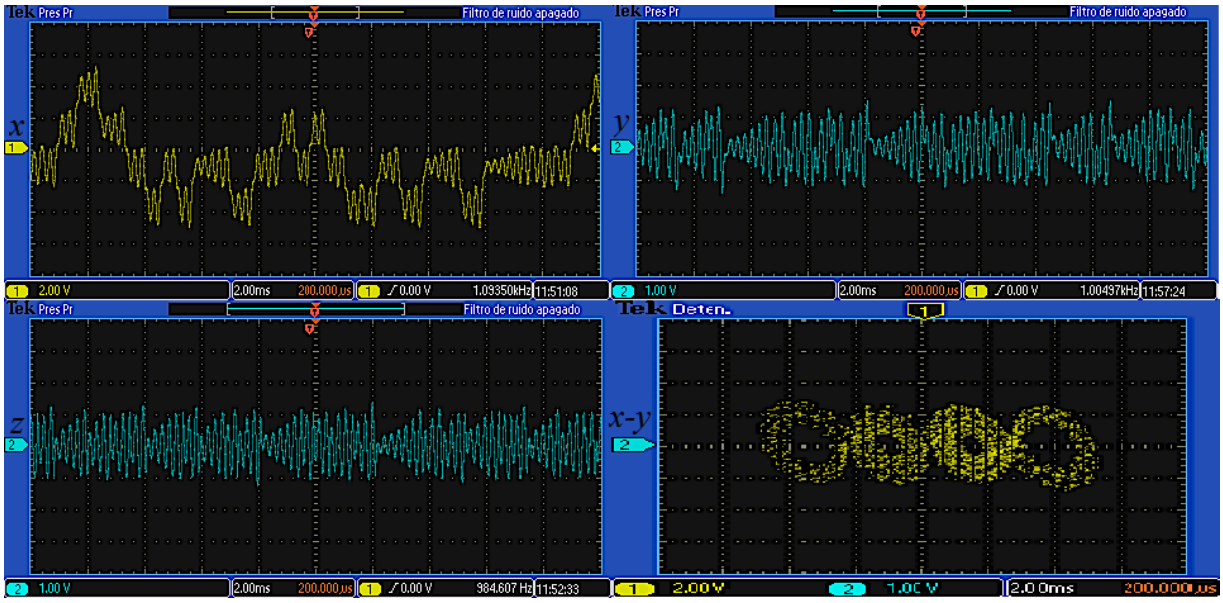

Figura 4.25. Señales utilizando RK4 en tiempo real con una SNLF para cuatro enrollamientos.

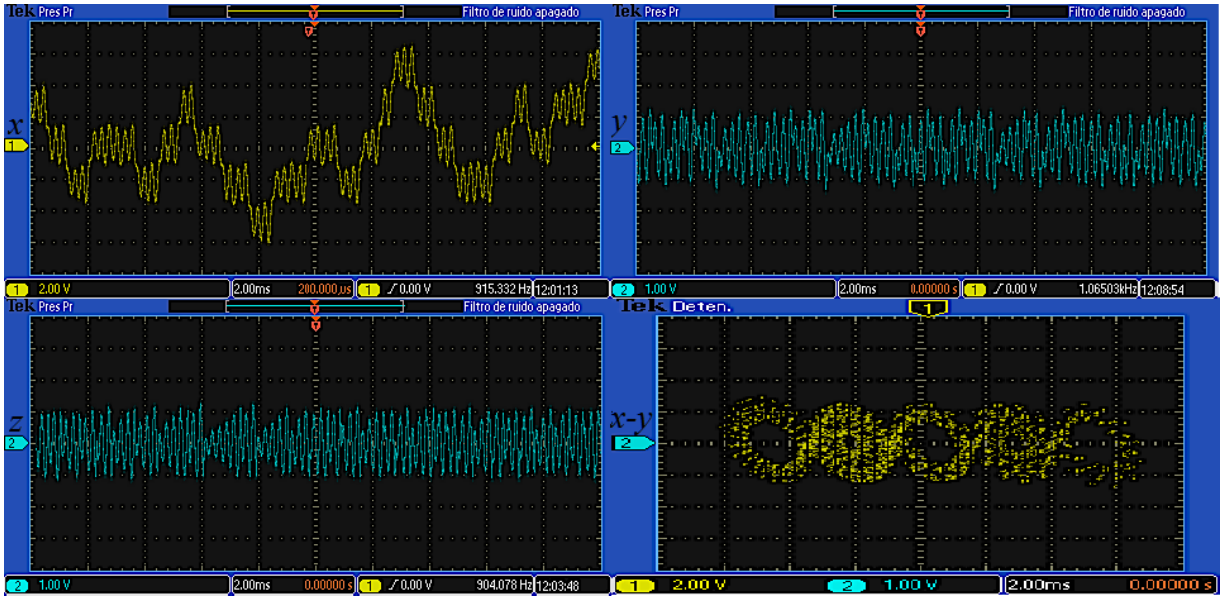

Figura 4.26. Señales utilizando RK4 en tiempo real con una SNLF para cinco enrollamientos.

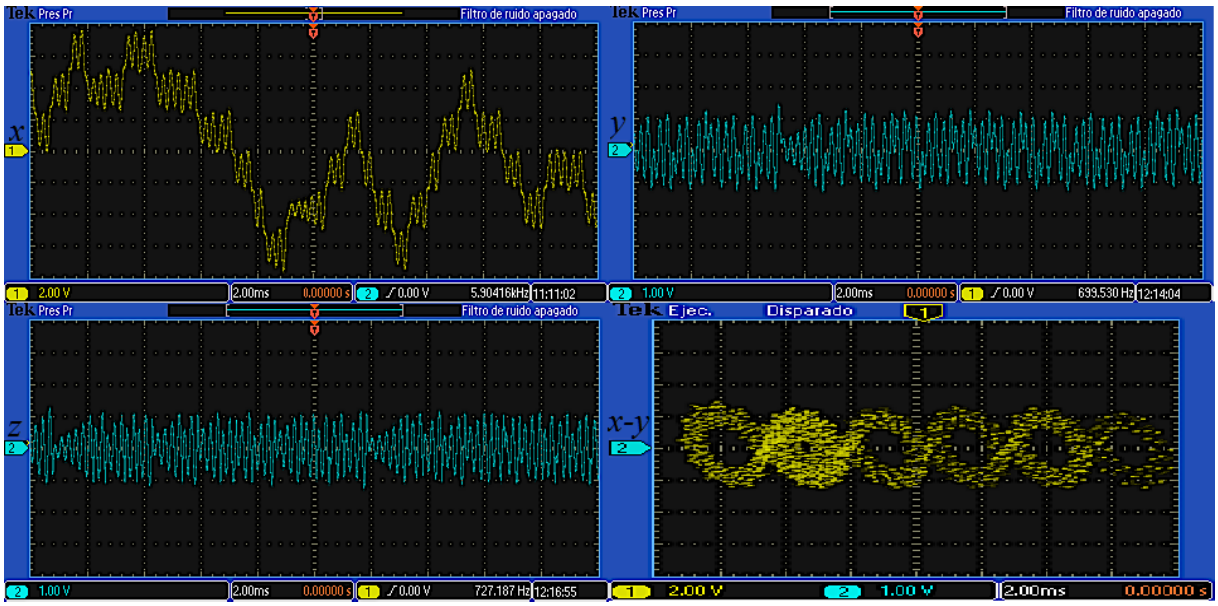

Figura 4.27. Señales utilizando RK4 en tiempo real con una SNLF para seis enrollamientos.

Como se puede observar en las figuras 4.18-4.27, los atractores obtenidos son muy similares a los resultados ideales obtenidos en co-simulación de la sección 4.3.2, solo que obviamente tienen menos resolución debido al ruido y al muestreo de los convertidores analógico-digital del FPGA, pero si se comparan con los atractores implementados en Matlab mediante código en la sección 3.2 son completamente similares.

4.7 IMPLEMENTACIÓN DE OSCILADOR CAÓTICO BASADO EN EL CIRCUITO DE CHUA MEDIANTE CÓDIGO VHDL.

En esta sección se presenta la metodología que se siguió para la implementación en un FPGA del oscilador caótico basado en el circuito de Chua, diseñando nuevamente una arquitectura a partir de los métodos numéricos de Euler y Runge-Kutta, esta arquitectura es diseñada mediante código VHDL para su co-simulación y su posterior implementación en FPGA.

4.7.1 Arquitectura en base al algoritmo de Euler.

Se aplica el método de Euler para discretizar el sistema de ecuaciones (33-35) que describen el oscilador caótico basado en el circuito de Chua, tal y como se hizo en la sección 4.6.1 con el oscilador caótico basado en SNLF.

Aplicando entonces la formula iterativa de Euler (49) al sistema de ecuaciones (33-35), se obtiene el siguiente conjunto de ecuaciones discretas (72).

$$\begin{aligned} x(i+1) &= x(i) + step * \alpha(y(i) - x(i) - g(x(i))) \\ y(i+1) &= y(i) + step * (x(i) - y(i) - z(i)) \\ z(i+1) &= z(i) + step * (-\beta y(i)) \end{aligned} \tag{72}$$

Como en la primera ecuación de (72) la función que describe al diodo de Chua $g(x(i))$ se encuentra implícita, esta ecuación depende de dicha función, por lo que dependiendo del valor que tenga $x(i)$ corresponde el valor que tendrá $g(x(i))$, para abordar con mayor simplicidad la explicación, se tomará la función del diodo de Chua para dos enrollamientos mostrada en la figura 2.18, y su descripción PWL (37). Esta función se describe por tres condiciones, por lo cual la ecuación $x(i+1)$ del sistema de ecuaciones (72) tendrá tres variantes, como se muestra en la tabla 4.3.

Tabla 4.3. Condiciones para ecuación $\boldsymbol{x(i+1)}$.

Condición	$\boldsymbol{x(i+1)}$
$\boldsymbol{x(i) > P}$	$x(i) + step * \alpha(y(i) - x(i) - (m_0 x(i) + P(m_0 - m_1)))$
$\boldsymbol{-P \leq x \leq P}$	$x(i) + step * \alpha(y(i) - x(i) - (m_0 x(i)))$
$\boldsymbol{x(i) < -P}$	$x(i) + step * \alpha(y(i) - x(i) - (m_1 x(i) + P(m_1 - m_0)))$

Para la implementación del sistema de ecuaciones (72) se re-utilizan nuevamente los componentes sumador, restador y multiplicador, ya diseñados en la sección 4.6.1 mostrados en la figura 4.13, se utiliza también palabras con punto fijo de 19 bits con la misma estructura de la tabla 4.2.

En este diseño se utilizan los mismos valores de la tabla 3.1 usados en las simulaciones de MATLAB en la sección 3.3.

La implementación de la primera ecuación $x(i+1)$ del sistema de ecuaciones (72) se muestra en el diagrama de bloques de la figura 4.28, de la misma manera la implementación de la segunda ecuación $y(i+1)$ se muestra en la figura 4.29, y la implementación de la tercera ecuación $z(i+1)$ se muestra en la figura 4.30. En las figuras se puede observar que *Xini*, *Yini* y *Zini* representan los estados presentes del sistema y *Xout*, *Yout* y *Zout* los estados futuros o salidas del sistema. Se

muestra también que se hace uso de un comparador, que al igual que en la implementación del oscilador caótico basado en SNLF es uno de los componentes más importantes del sistema ya que en él se define el comportamiento de la función del diodo de Chua, este se realiza a partir de condiciones, para efectos prácticos solo se muestra en diagrama de bloques la implementación para dos enrollamientos debido a su simplicidad, pero de la misma manera que con el oscilador caótico basado en SNLF al añadir más condiciones al comparador dependiendo de la función del diodo de Chua que se requiere describir, se obtendrán el número de enrollamientos deseados.

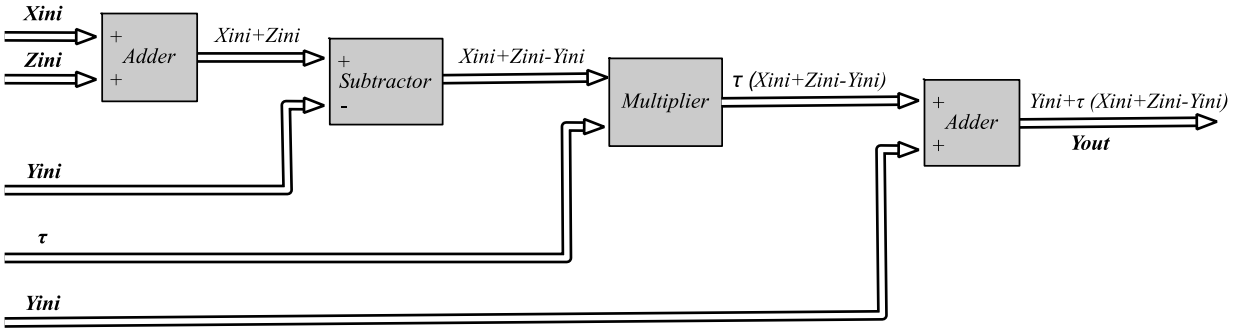

Figura 4.28. Diagrama de bloques de ecuación $x(i+1)$ del sistema de ecuaciones (49).

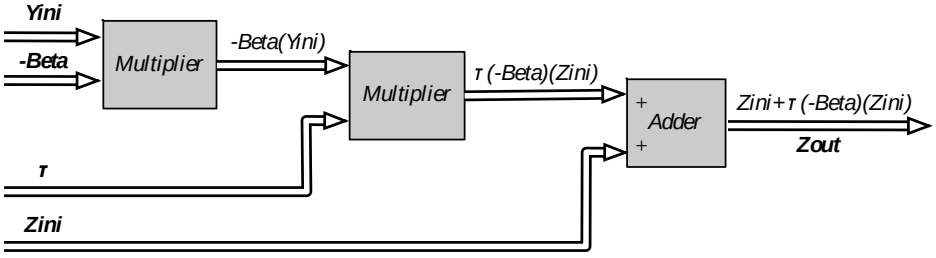

Figura 4.29. Diagrama de bloques de ecuación $y(i+1)$ del sistema de ecuaciones (49).

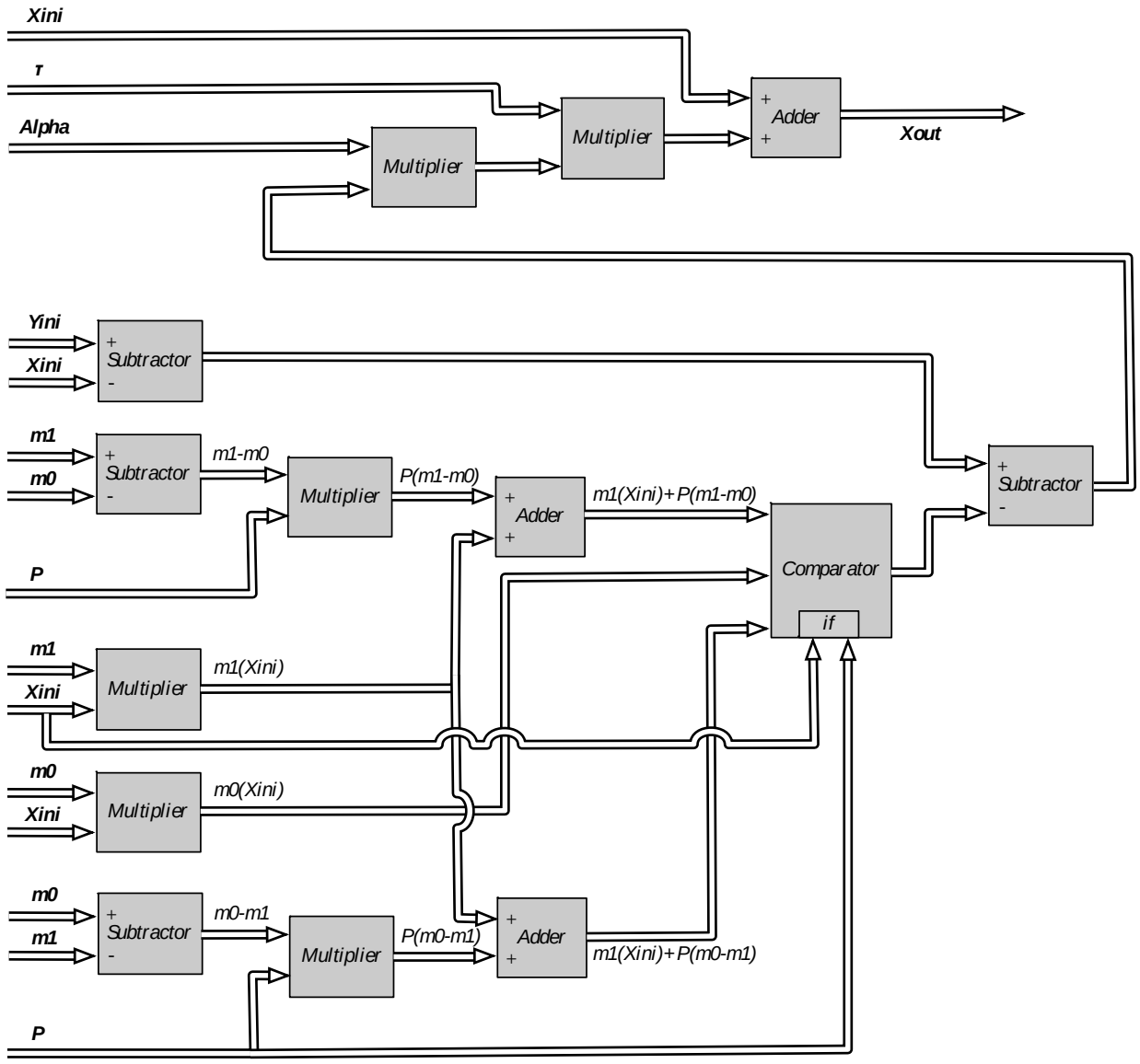

Figura 4.30. Diagrama de bloques de ecuación $z(i+1)$ del sistema de ecuaciones (49).

Los diagramas de bloques de las figuras 4.28, 4.29 y 4.30 muestran así el sistema implementado a partir de las ecuaciones (72), sin embargo este todavía está en el tercer nivel del diseño, a la cual se le dio el nombre de Unidad de Oscilador Caótico de Chua la cual es representada en la figura 4.31, donde *Xini*, *Yini* y *Zini* son las señales de las condiciones iniciales de 19 bits, también se observan las señales de entrada *Clock*, *Reset* de 1 bit, las cuales se han utilizado para garantizar la sincronización de los componentes dentro de la unidad.

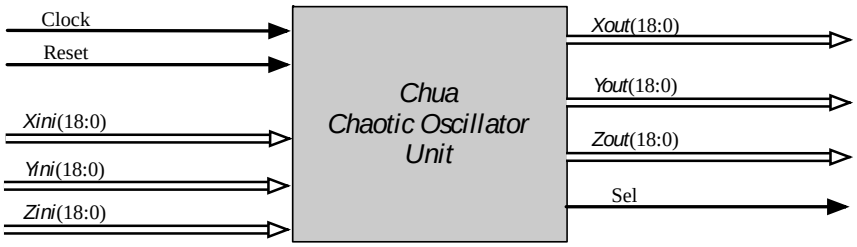

Figura 4.31. Unidad de Oscilador Caótico de Chua.

Tal y como se explica en la sección 4.6.1, si se implementa el sistema así como tal, solo se obtendrá la primera iteración, para corregir esto se realiza una retro alimentación igual que con el oscilador caótico basado en SNLF tal y como se muestra en la figura 4.32.

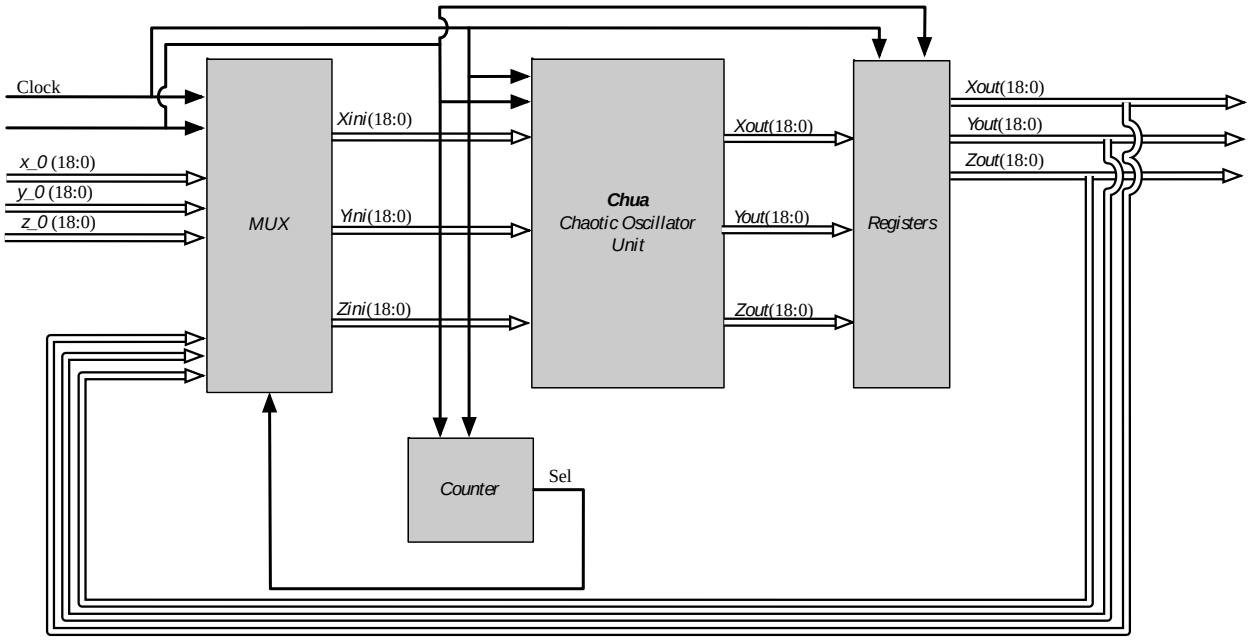

Figura 4.32. Diagrama de bloques del Oscilador Caótico de Chua en su segundo nivel de diseño.

En la figura 4.33 se muestra la caja negra del sistema completo, el cual indica el nivel jerarquico superior *Top-level* del diseño.

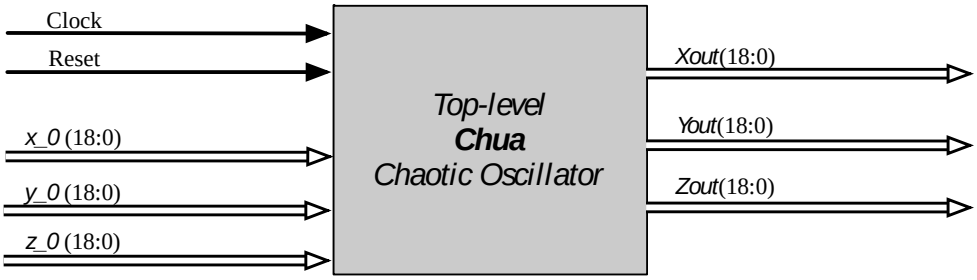

Figura 4.33. Top-level del oscilador caótico basado en el circuito de Chua.

4.7.2 Compilación de arquitectura y co-simulación Active-MATLAB para Euler.

En esta subsección se presenta la compilación del sistema a partir del método de Euler con código VHDL, se muestra también los resultados de la co-simulación.
Para la compilación del sistema descrito anteriormente, se utiliza nuevamente el software Active-HDL, realizando bloque por bloque mediante código VHDL. Para la co-simulación Se utilizan los mismos bloques en Simulink presentados anteriormente en la figura 4.15.
Nuevamente se utiliza un *CLOCK* de 10 kHz y el valor lógico de "1" para el *RESET*, los resultados de la co-simulación se almacenan como vectores en el "workspace (espacio de trabajo)" de MATLAB. En la figura 4.34 se muestra la co-simulación utilizando la función del diodo de Chua de la figura 3.8 para seis enrollamientos, se visualizan las señales *Xout* (Signal x), *Yout* (Signal y), *Zout* (Signal z) y el atractor *X-Y*.

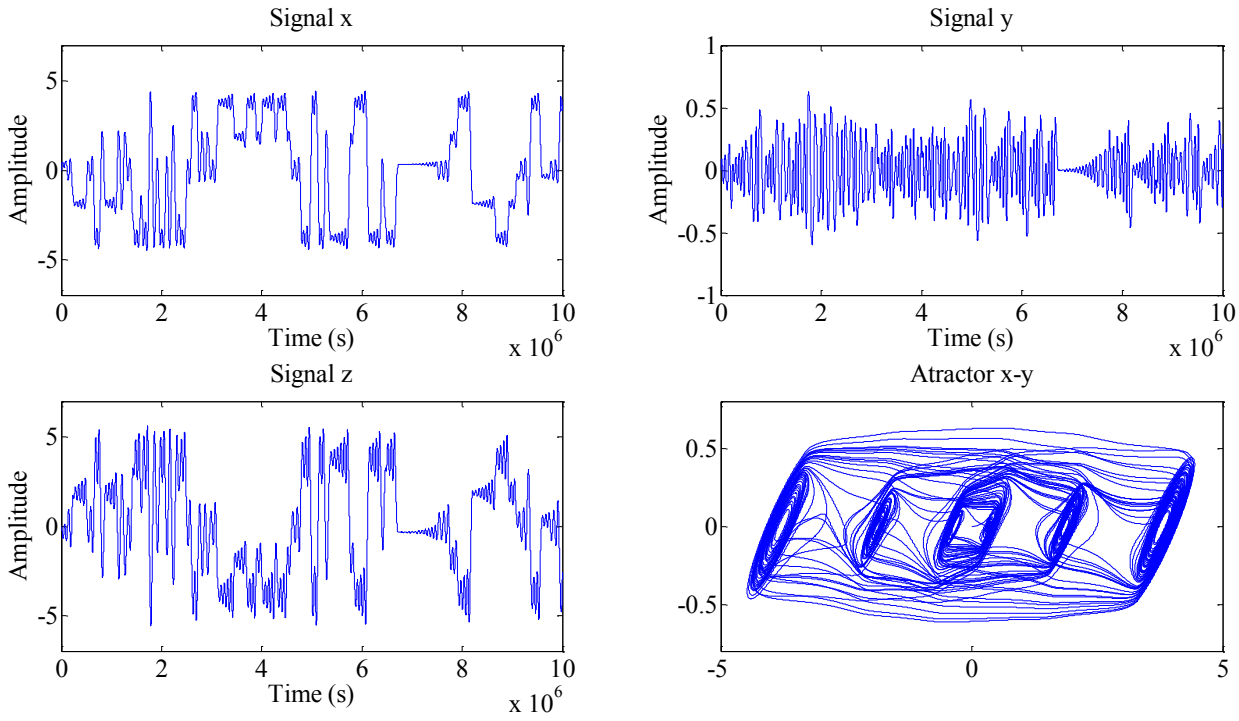

Figura 4.34. Oscilador caótico resuelto por Euler en co-simulación con una función de Chua para seis enrollamientos.

4.7.3 Compilación de arquitectura y co-simulación Active-MATLAB para Runge-Kutta.

De manera similar se realiza la arquitectura del oscilador caótico resuelto mediante el algoritmo de Runge-Kutta, la arquitectura se realiza igualmente re-utilizando cada bloque (sumador, restador, multiplicador, multiplexor, contador, registros) ya diseñados anteriormente mediante código VHDL. Debido a que la arquitectura obtenida es muy grande ¿Por qué?, especificar solo se muestran los resultados obtenidos, sin embargo la metodología seguida es la misma que con el método de Euler. Utilizando el software Active-HDL para la compilación, se puede utilizar nuevamente el modelo implementado en Simulink de la figura 4.15 para la co-simulación. Obteniendo así las siguientes señales, donde en la figura 4.35 se muestra la co-simulación del sistema utilizando una función del diodo de Chua para seis enrollamientos.

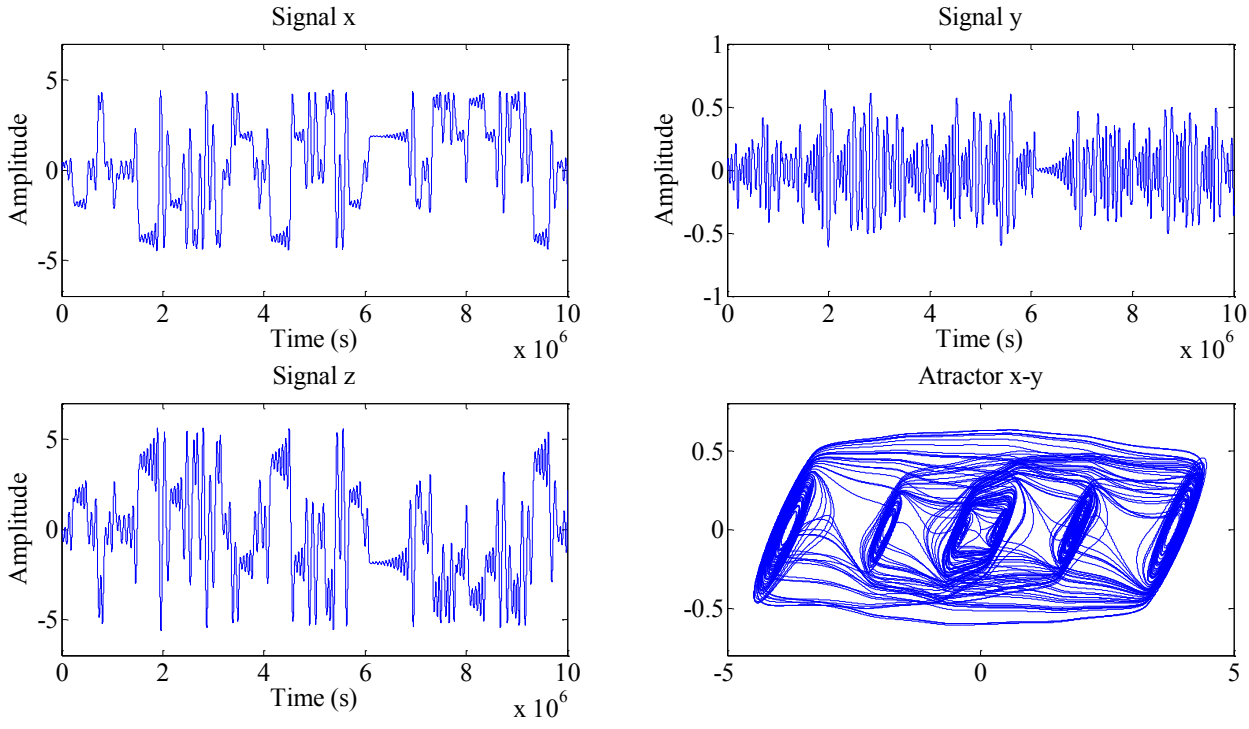

Figura 4.35. Oscilador caótico resuelto por Runge-Kutta en co-simulación con una función de Chua para seis enrollamientos.

Como se puede observar no hay tanta diferencia entre el método numérico utilizado para la solución Euler o Runge-Kutta para este tipo de sistemas. Cabe recalcar en este caso que para este sistema fue necesario utilizar un paso (*step)* de 0.001 más pequeño que el utilizado en el oscilador caótico basado en SNLF que fue de 0.01, debido a que este sistema necesita de mayor precisión en la solución, porque en este caso los enrollamientos generan pendientes y en el caso del oscilador SNLF son los niveles de saturación los cuales son solo valores constantes.

4.7.4 Resultados experimentales.

Se utiliza nuevamente el kit de desarrollo DE2i-150 de Altera, que incluye el dispositivo FPGA Cyclone IV Edition. A partir del modelo matemático discretizado mediante los algoritmos de Euler y Runge-Kutta, y la descripción del modelo en arquitectura mediante bloques, para realizar así cada módulo con código VHDL, tal y como se hizo con el oscilador caótico basado en SNLF.

Los valores de las condiciones iniciales que se utilizan son las mismas que en simulación y co-simulación: *x(0)=0.1*, *y(0)=0* y *z(0)=0*, las cuales permiten obtener la primera iteración, también se utilizaron los mismos valores de las constantes de la tabla 3.1, para reducir el consumo de recursos de la tarjeta FPGA todos estos valores son embebidos dentro del diseño.

A continuación se presentan las imágenes de los multi-enrollamientos en tiempo real del sistema caótico de Chua resuelto por Euler y Runge-Kutta orden 4, las señales fueron adquiridas a través de

un osciloscopio donde en todas las figuras se muestran las señales de las variables de estado *x*, *y*, *z* y el atractor *x*-*y*. En la figura 4.36 se muestran las señales obtenidas utilizando el método de Euler con una función del diodo de Chua para dos enrollamientos, en la figura 4.40 con una función del diodo de Chua para tres enrollamientos, en la figura 4.38 con una función del diodo de Chua para cuatro enrollamientos, en la figura 4.39 con una función del diodo de Chua para cinco enrollamientos y por último en la figura 4.40 con una función del diodo de Chua para seis enrollamientos.

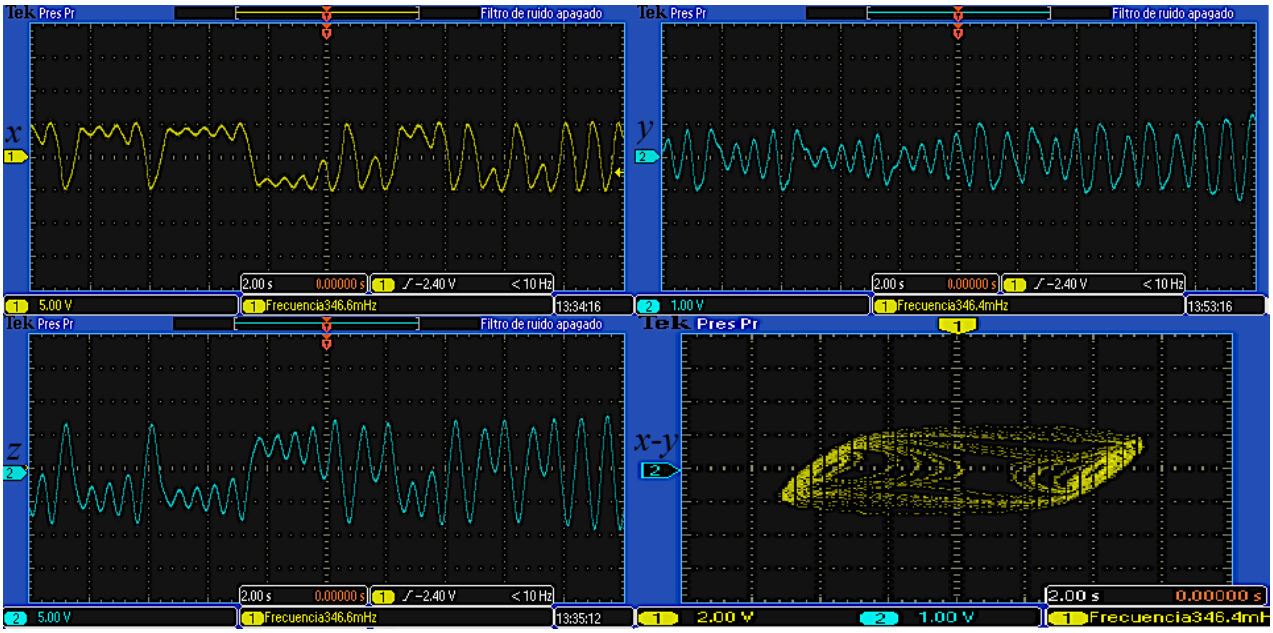

Figura 4.36. Señales utilizando Euler en tiempo real con una función de Chua para dos enrollamientos.

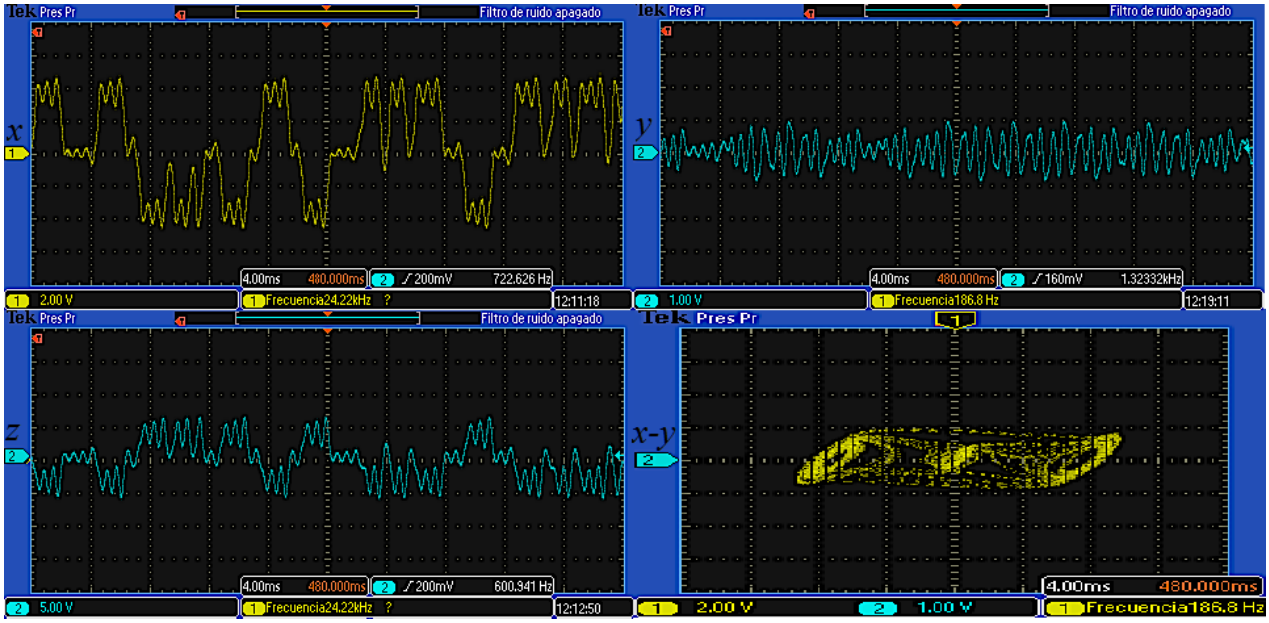

Figura 4.37. Señales utilizando Euler en tiempo real con una función de Chua para tres enrollamientos.

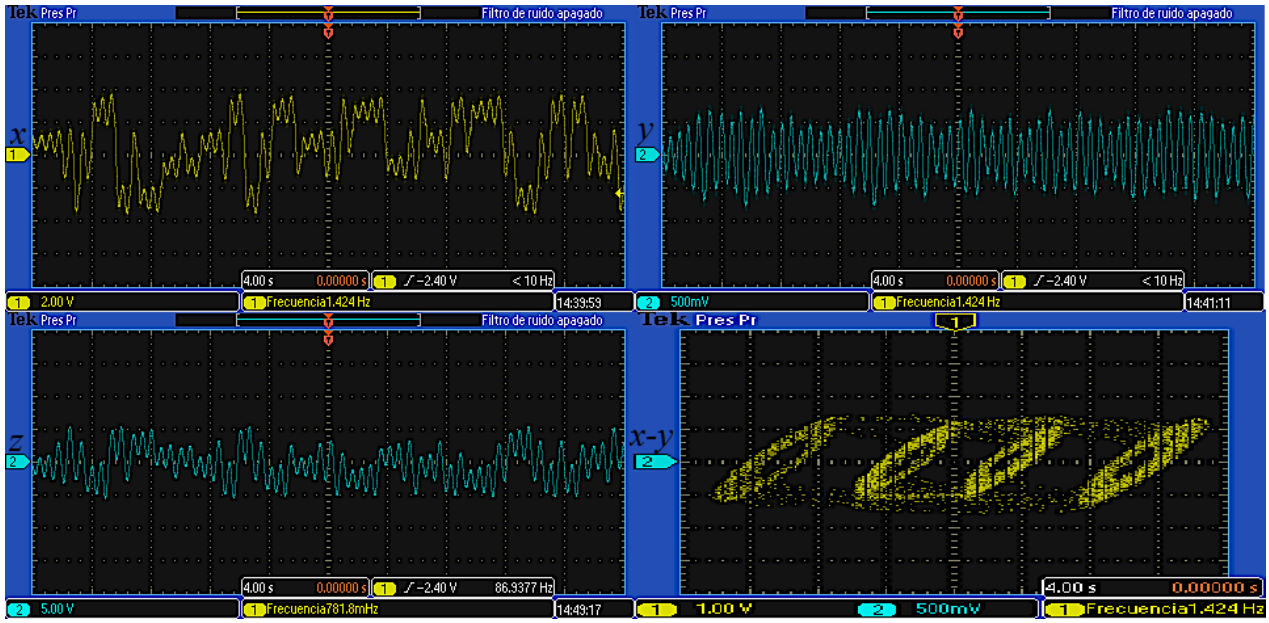

Figura 4.38. Señales utilizando Euler en tiempo real con una función de Chua para cuatro enrollamientos.

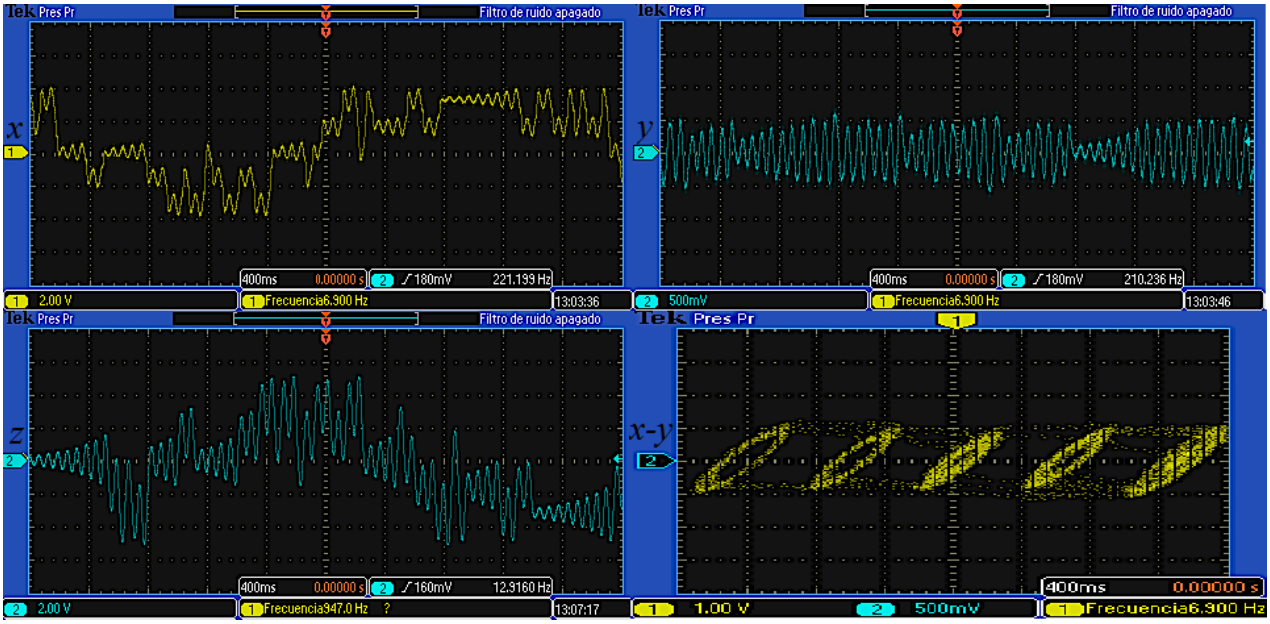

Figura 4.39. Señales utilizando Euler en tiempo real con una función de Chua para cinco enrollamientos.

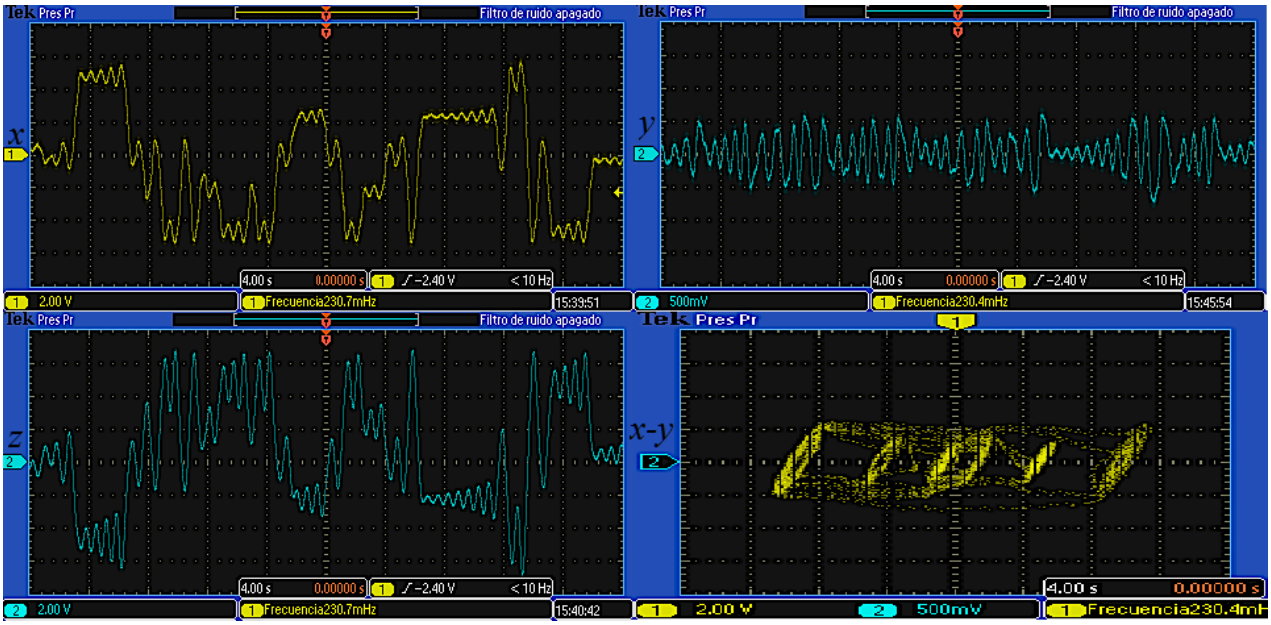

Figura 4.40. Señales utilizando Euler en tiempo real con una función de Chua para seis enrollamientos.

De la misma manera se capturaron las señales de la implementación mediante el algoritmo de Runge-Kutta de orden 4 (RK4), en la figura 4.41 se muestra el atractor *x-y* utilizando Runge-Kutta con una función del diodo de Chua para dos enrollamientos, en la figura 4.42 para una función del diodo de Chua de tres enrollamientos, en la figura 4.43 para una función del diodo de Chua de cuatro

enrollamientos, en la figura 4.44 para una función del diodo de Chua de cinco enrollamientos y por último en la figura 4.45 para una función del diodo de Chua de seis enrollamientos.

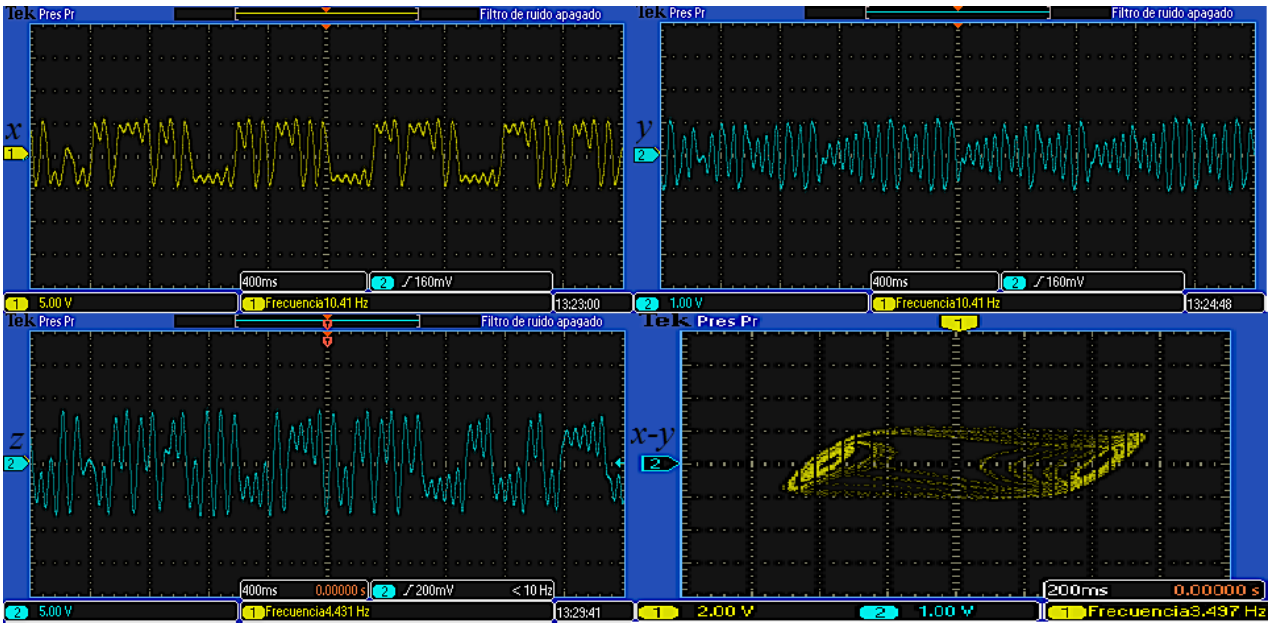

Figura 4.41. Señales utilizando RK4 en tiempo real con una función de Chua para dos enrollamientos.

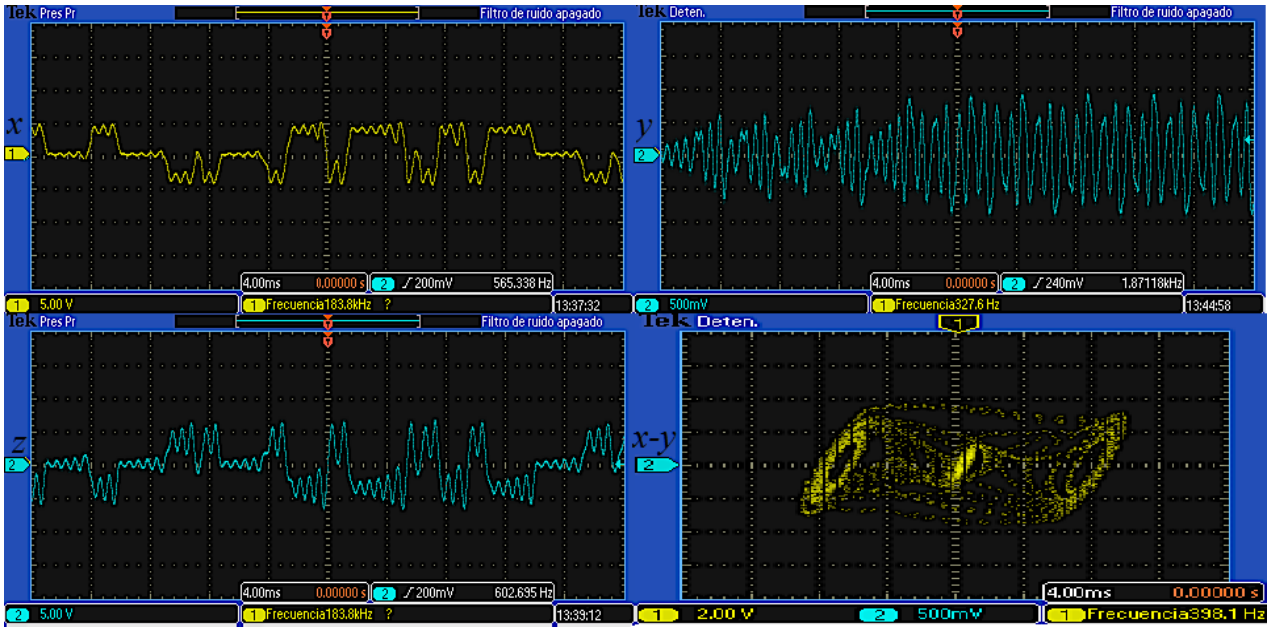

Figura 4.42. Señales utilizando RK4 en tiempo real con una función de Chua para tres enrollamientos.

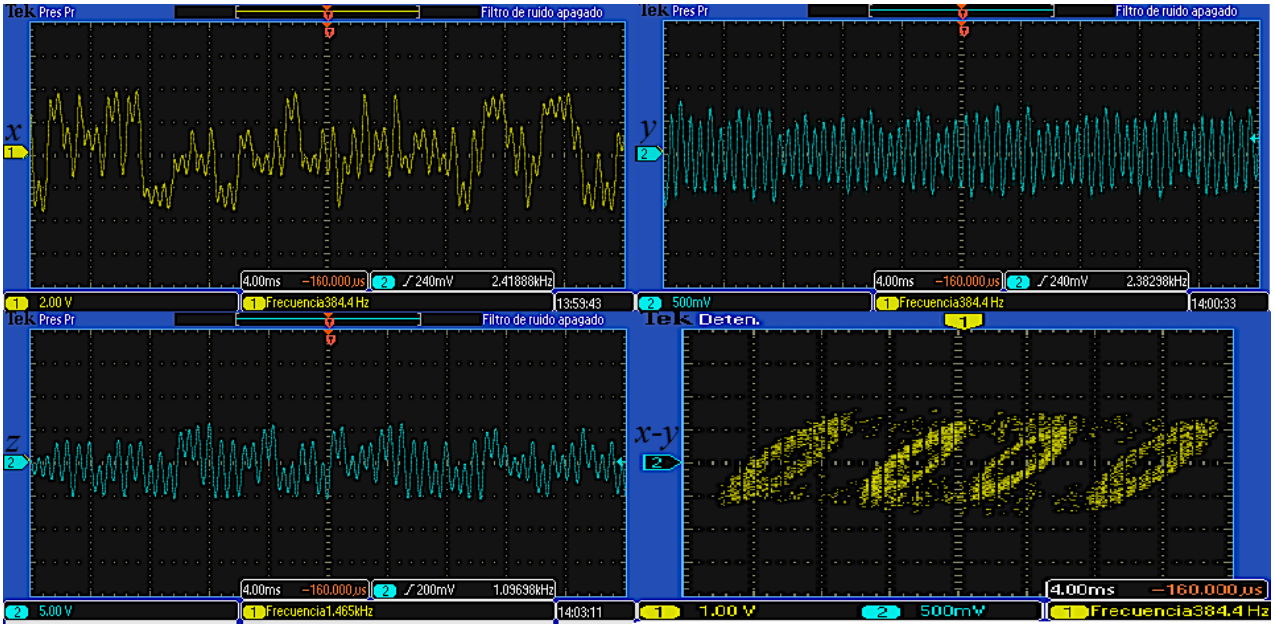

Figura 4.43. Señales utilizando RK4 en tiempo real con una función de Chua para cuatro enrollamientos.

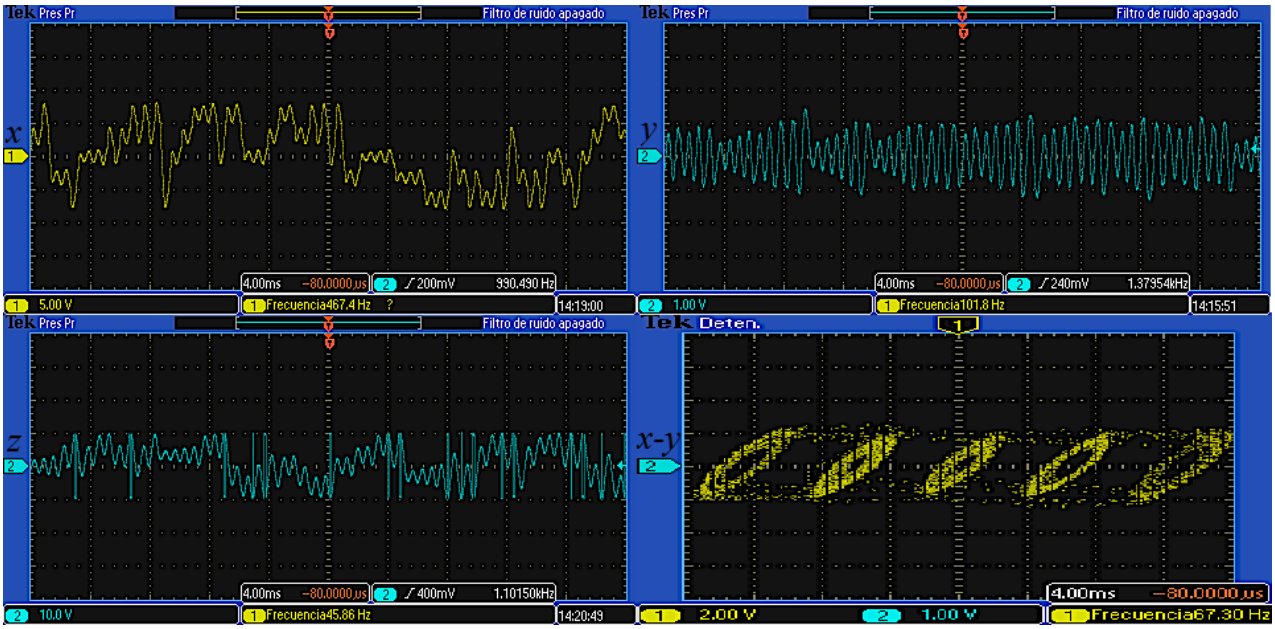

Figura 4.44. Señales utilizando RK4 en tiempo real con una función de Chua para cinco enrollamientos.

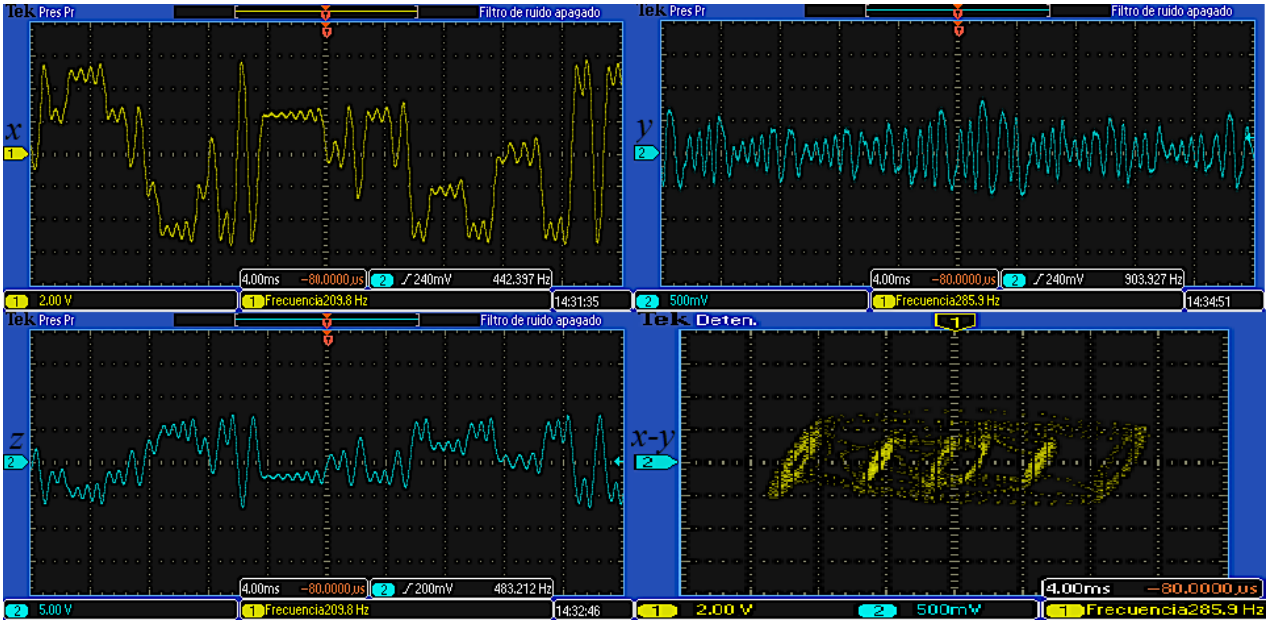

Figura 4.45. Señales utilizando RK4 en tiempo real con una función de Chua para seis enrollamientos.

Como se observa en las figuras 4.36 - 4.45, los atractores obtenidos son muy similares a los resultados ideales obtenidos en co-simulación de la sección 4.7.2 y 4.7.3, solo que obviamente tienen menor resolución debido al ruido y al muestreo de los convertidores analógico-digital del FPGA. No obstante si se comparan también con los atractores simulados en Matlab mediante código en la sección 3.3, estos resultan completamente similares.

4.8 CONCLUSIÓN.

Se llevó cabo las descripción de la metodología empleada para realizar la implementación en el dispositivo FPGA-DSP Cyclone III Edition de Altera del oscilador caótico de Chua mediante la herramienta DSP-Builder y Matlab-Simulink. Además se mostró otra metodología para la implementación en FPGA para los osciladores caóticos basados en SNLF y el circuito de Chua a través del diseño de arquitecturas mediante código VHDL, ambos tipos de osciladores resueltos mediante el método de Euler y Runge-Kutta. Los resultados fueron mostrados en co-simulación y de forma experimental, los cuales fueron satisfactorios ya que son similares a los obtenidos en el capítulo 3 donde se simulan dichos osciladores caóticos en Matlab: estas señales son factibles de aplicarse en la seguridad de sistemas de comunicación, ya que con estos diseños se puede llevar al Hardware en forma de circuito integrado, y sustituir otros osciladores caóticos ya implementados de manera analógica.

Capítulo 5

Aplicación en transmisión de imágenes

5.1 INTRODUCCIÓN

En este capítulo se muestra la sincronización de dos sistemas caóticos mediante formas hamiltonianas y observadores, los sistemas caóticos que se sincronizan están basados en SNLF y se realizan con atractores tanto de 2 como de 6 enrollamientos, ambos casos basados en un sistema maestro-esclavo, y como principal aportación se muestra la implementación en FPGA de dicha sincronización. La segunda aportación es la prueba de efectividad de la sincronización, realizando una transmisión de una imagen en escala de grises, enmascarándola de manera aditiva con la señal caótica del Maestro, y desenmascarándola con ayuda de la señal generada por el Esclavo, mostrando los resultados en co-simulación entre Active-HDL y Matlab.

5.2 SIMULACIÓN DEL PROCESO DE SINCRONIZACIÓN DE OSCILADORES CAÓTICOS.

Se simularon los osciladores caóticos sincronizados (69) y (70) de la sección 2.10, se utilizaron las ganancias obtenidas de [10] $K = (k_1,\ k_2,\ k_3)^T$ con valores de $k_1 = 2,\ k_2 = 5,\ k_3 = 7$. Se utilizaron condiciones iniciales distintas, para el sistema el sistema maestro: $x(0) = [0,\ 0,\ 0.1]$; y para el sistema esclavo: $y(0) = [1, -0.5,\ 3]$, esto con el propósito de verificar que cuando los dos sistemas se inician con condiciones iniciales diferentes, después de determinado tiempo la señal del esclavo se sincroniza con la señal del maestro mediante el observador propuesto. Como constantes se utilizaron: $a = b = c = d_1 = 0.7$. La sincronización se realiza resolviendo los sistemas maestro y esclavo mediante el método de Euler a través de código en Matlab, se utilizó un tamaño de paso *step* = 0.01.

En la figura 5.1 se muestran las trayectorias de estado de los sistemas para dos enrollamientos maestro y esclavo (69) y (70), respectivamente. Además se grafican las variables *x1* con *x2* (maestro) y *y1* con *y2* (esclavo), se aprecia claramente que los atractores son idénticos, salvo al inicio de la simulación ya que tienen condiciones iniciales diferentes.

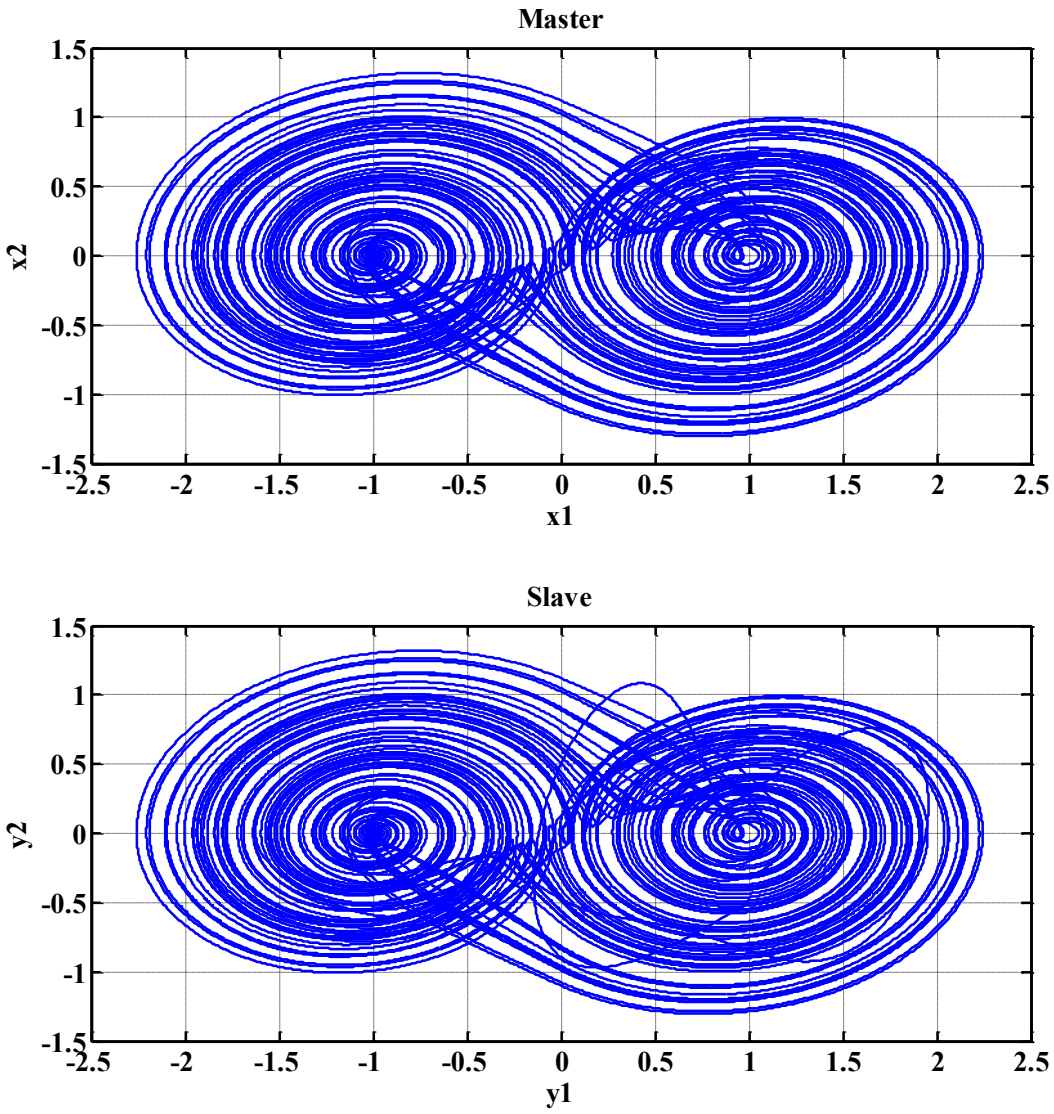

Figura 5.1. Sincronización Maestro-Esclavo de oscilador caótico de 2-enrollamientos.

En la figura 5.2 se puede observar el error de sincronización, simplemente se grafica la diferencia entre las variables de estado del maestro con las del esclavo, donde $e1 = x1 - y1$, $e2 = x2 - y2$ y $e3 = x3 - y3$, se puede apreciar que los errores de las variables de estados tardan un determinado tiempo en estabilizarse. En la figura 5.3 se muestra el error de fase entre los osciladores maestro y esclavo, se grafica *x1* con *y1*, *x2* con *y2* y *x3* con *y3*; lo ideal sería que los tres errores de fase fueran pendientes de 45° totalmente rectas y sin perturbaciones, pero debido a que los sistemas inician con condiciones iniciales diferentes, muestran un poco de perturbaciones.

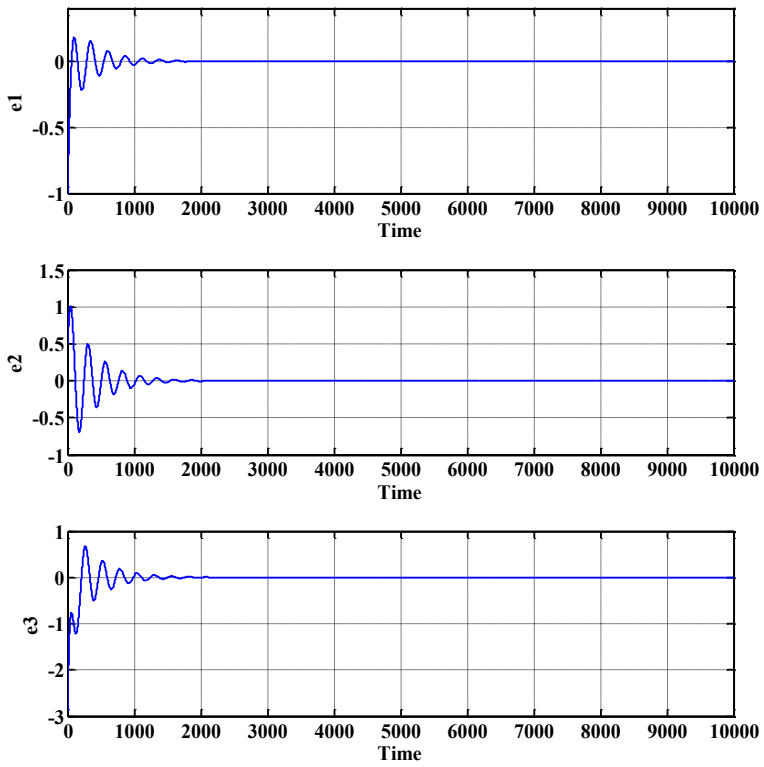

Figura 5.2. Error de sincronización entre los estados de maestro-esclavo 2-enrollamientos.

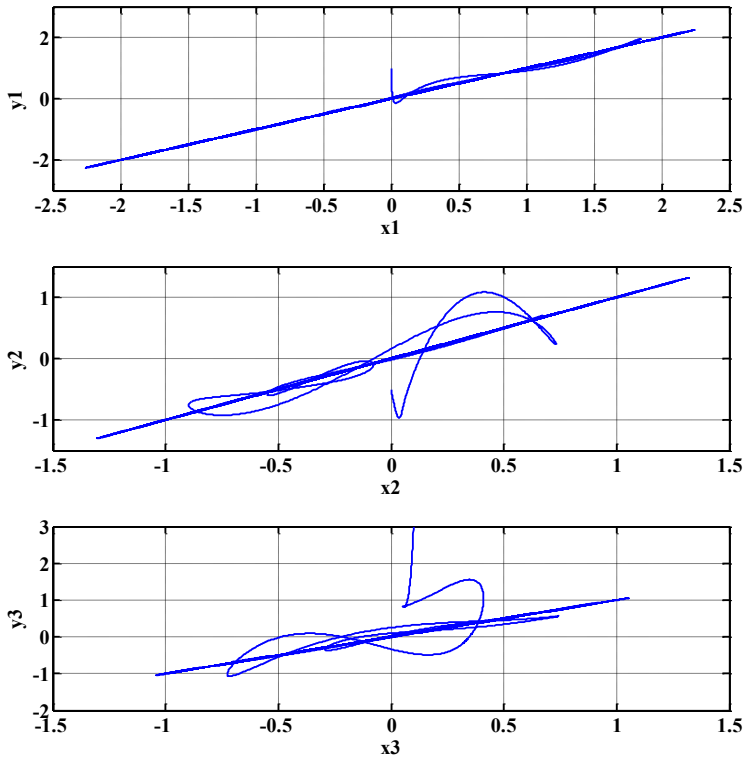

Figura 5.3. Error de sincronización entre los estados de maestro-esclavo 2-enrollamientos.

De la misma manera se muestran los resultados de la simulación en Matlab para la sincronización de los sistemas maestro-esclavo pero ahora con atractores de seis enrollamientos, se utilizan los mismos valores que para los sistemas de dos enrollamientos. En la figura 5.4, se muestran las trayectorias de estado de los sistemas para seis enrollamientos maestro y esclavo (69) y (70), respectivamente, se grafican las variables *x1* con *x2* (maestro) y *y1* con *y2* (esclavo). En la figura 5.5 se muestra el error de sincronización, graficando la diferencia entre las variables de estado del maestro con las del esclavo, donde $e1 = x1 - y1$, $e2 = x2 - y2$ y $e3 = x3 - y3$. En la figura 5.6 se muestra el error de fase entre los osciladores maestro y esclavo, se grafica *x1* con *y1*, *x2* con *y2* y *x3* con *y3*.

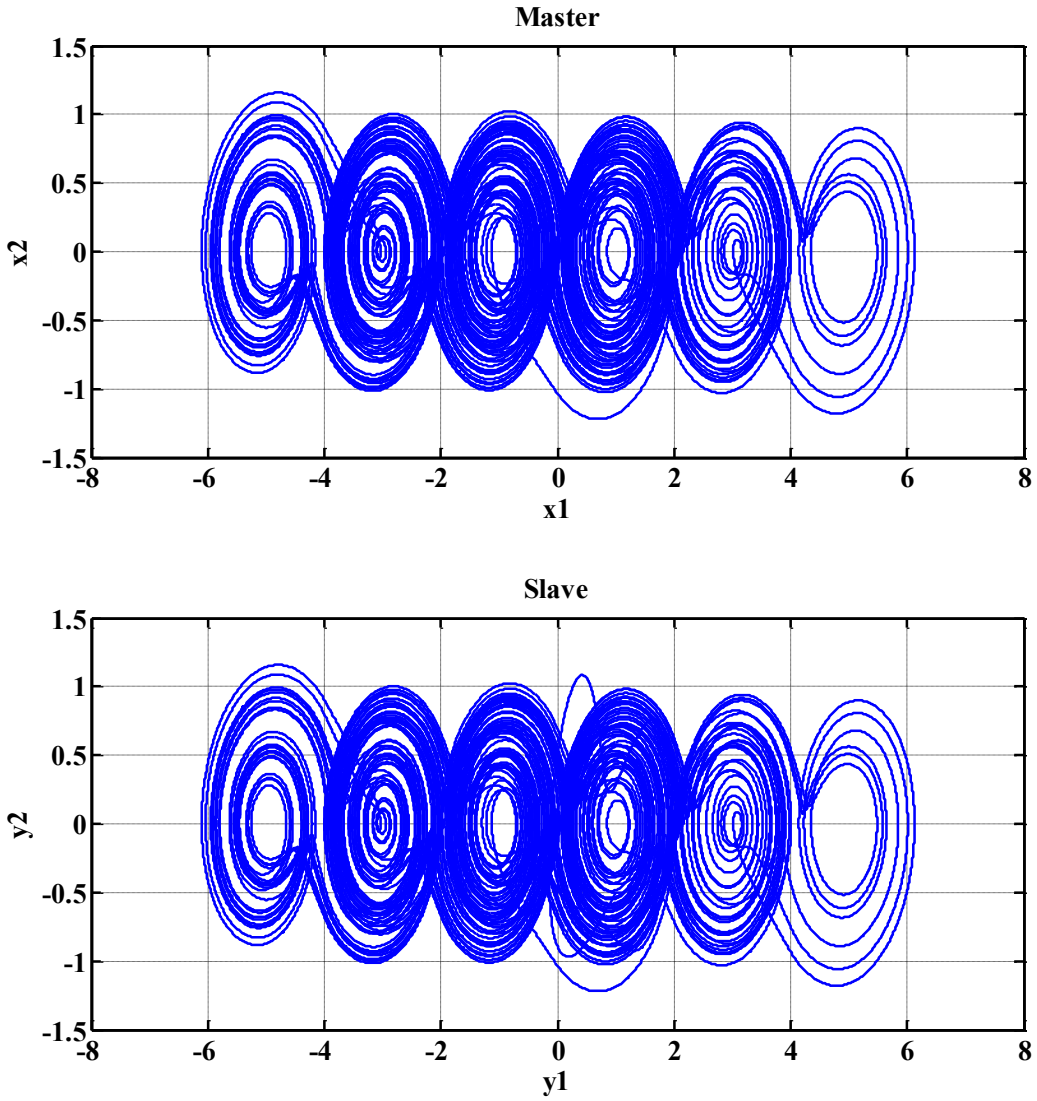

Figura 5.4. Sincronización Maestro-Esclavo de oscilador caótico de 6-enrollamientos.

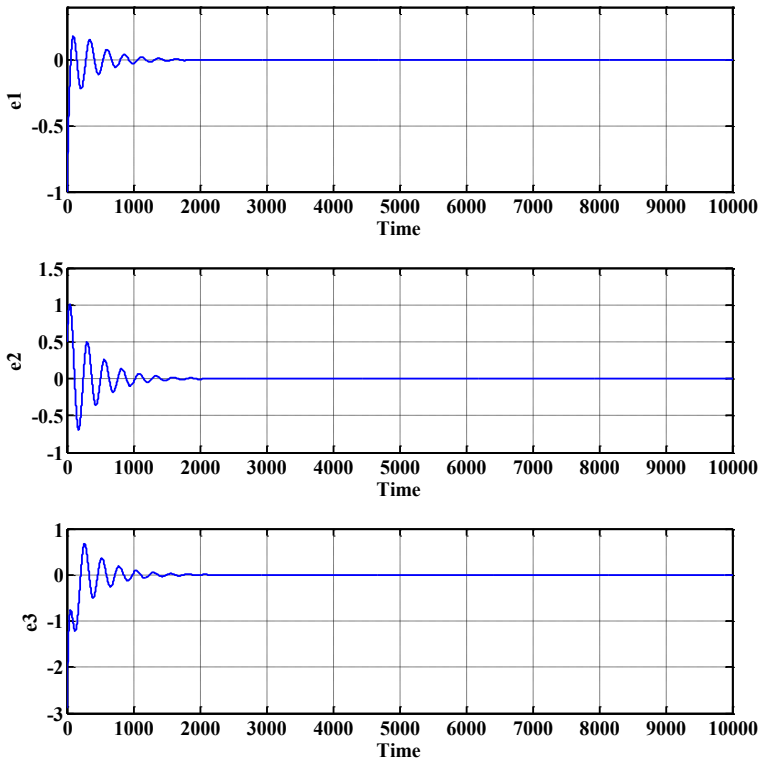

Figura 5.5. Error de sincronización entre los estados de maestro-esclavo 6-enrollamientos.

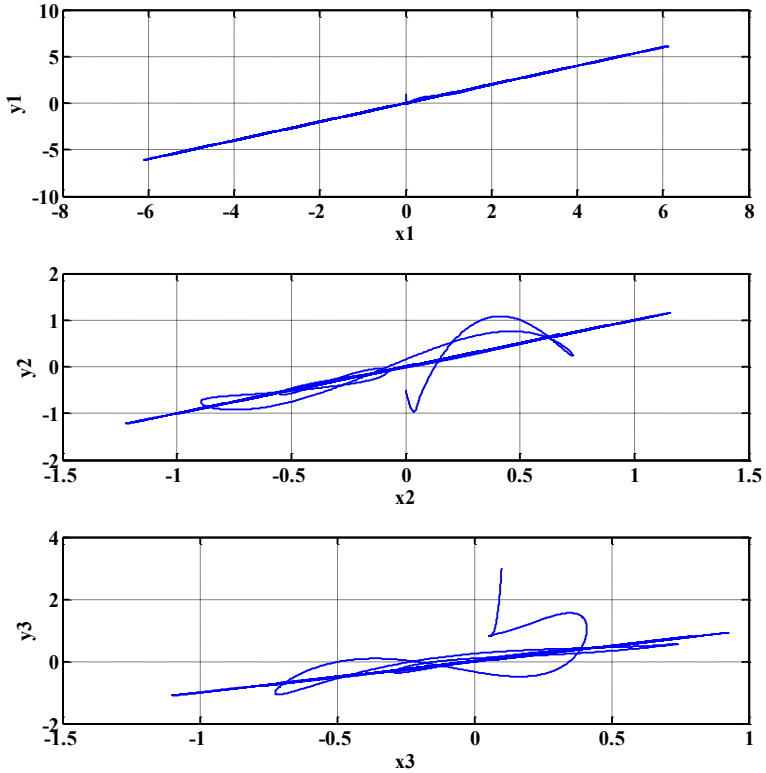

Figura 5.6. Error de sincronización entre los estados de maestro-esclavo 2-enrollamientos.

5.3 IMPLEMENTACIÓN DE SINCRONIZACIÓN DE OSCILADORES CAÓTICOS EN FPGA.

Con el propósito de resolver y discretizar los sistemas maestro-esclavo, se aplica el método de Euler a los sistemas de ecuaciones (69) y (70). Se utiliza solamente el método de Euler ya que como se observa en las secciones anteriores de la implementación, no existe tanta diferencia a simple vista entre usar un método u otro por lo que para realizar una arquitectura más simple, se elige el método de Euler hacia delante por ser más fácil de utilizar, principalmente para aquellos que inician con procesos de discretización, y porque permite obtener la solución deseada si necesidad de gran exactitud. Así mismo, al ser un algoritmo de fácil aplicación, permite al dispositivo FPGA tener una mayor velocidad de procesamiento, mientras que con Runge-Kutta puede existir una mayor complejidad de cómputo, debido a las excesivas operaciones que esta contiene.

Aplicando entonces el método de Euler a los sistemas de ecuaciones (69) y (70), se obtienen los sistemas de ecuaciones discretas:

$$\begin{aligned} x_1(i+1) &= x_1(i) + step * x_2(i) \\ x_2(i+1) &= x_2(i) + step * x_3(i) \\ x_3(i+1) &= x_3(i) + step *(-a * x_1(i) - b * x_2(i) - c * x_3(i) + d_1 f(x_1(i); k, \alpha, p, q)), \end{aligned} \tag{73}$$

$$\begin{aligned} y_1(i+1) &= y_1(i) + step * (y_2(i) + k_1 * (x_1(i) - y_1(i))) \\ y_2(i+1) &= y_2(i) + step * (y_3(i) + k_2(x_1(i) - y_1(i))) \\ y_3(i+1) &= y_3(i) + step * (-a * y_1(i) - b * y_2(i) - c * y_3(i) + d_1 f(x_1(i); k, \alpha, p, q) + k_3(x_1(i) - y_1(i))), \end{aligned} \tag{74}$$

tal y como se realizó en las secciones 4.6.1 y 4.7.1. $x_1(i)$, $x_2(i)$ y $x_3(i)$, conforman los estados del sistema maestro en tiempo discreto y $y_1(i)$, $y_2(i)$ y $y_3(i)$ los del esclavo.

Para la implementación de los sistemas de ecuaciones discretizados (73) y (74) se utilizaron nuevamente los componentes de la figura 4.13 diseñados anteriormente mediante código VHDL. Además se utilizó la misma estructura de la palabra de punto fijo usada en las implementaciones anteriores de la tabla 4.2, donde se escogieron 1 bit para el signo, 4 bits para el entero y 14 bits para la parte fraccionaria, con un total de 19 bits por cada palabra. La metodología de implementación es exactamente la misma que la seguida en la sección 4, el propósito es formar las ecuaciones (73) y (74) mediante los componentes sumador y restador definidos en la figura 4.13. Para el diseño se utilizan los mismos valores usados anteriormente en las simulaciones de Matlab mostradas en la sección 5.2. Las implementaciones de las ecuaciones (73) y (74) se muestran en los diagramas de bloques de las figuras 5.7 y 5.8, maestro y esclavo respectivamente.

En la figura 5.7 se puede observar que $x_1(i)$, $x_2(i)$ y $x_3(i)$ representan los estados presentes del sistema y $x_1(i+1)$, $x_2(i+1)$ y $x_3(i+1)$ los estados futuros. Nuevamente se utilizó un comparador, el cual como se mencionó en la sección 4, este es uno de los componentes más importantes del sistema ya que en él se define el comportamiento de la SNLF, para efectos prácticos solo se muestra en diagrama de bloques la implementación para dos enrollamientos.

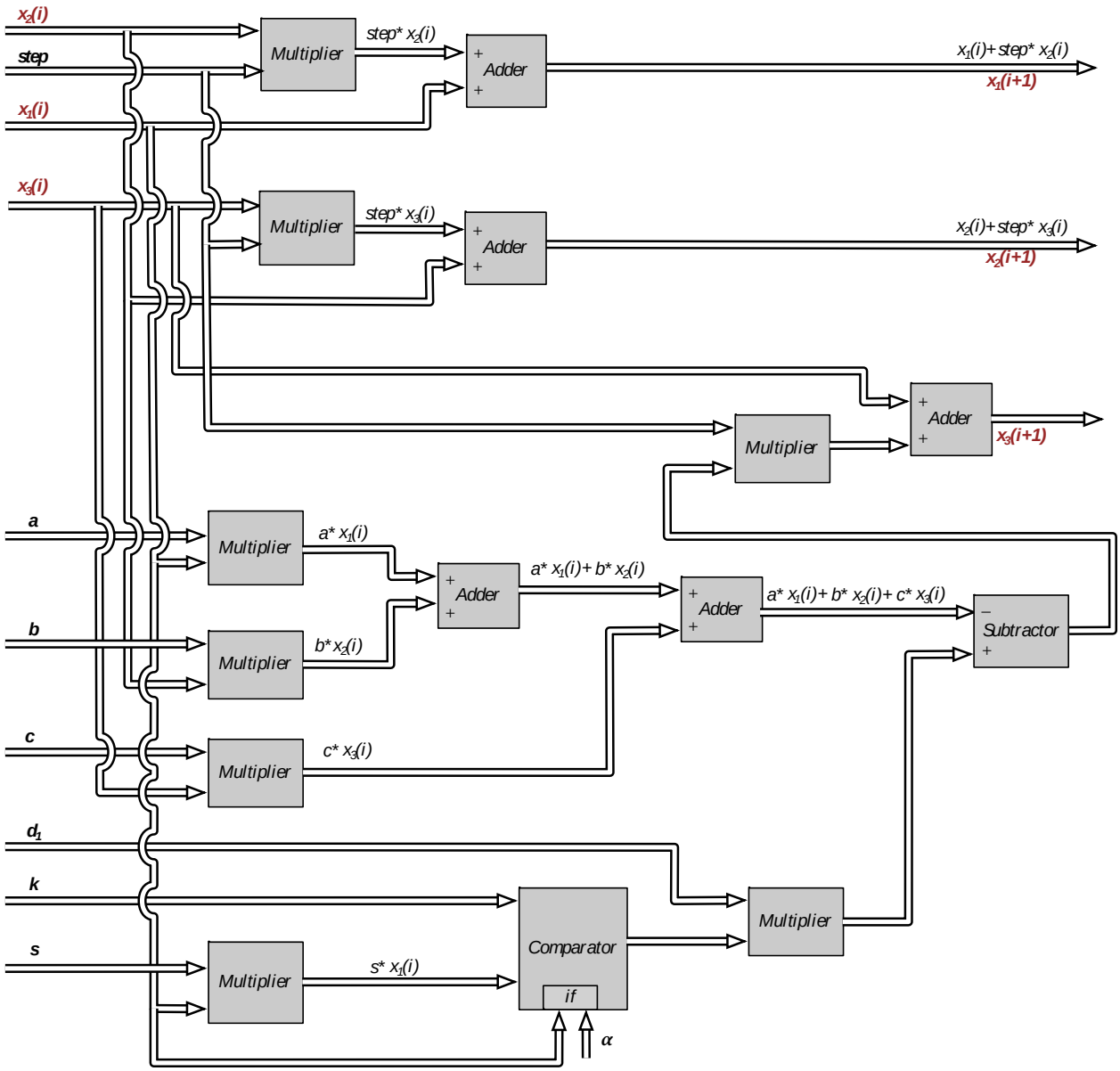

Figura 5.7. Diagrama de bloques del sistema de ecuaciones (73).

De la misma manera en la figura 5.8 $y_1(i)$, $y_2(i)$ y $y_3(i)$ representan los estados presentes del sistema y $y_1(i+1)$, $y_2(i+1)$ y $y_3(i+1)$ los estados futuros. En el diagrama se observa que el sistema depende claramente del estado $x_1(i)$, del maestro.

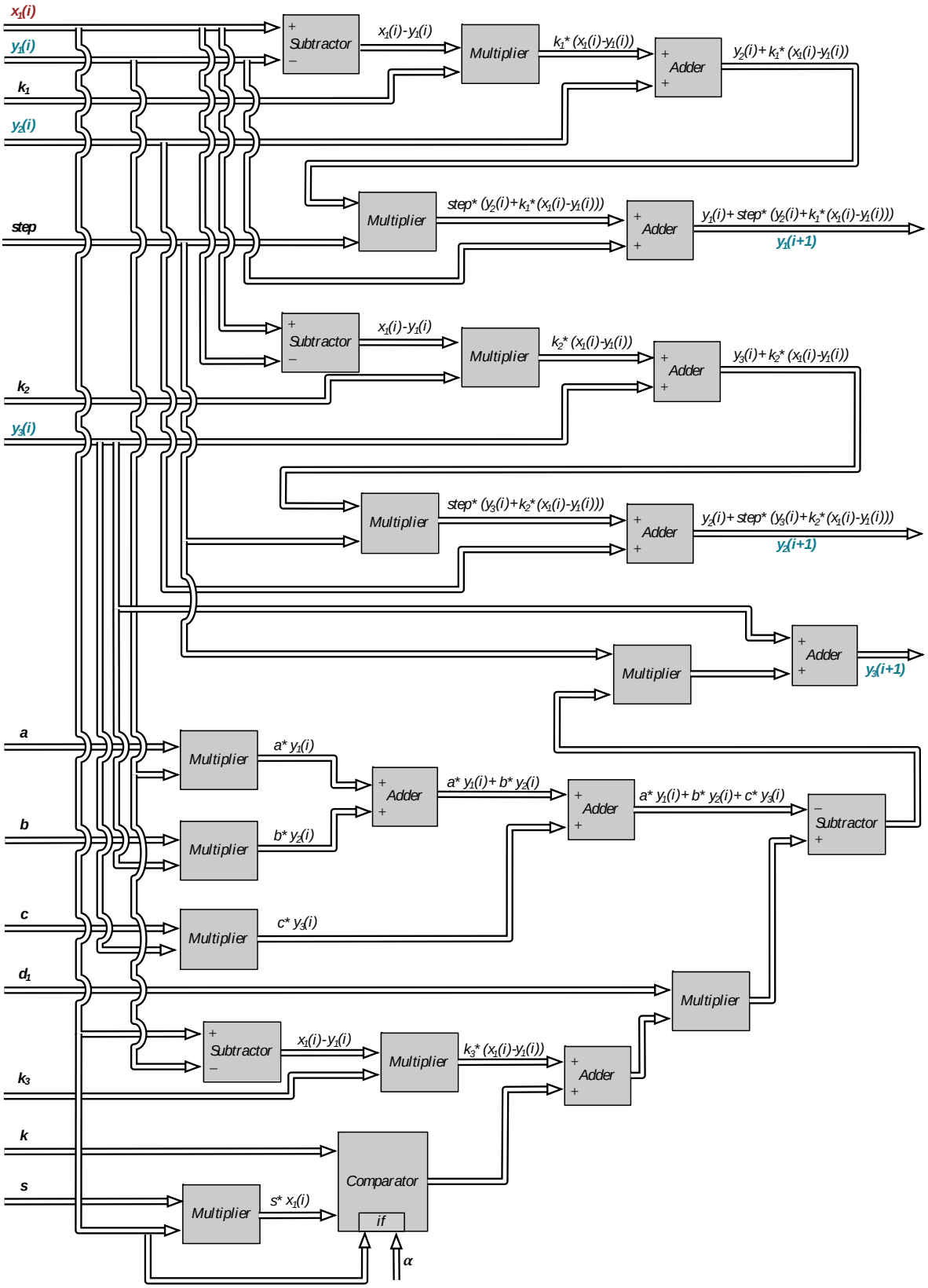

Figura 5.8. Diagrama de bloques del sistema de ecuaciones (74).

Los diagramas de bloques de las figuras 5.7 y 5.8 muestran así los sistemas implementados a partir de las ecuaciones (73) y (74), sin embargo este todavía está a un tercer nivel del diseño. Para que el sistema se sincronice se deben unir los dos sistemas Maestro y Esclavo tal y como se muestra en la figura 12, donde X_{inim}, Y_{inim} y Z_{inim} son las señales de las condiciones iniciales de la Unidad Maestro, estas representan a los estados presentes $x_1(i)$, $x_2(i)$ y $x_3(i)$ de la figura 5.7, de la misma manera las condiciones iniciales X_{inis}, Y_{inis} y Z_{inis} representan a $y_1(i)$, $y_2(i)$ y $y_3(i)$ de la figura 5.8, naturalmente las salidas X_{inim}, Y_{inim}, $Z_{inim,}$ X_{inims}, Y_{inims} y Z_{inis} representan los estados futuros $x_1(i+1)$, $x_2(i+1), x_3(i+1)$, $y_1(i+1)$, $y_2(i+1)$ y $y_3(i+1)$, también se observan las señales de entrada *Clock*, *Reset* de 1 bit, las cuales se han utilizado para garantizar la sincronización de los componentes dentro de la unidad.

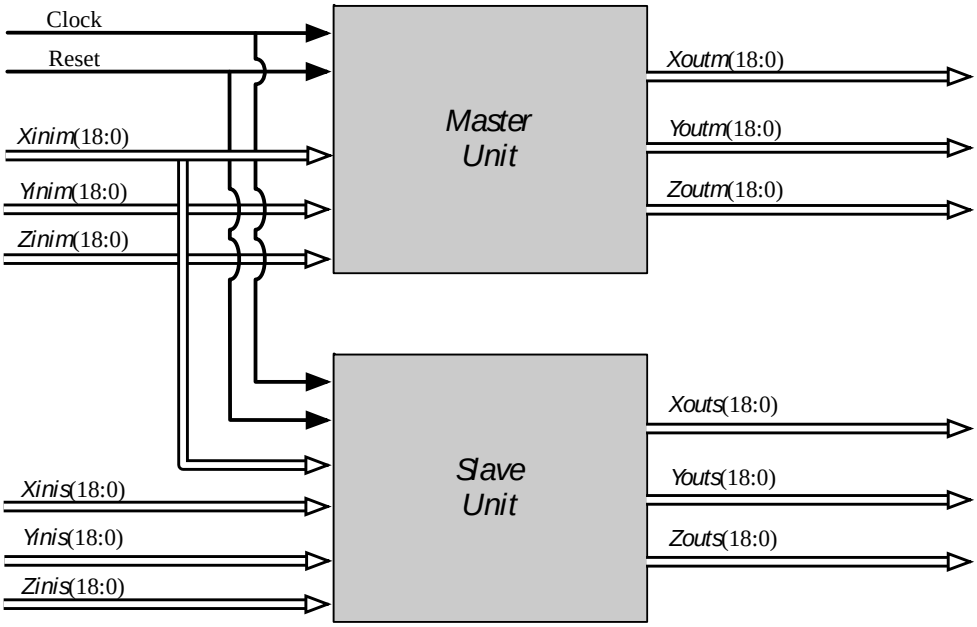

Figura 5.9. Unidad de sincronización Maestro-Esclavo.

Si el sistema se implementa tal y como se muestra en la figura 5.9, solo se obtendrá la primera iteración de las soluciones, ya que aún no cuenta con retroalimentación, para ello se necesita de un multiplexor.

En la figura 5.10 se muestra el diagrama de bloques de los sistemas en su segundo nivel de diseño, en él se observan seis unidades que componen el modelo: unidad maestro, unidad esclavo, multiplexor, registros y restadores. La unidad de multiplexor se encarga de mantener el sistema en un bucle lo cual permite las iteraciones, esta unidad le proporciona a las unidades maestro y esclavo las condiciones iniciales, estas condiciones permiten obtener la primera iteración, la cual necesita de ocho ciclos de reloj debido a todas las operaciones necesarias, después de eso el sistema necesita retroalimentarse por lo que después de los 8 ciclos de reloj, mediante un contador se manda la señal *Sel* de un bit al Multiplexor, esta señal sirve para indicar que se empiece la retroalimentación del sistema y que de ahí en adelante las condiciones iniciales serán las mismas salidas *Xout_Master*, *Yout_Master*, *Zout_Master*, *Xout_Slaves*, *Yout_Slave* y *Zout_Slave* de los sistemas caóticos; la unidad de registros se encarga de capturar las señales X_{outm}, Y_{outm}, X_{outs}, Y_{outs} y dar el tiempo necesario

para que las señales Z_{outm} y Z_{outs} estén listas, debido a que estas señales necesitan de más operaciones lo cual conlleva a mas ciclos de reloj (ocho ciclos); los restadores se utilizan para obtener el error de sincronización de los estados, tal y como se hizo en Matlab, restando las señales: *Xout_Master-Xout_Slave* , *Yout_Master-Yout_Slave* y *Zout_Master-Zout_Slave*.

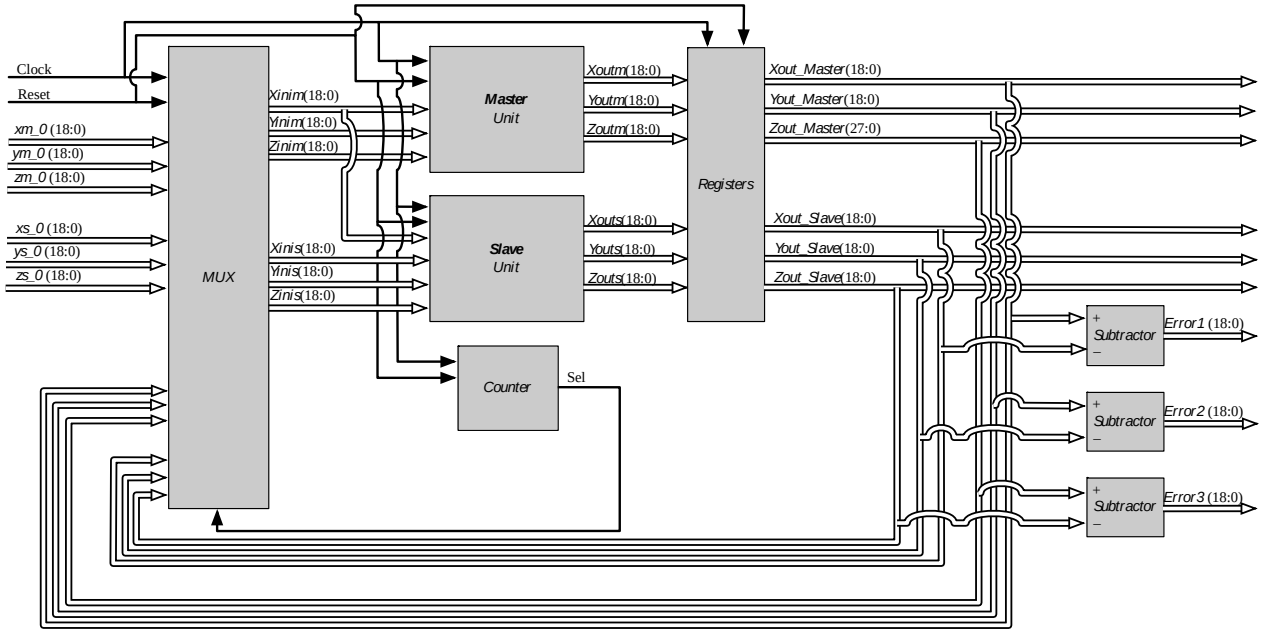

Figura 5.10. Diagrama de bloques de sincronización Maestro-Esclavo retroalimentado.

Finalmente en la figura 5.11 se muestra la caja negra del sistema de la sincronización completo, el cual indica el Top-level del diseño, es decir, el primer nivel. Las condiciones iniciales son embebidas dentro del diseño para mayor simplicidad.

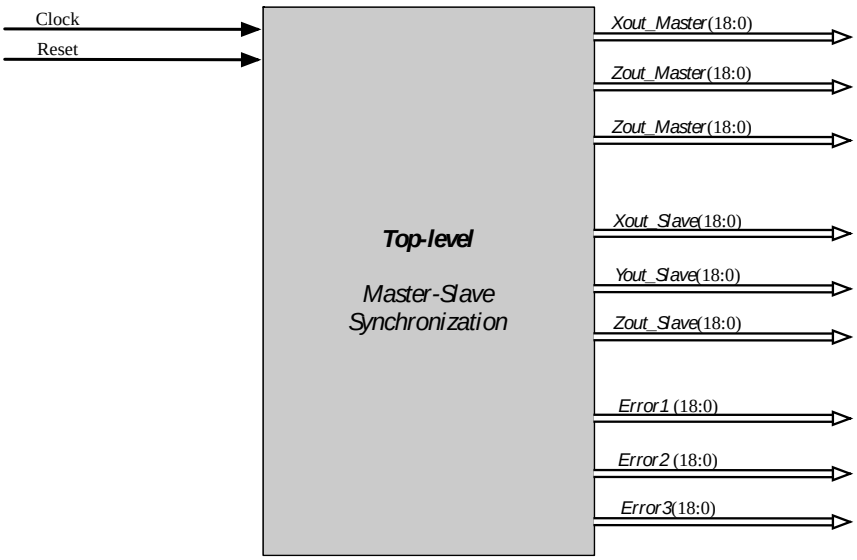

Figura 5.11. Top-level del diseño de Sincronización Maestro-Esclavo.

5.3.1 Co-simulación Active-Matlab.

A través del software Active, se generó el archivo *.m del Top-level del sistema diseñado anteriormente nuevamente con el nombre de "*osc"*. La integración de los bloques en Simulink es presentada en la figura 5.12.

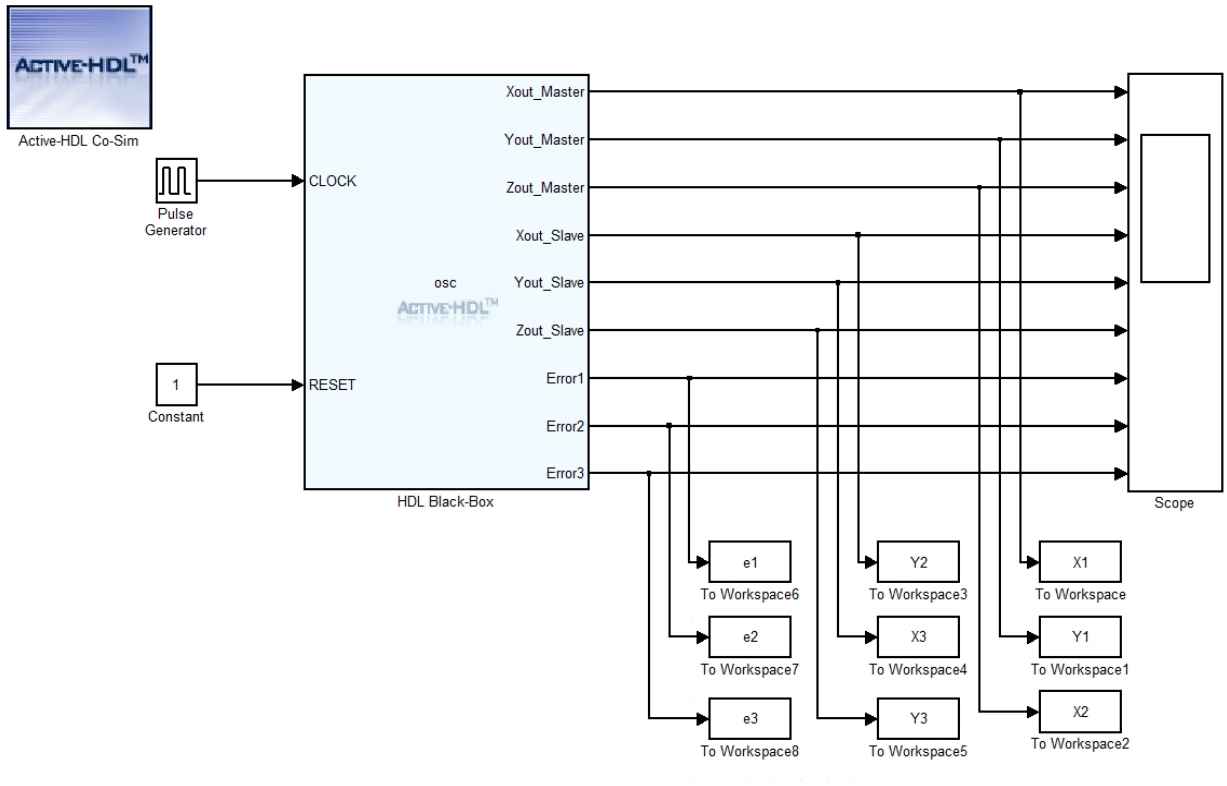

Figura 5.12. Sincronización Maestro-Esclavo en Co-simulación con Simulink.

Para la señal de reloj "*CLOCK"* se utilizó un generador de pulsos con una frecuencia de 10 kHz, y para la señal de reinicio "*RESET"* simplemente una constante con el valor lógico de "1", también se utilizó un bloque *Scope* para visualizar las señales resultantes, y los bloques "To workspace" los cuales capturan cada uno de los datos obtenidos y los almacena en vectores (*arrays*), para visualizar los datos de forma numérica o simplemente graficarlos. Se utilizó un tiempo de simulación de $2x10^4$ segundos, y como se está utilizando un paso *step=0.01*, se generaron un total de $2x10^6$ datos.

A través de la co-simulación se obtuvieron los resultados de la implementación en VHDL, se almacenaron cada una de las señales en el Workspace de Matlab, en la figura 5.13 se muestran las trayectorias de estado de los sistemas para el caso dos enrollamientos maestro y esclavo, se graficaron las variables *X1* con *X2* (maestro) y *Y1* con *Y2* (esclavo). Se aprecian claramente que los atractores son idénticos, salvo al inicio de la simulación ya que tienen condiciones iniciales diferentes, estos resultados son similares a los obtenidos en simulación en Matlab.

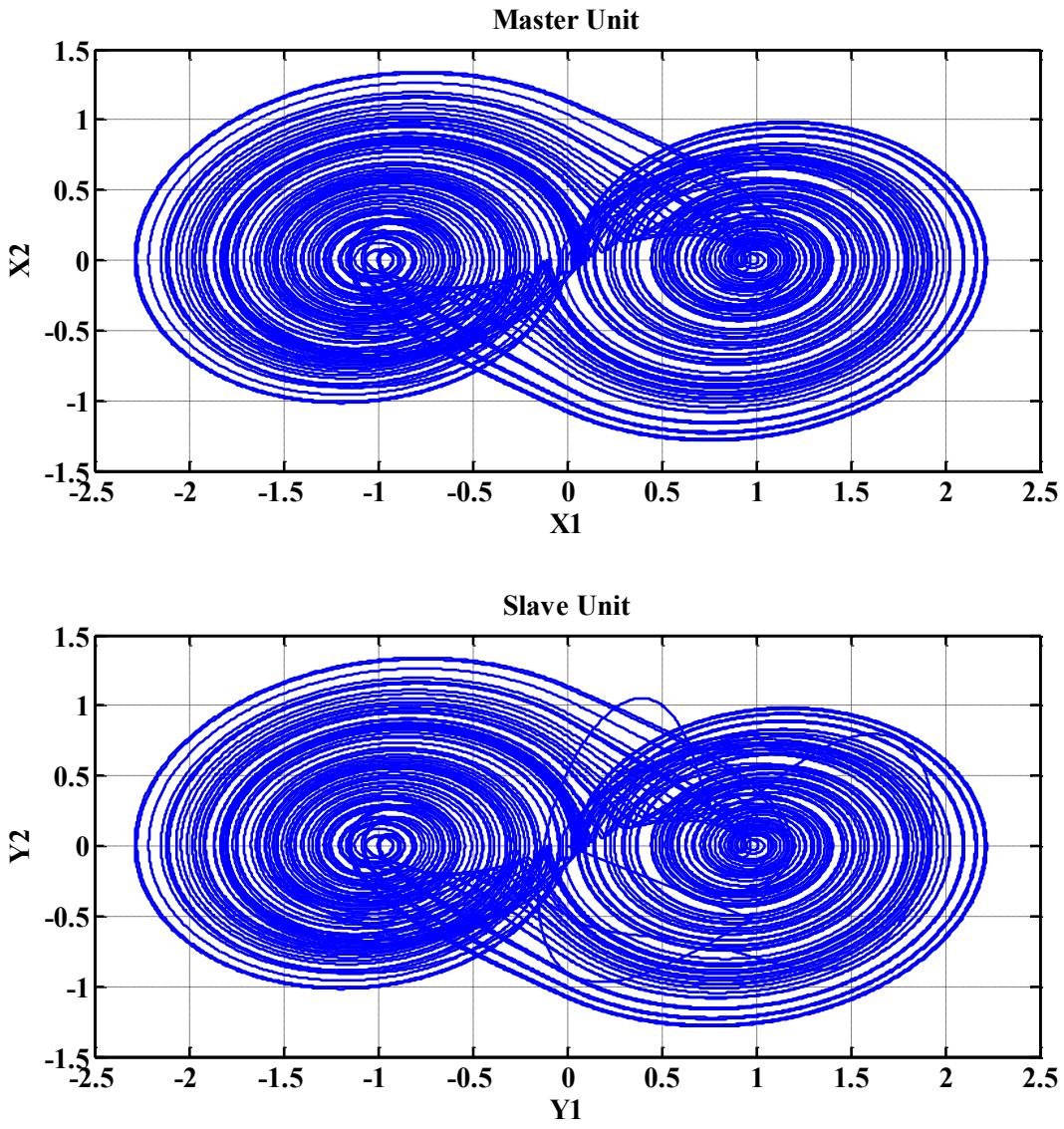

Figura 5.13. Co-simulación sincronización Maestro-Esclavo 2-enrollamientos.

En la figura 5.14 se muestra el error de sincronización, se aprecia que los errores de las variables de estados tardan un determinado tiempo en estabilizarse tal y como en la simulación en Matlab. En la figura 5.15 se muestra el error de fase entre los osciladores maestro y esclavo, se grafica *X1* con *Y1*, *X2* con *Y2* y *X3* con *Y3*, lo ideal sería que los tres errores de fase, fueran pendientes de 45° totalmente rectas y sin perturbaciones, pero estas pendientes al igual que en la simulación en Matlab muestran algunas perturbaciones debido a que cuentan con condiciones iniciales diferentes.

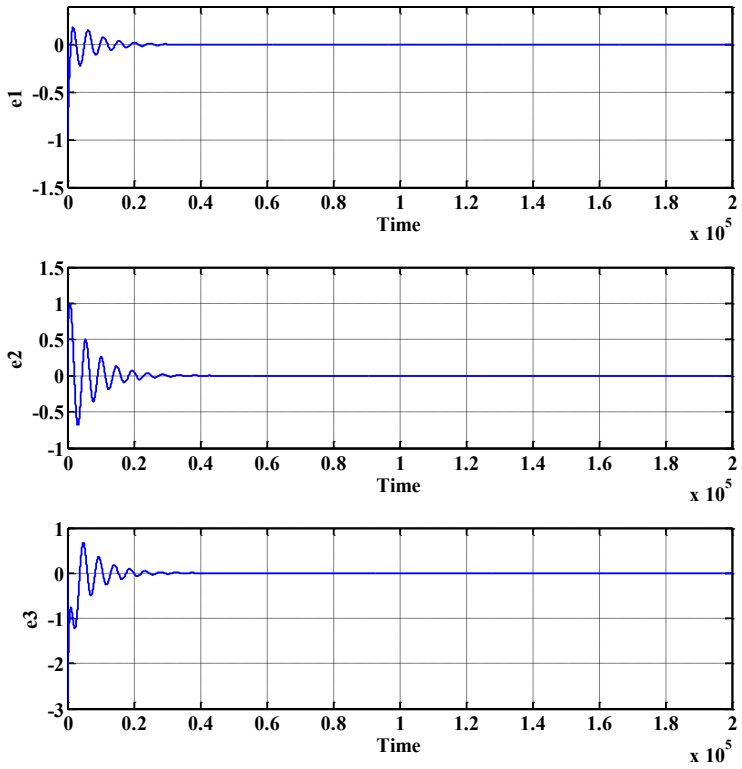

Figura 5.14. Co-simulación sincronización Maestro-Esclavo 2-enrollamientos.

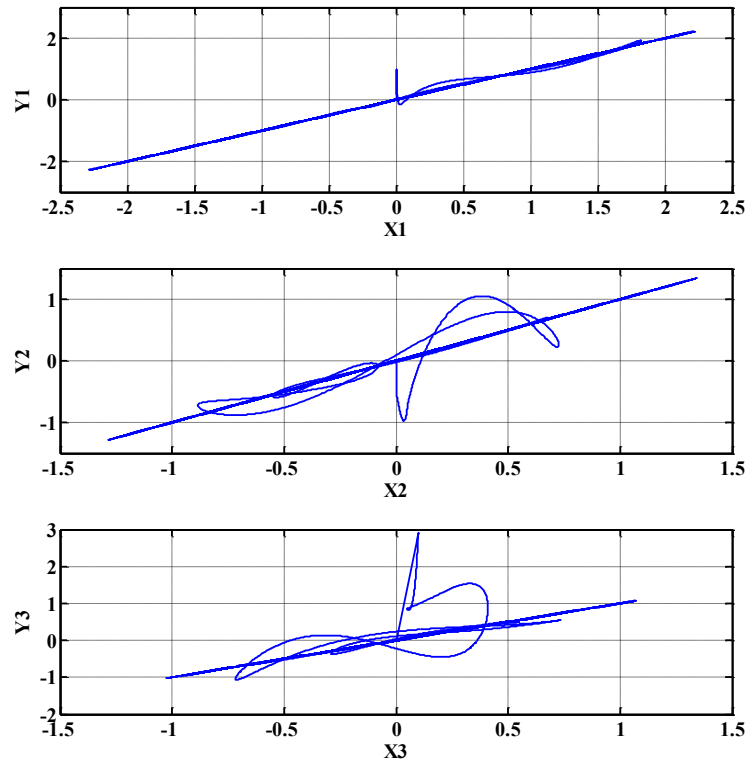

Figura 5.15. Co-simulación de error de los diagramas de fase de los estados con 2-enrollamientos.

De la misma manera en la figura 5.16, 5.17 y 5.18 se muestran, las trayectorias de estado de los sistemas maestro-esclavo para el caso de seis enrollamientos, su error de sincronización y su error de fase entre los sistemas maestro-esclavo, respectivamente.

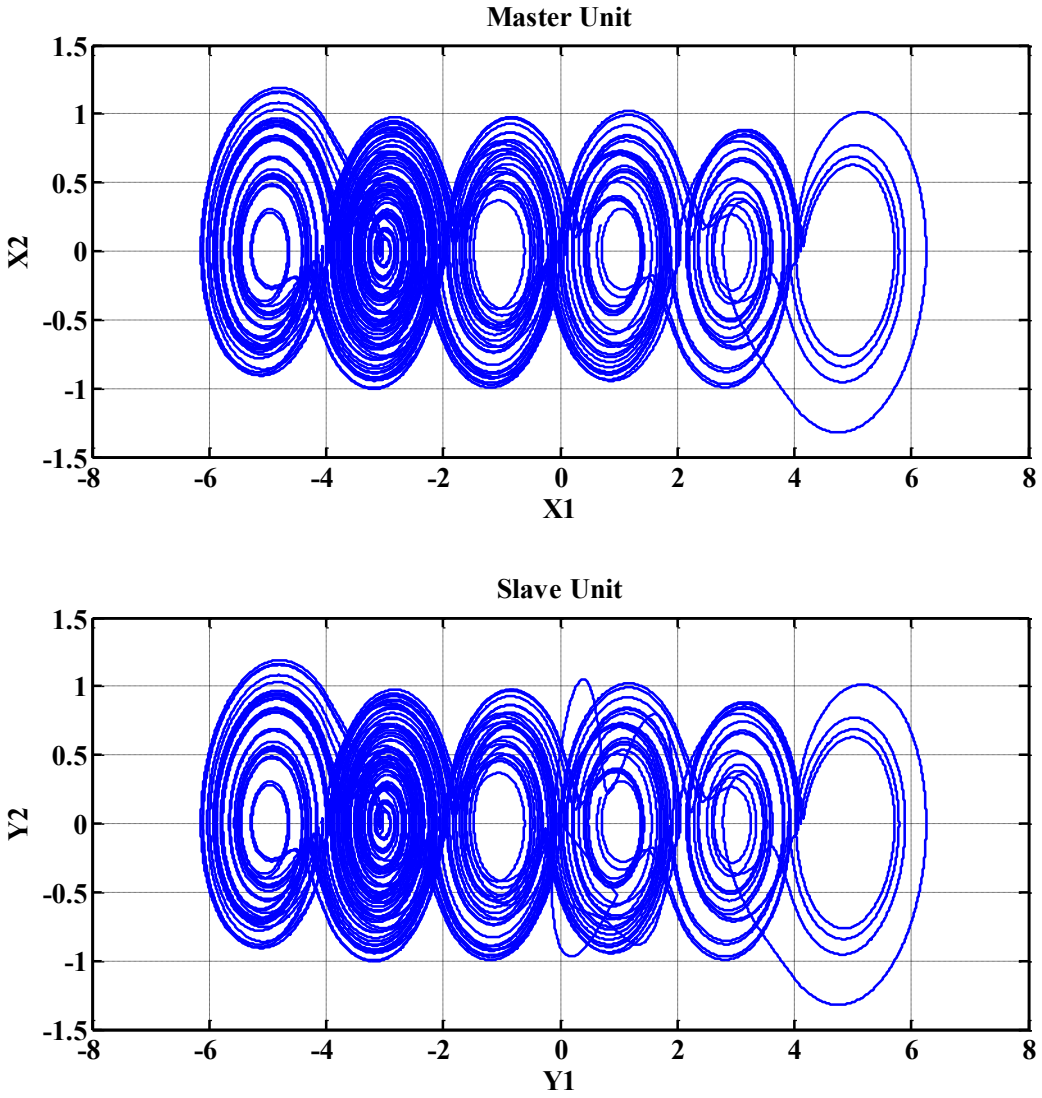

Figura 5.16. Co-simulación sincronización Maestro-Esclavo 6-enrollamientos.

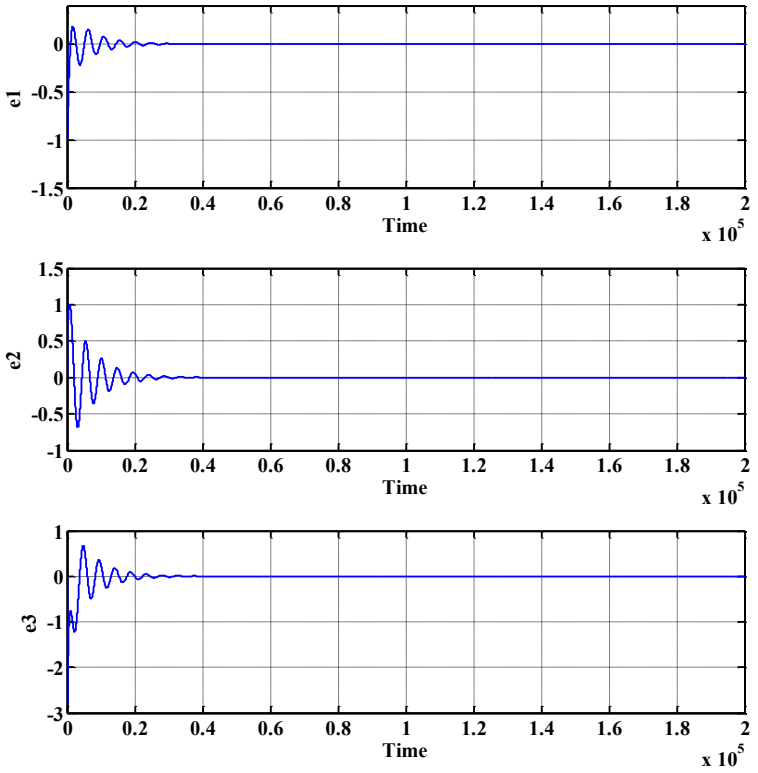

Figura 5.17. Co-simulación de error de sincronización entre los estados de maestro-esclavo 6-enrollamientos.

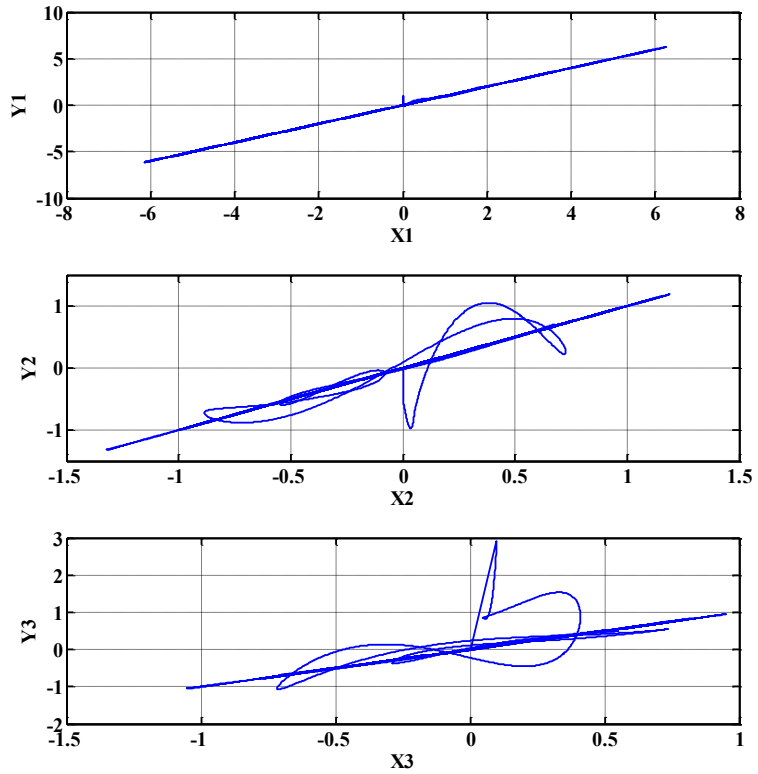

Figura 5.18. Co-simulación de error de los diagramas de fase de los estados con 6-enrollamientos.

5.4 TRANSMISIÓN DE IMÁGENES

Los sistemas caóticos sincronizados diseñados tienen muchas aplicaciones en el campo de la criptografía, esto debido a que hoy en día se transmite mucha información a través de canales de comunicación los cuales son inseguros, como la Internet, donde la información puede ser interceptada por un usuario no autorizado a la red, y lamentablemente esta información puede ser descifrada. Para probar que la sincronización de los sistemas caóticos diseñados anteriormente funciona correctamente, se realizó la transmisión de dos imágenes de diferente tamaño, una imagen monocromática de 16x16 llamada "*cohete.jpg*" y una imagen en tono de grises de 480x640 llamada "*Einstein.jpg*".

Las señales de las imágenes se añadieron a la señal caótica del sistema maestro, de esta manera la señal resultante tomó la caoticidad de la señal caótica y se transmitió, posteriormente la señal se recibió, se realizó el proceso contrario y se le restó la señal caótica del esclavo, obteniendo así la señal de la imagen. Las señales caóticas que se utilizaron en este caso son las *Xout_Master* y *Xout_Slave* del sistema diseñado de la figura 5.11, estas son las que presentaron menos perturbaciones (ver figuras 5.15 y 5.18), esto indica que son las que tienen menos error de sincronización en fase ya que son los que más se asemejan a la pendiente ideal.

Como modo de prueba la imagen fue incluida en el diseño de los sistemas sincronizados anteriormente, para poder incluir la imagen en el diseño digital se convirtió la imagen que se transmitió a palabras en bits, para esto se utilizó Matlab, que mediante código en un archivo *.m* permitió convertir una cadena de palabras hexadecimales a una escala de 16 bits, para que así pudiera ser procesada junto con el diseño de los sistemas sincronizados en FPGA.

Para la imagen de "*cohete.jpg*" de 16x16 bits se obtuvieron 256 palabras en hexadecimal de 16 bits cada una, y para la imagen de "*Einstein.jpg*" 307,200 palabras. Estas cadenas en hexadecimales se almacenaron en una ROM diseñada mediante código VHDL e incluida también en el diseño. La ROM se realizó de manera que en la salida de ésta se envíen las tramas almacenadas de la imagen una tras otra, tomando en cuenta que se tenían que sincronizar con la salida *Xout_Master* para la suma, ya que como se sabe los sistemas diseñados iteran cada 8 ciclos de reloj, por lo que también cada 8 ciclos de reloj se tenían que enviar las tramas de la ROM. Posteriormente se realizó la suma de la salida de la ROM y la señal *Xout_Master* mediante un sumador, el cual se activó con una condición que es cuando la salida *Error1* del Top-level del diseño digital (figura 14) es cero '0' o sea el error de sincronización entre *Xout_Master* y *Xout_Slave*, esto se realizó para asegurar que la transmisión se realizaría sin error de sincronización, ya que como se comentó anteriormente se tarda un determinado tiempo para que se sincronicen por completo los dos osciladores, en otras palabras que el error se estabilice (ver figuras 5.14 y 5.17), concurrentemente la salida del sumador entró a un restador para restarle la señal *Xout_Slave*, los datos obtenidos del restador que son los de la imagen recibida se almacenaron en una memoria RAM.

En la figura 5.19 se muestra el diagrama a bloques de la representación de la transmisión de la imagen, y en la figura 5.20 ya se muestra el Top-level del diseño, mostrando solamente la caja negra de todo el sistema de la transmisión.

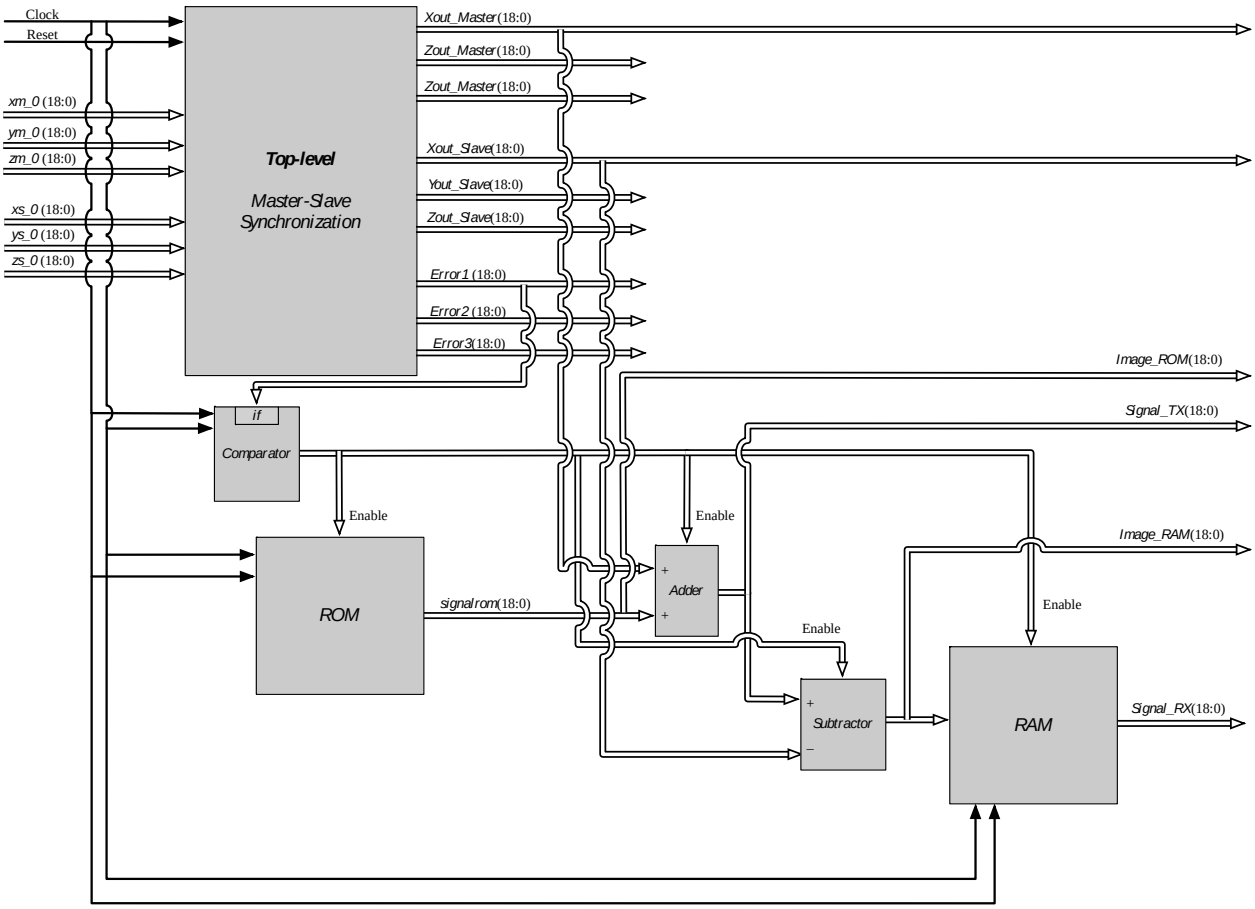

Figura 5.19. Diagrama a bloques de implementación de la transmisión de una imagen.

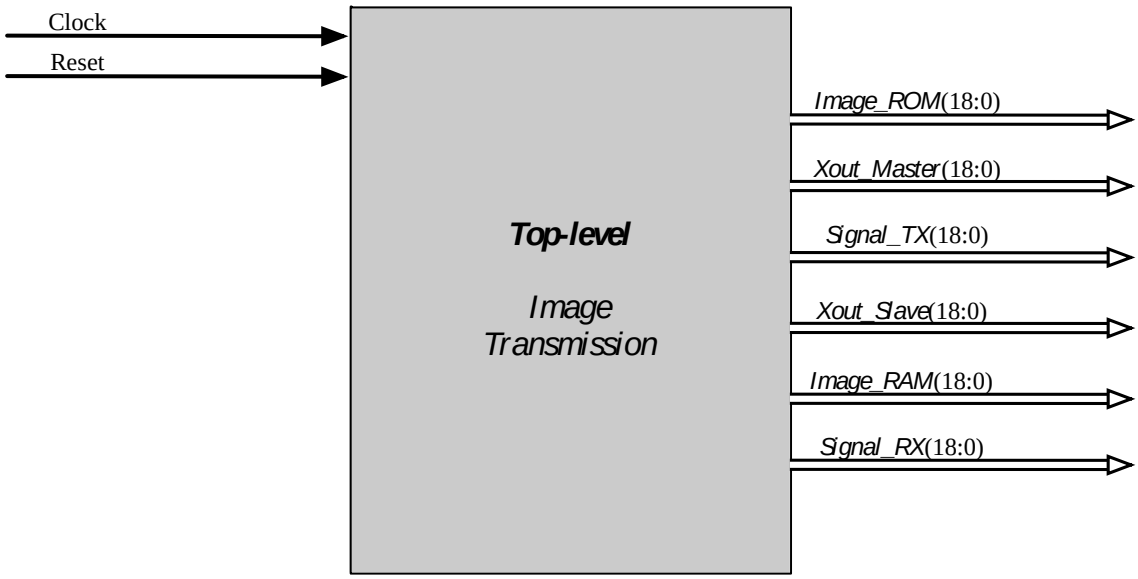

Figura 5.20. Top-level de implementación de la transmisión de una imagen.

Nuevamente se realizó la co-simulación del sistema, en este caso con un tiempo de simulación de $70x10^3$ segundos, observando así las señales de forma numérica en Matlab, el diagrama en Simulink del Top-level importado desde Active se muestra en la figura 5.21, se observa que la señales obtenidas se almacenaron en el "workspace" (espacio de trabajo) de Matlab para poder analizarlas con más detenimiento. En la figura 5.22 se observa co-simulación con Active HDL-Simulink, esto para el caso utilizando el sistema sincronizado con dos enrollamientos, se acota toda la simulación solo en el periodo donde se transmite la imagen de 256 bits para una mejor visualización. En la figura 5.23 se muestra la comparación de las imágenes en el proceso de transmisión, la precargada, transmitida y recibida. En la tabla 5.1 se muestran los recursos utilizados si el diseño se implementa en un FPGA Stratix IV: EP4SGX230KF40C2.

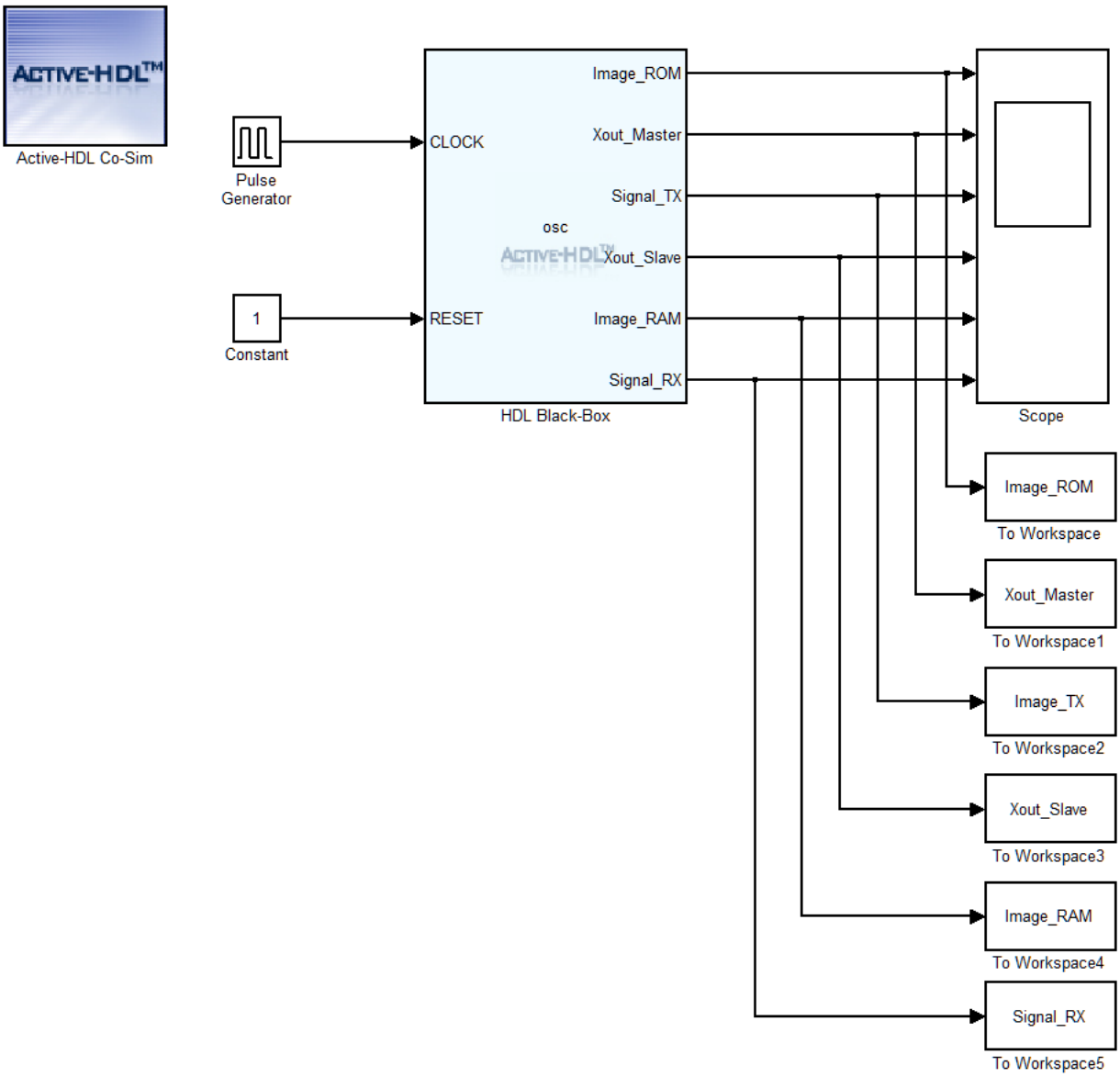

Figura 5.21. Diagrama de transmisión de Imagen en Co-simulación con Simulink.

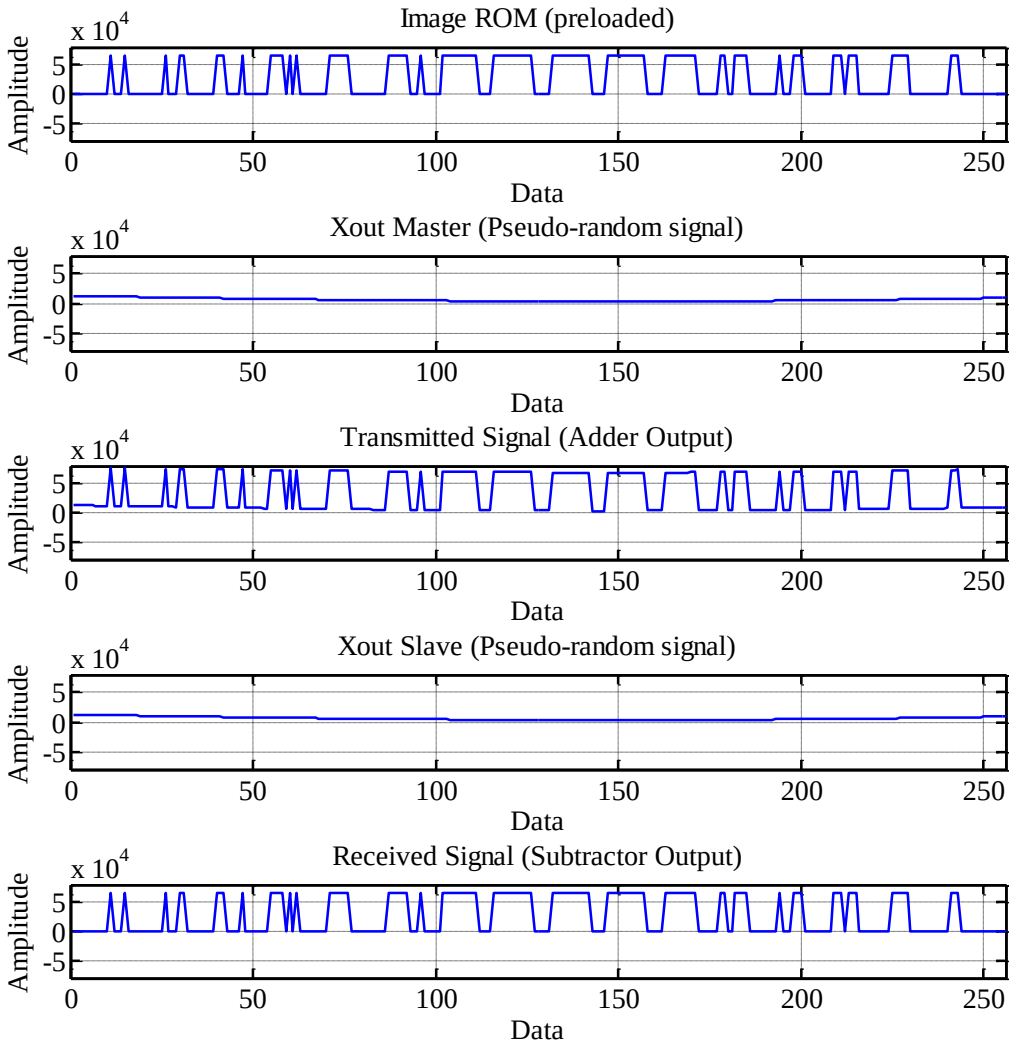

Figura 5.22. Co-simulación de transmisión de imagen 16x16 bits utilizando señales caóticas de 2 enrollamientos.

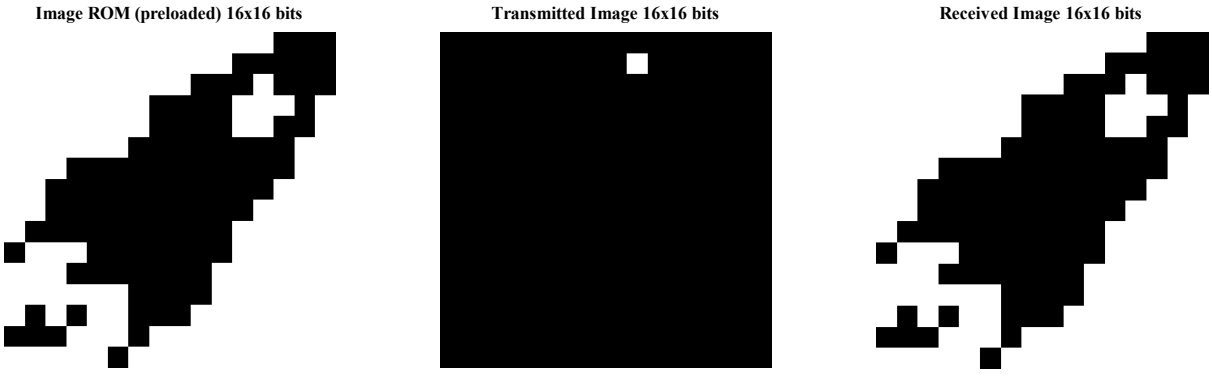

Figura 5. 23. Comparación de Imagen 16x16 bits en proceso de transmisión utilizando señales caóticas de 2 enrollamientos.

Tabla 5.1. Reporte final de recursos en transmisión de imagen de 16x16 bits utilizando señales caóticas de 2 enrollamientos.

Recursos	Usados	Disponibles	Utilización
ALUTs Combinacionales	528	182,400	< 1%
Memoria de ALUTs	0	91,200	0%
Total de registros	549	182,400	< 1%
Total de pines	192	888	22%
Total bloques de bits de memoria	4,883	14,625,792	< 1%
Bloques de DSP elementos de 18-bits	64	1,288	5%

De la misma manera se realizó la co-simulación de la transmisión de la imagen pero ahora encriptando la imagen con señales caóticas de 6 enrollamientos. En la figura 5.24 y 5.25 se muestran la co-simulación y la comparación de la imagen a lo largo del proceso de transmisión y en la tabla 5.2 la tabla de recursos utilizados, utilizando el mismo FPGA.

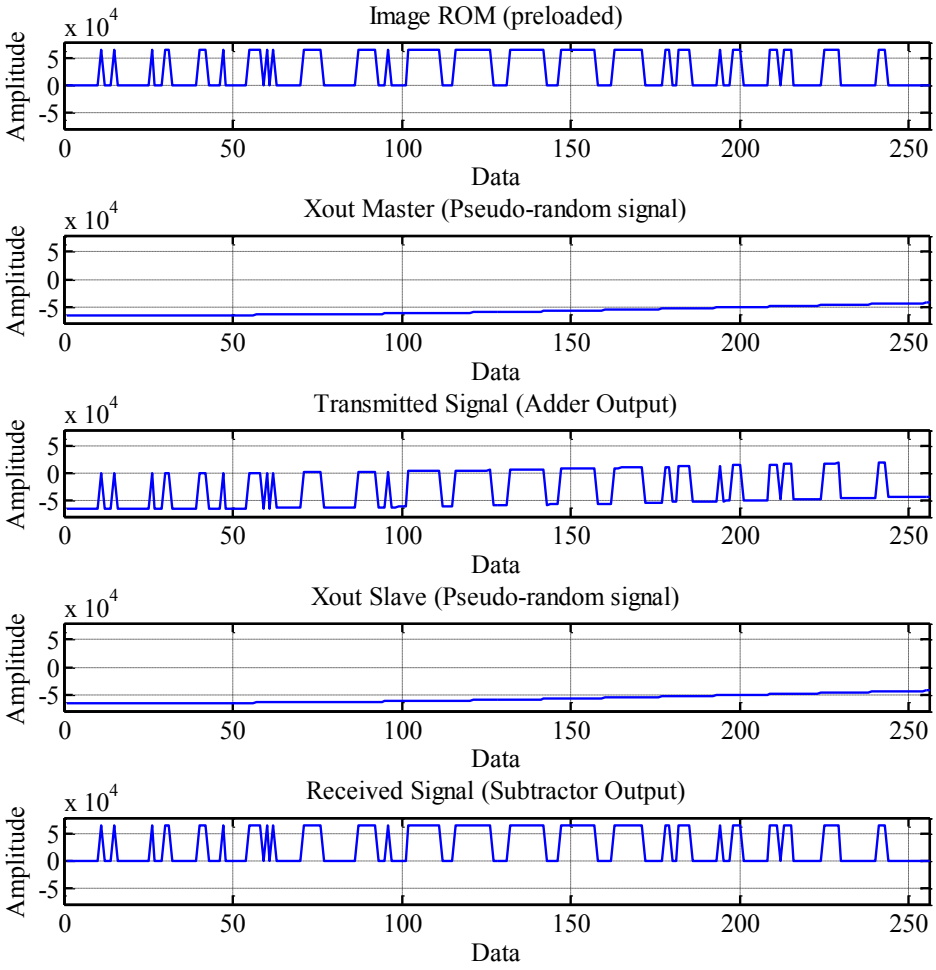

Figura 5.24. Co-simulación de transmisión de imagen 16x16 bits utilizando señales caóticas de 6 enrollamientos.

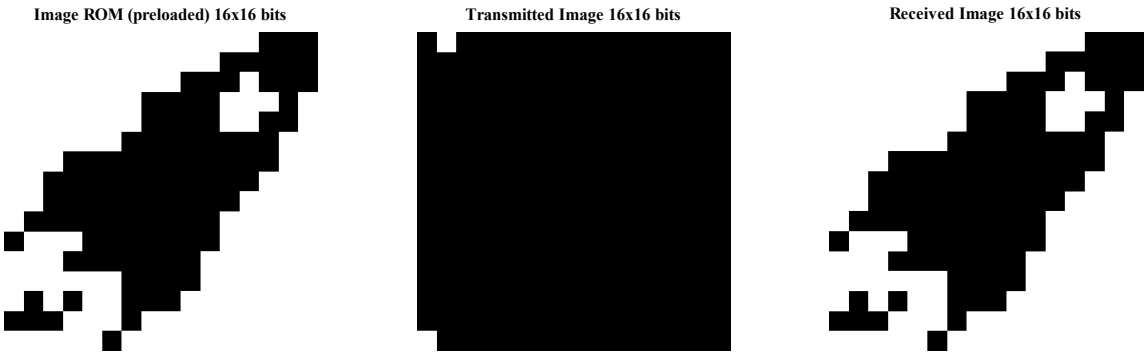

Figura 5.25. Comparación de Imagen 16x16 bits en proceso de transmisión utilizando señales caóticas de 6 enrollamientos.

Tabla 5.2. Reporte final de recursos en transmisión de imagen de 16x16 bits utilizando señales caóticas de 6 enrollamientos.

Recursos	Usados	Disponibles	Utilización
ALUTs Combinacionales	777	182,400	< 1%
Memoria de ALUTs	0	91,200	0%
Total de registros	569	182,400	< 1%
Total de pines	192	888	22%
Total bloques de bits de memoria	4,883	14,625,792	< 1%
Bloques de DSP elementos de 18-bits	92	1,288	7%

Se puede observar como la imagen del *Cohete* es de una resolución muy pequeña y casi no se nota la diferencia entre encriptar con señal caótica con 2 enrollamientos y la señal caótica de 6 enrollamientos, para notar la diferencia se utilizó una imagen con mayor resolución, se transmitió una imagen de 480x640 bits en tono de grises. Los resultados de la co-simulación de Active-Simulink y la comparación de las imágenes en el proceso de transmisión, utilizando la señal caótica de dos enrollamientos se muestran en las figuras 5.26, 5.27 respectivamente, en la tabla 5.3 se muestra su respectiva tabla de recursos. De la misma manera en las figuras 5.27 y 5.28 se muestran los resultados de la transmisión utilizando señales caóticas de 6 enrollamientos, mostrando también sus recursos utilizados en la tabla 5.4.

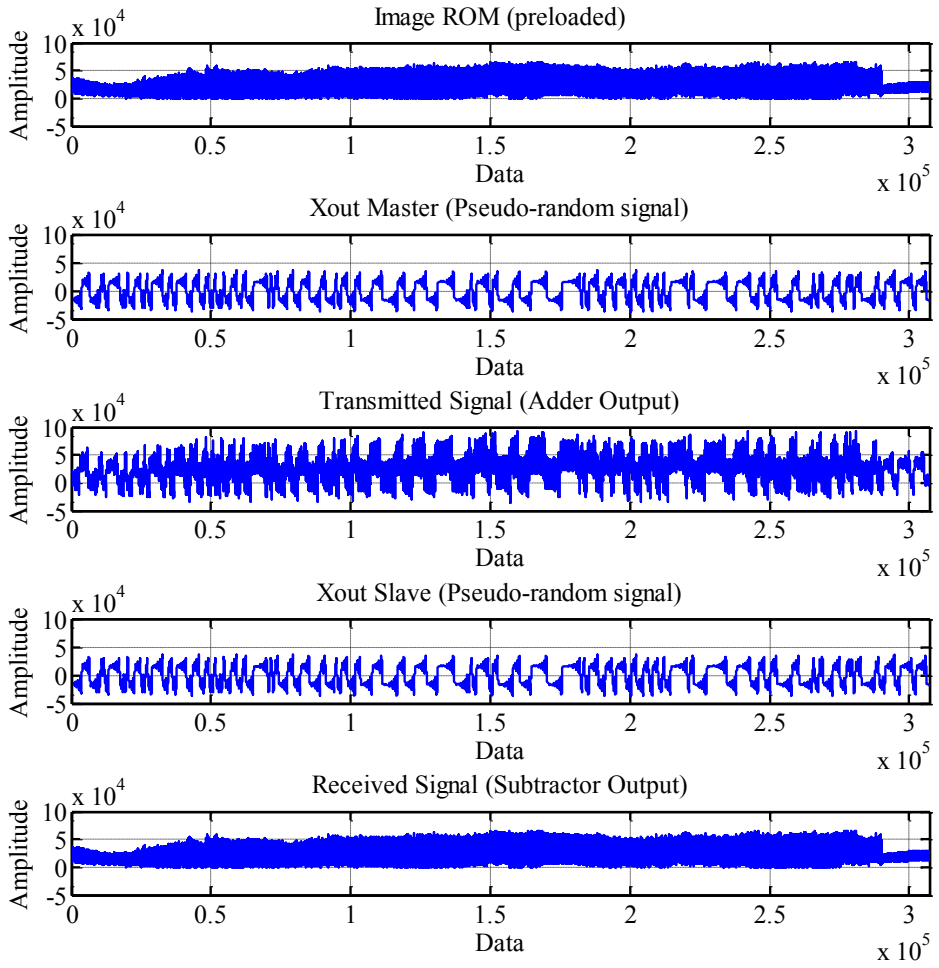

Figura 5.26. Co-simulación de transmisión de imagen 480x640 bits utilizando señales caóticas de 2 enrollamientos.

Figura 5.27. Comparación de Imagen 480x640 bits en proceso de transmisión utilizando señales caóticas de 2 enrollamientos.

Tabla 5.3. Reporte final de recursos en transmisión de imagen de 480x640 bits utilizando señales caóticas de 2 enrollamientos.

Recursos	Usados	Disponibles	Utilización
ALUTs Combinacionales	49,005	182,400	27%
Memoria de ALUTs	0	91,200	0%
Total de registros	611	182,400	< 1%
Total de pines	192	888	22%
Total bloques de bits de memoria	5,836,819	14,625,792	40%
Bloques de DSP elementos de 18-bits	64	1,288	5%

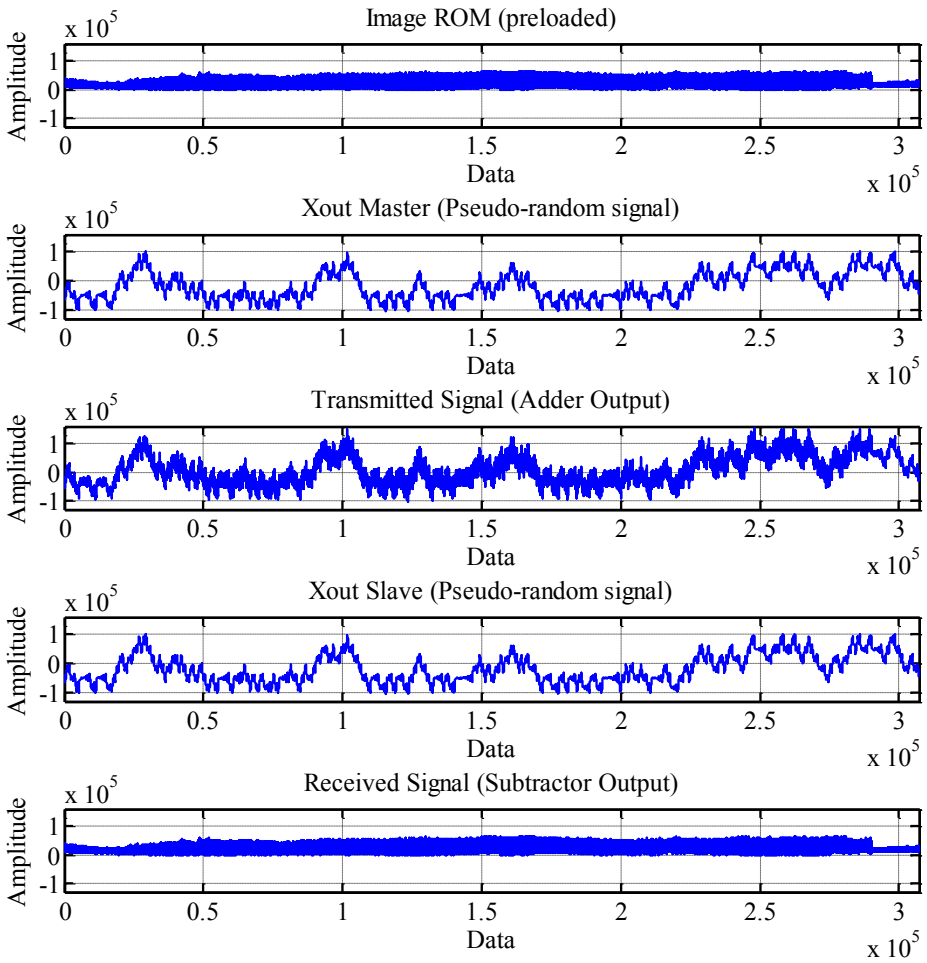

Figura 5.28. Co-simulación de transmisión de imagen 480x640 bits utilizando señales caóticas de 6 enrollamientos.

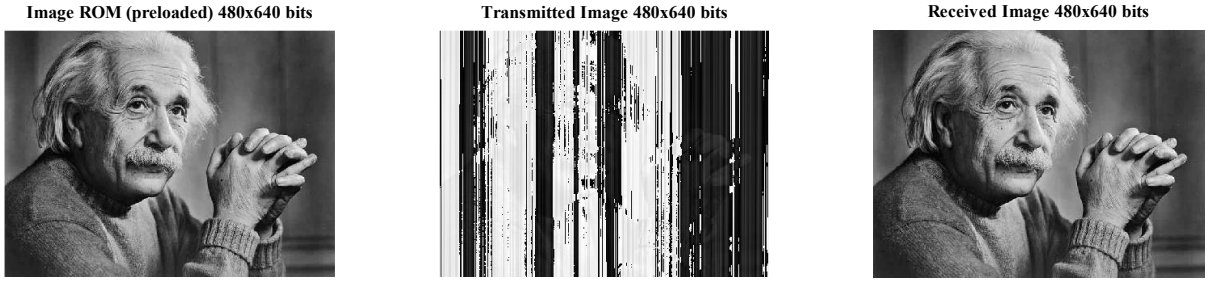

Figura 5.29. Comparación de Imagen 480x640 bits en proceso de transmisión utilizando señales caóticas de 6 enrollamientos.

Tabla 5.4. Reporte final de recursos en transmisión de imagen de 480x640 bits utilizando señales caóticas de 6 enrollamientos.

Recursos	Usados	Disponibles	Utilización
ALUTs Combinacionales	49,251	182,400	27%
Memoria de ALUTs	0	91,200	0%
Total de registros	631	182,400	< 1%
Total de pines	192	888	22%
Total bloques de bits de memoria	5,836,819	14,625,792	40%
Bloques de DSP elementos de 18-bits	92	1,288	7%

En las figuras 5.27 y 5.29 se puede visualizar gráficamente la diferencia entre usar dos y seis enrollamientos, cuando se utilizan dos enrollamientos en la imagen transmitida se alcanza a distinguir la forma de la imagen original, por otra parte al utilizar seis enrollamientos la imagen transmitida se ve más robusta se distingue menos la forma de la imagen original.

5.4.1 Análisis de correlación

Se evaluó que tan viable es la encriptación usada para la transmisión, para eso se realizó un análisis de correlación lineal. Un buen criptosistema debe de tener una correlación nula entre la imagen enviada y la transmitida, por otra parte debe de tener una alta correlación entre la imagen enviada y recibida ya que se supone que se debe recibir la imagen sin error alguno. El coeficiente de correlación puede ser medido entre "-1" y "1", donde "0" significa que tiene una correlación nula (los vectores de las imágenes son totalmente ortogonales), "1" que la correlación es tanto más fuerte cuando se aproxime a "1" (los vectores de las imágenes son colineales o paralelos) y "-1" que la correlación es tanto más fuerte cuando se aproxime a "-1" (los vectores de las imágenes son colineales de dirección opuesta) [53]. El coeficiente de correlación para matrices en 2-D se obtiene mediante el comando "*corr2(A,B)*" de Matlab, donde A y B son matrices o vectores del mismo tamaño. En la ecuación (75) se muestra la ecuación para obtener el coeficiente de correlación "*r*"

que computa el comando. En la tabla 5.5 se muestran las correlaciones entre las dos imágenes en el proceso de transmisión para el caso de 2 enrollamientos y 6 enrollamientos.

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \quad (75)$$

Tabla 5.5. Tabla de correlación de imágenes en el proceso de transmisión.

Imagen	Comparación	Núm. de enrollamientos	Correlación
Cohete.jpg (16x16 bits)	Imagen ROM vs Imagen Transmitida	2	0.9971
		6	0.9760
	Imagen ROM vs Imagen Recibida	2	1
		6	1
Einstein.jpg (480x640 bits)	Imagen ROM vs Imagen Transmitida	2	-0.3334
		6	-0.0314
	Imagen ROM vs Imagen recibida	2	1
		6	1

En la tabla 5.5 se puede observar que la correlación que más se acercó a 0 fue cuando se utilizaron las señales caóticas con 6 enrollamientos, para la imagen de "*cohete.jpg*" es muy poca la diferencia debido a que su resolución es muy pequeña, pero para la imagen de "*Einstein.jpg*", que tiene mayor resolución, se puede observar claramente que cuando se utilizaron 6 enrollamientos, la correlación se acerca más a "0". Lo cual corrobora que entre más enrollamientos tenga la señal caótica, y al encriptarla con una señal limpia, la diferencia entre la imagen limpia y encriptada es mayor.

5.5 CONCLUSIÓN

En este capítulo se presentó el desarrollo de la implementación en diseño para un FPGA de la sincronización de dos osciladores caóticos en un sistema maestro-esclavo mediante formas Hamiltonianananas, también se mostró una compacta aplicación que demuestra mediante la co-simulación entre Active-Matlab que los resultados obtenidos, tanto gráficamente como analíticamente que la sincronización de dos sistemas caóticos se puede utilizar para encriptar imágenes mediante la propiedad de aditividad, y así poder ser transmitida hacia el receptor o receptores. Se demostró que al utilizar señales caóticas de seis enrollamientos se mejora notablemente la robustez del sistema ya que es cuando la correlación entre la imagen limpia y encriptada tiende más a "0".

5.6 PRODUCTIVIDAD ACADÉMICA Y CIENTÍFICA

Este trabajo de tesis permitió la publicación de un artículo en una conferencia internacional, un artículo en una revista internacional, y el sometimiento de otro artículo en otra revista internacional:

1. P. J. Obeso-Rodelo, J. R. Cardenas, J. A. Galaviz-Aguilar, J. A. Sillas, J. M. Jiménez, J. C. Nuñez-Perez and E. Tlelo-Cuautle, "Metodología de diseño e implementación de un oscilador caótico de Chua en FPGA", in *Congreso Internacional en Ingeniería Electrónica*, vol. 36, pp. 312-317, Octubre 2014.

2. E. Tlelo-Cuautle, J. J. Rangel-Magdaleno, A. D. Pano-Azucena, P. J. Obeso-Rodelo and J. C. Nunez Perez, "FPGA realization of multi-scroll chaotic oscillators", *Comunications in Nonlinear Science and Numerical Simulation*, vol. 27, no. 1, pp. 66-80, Octubre 2015.

3. E. Tlelo-Cuautle, V. H. Carbajal-Gomez, P. J. Obeso-Rodelo, J. J. Rangel-Maldonado and J. C. Núñez-Pérez, "FPGA Realization of a Chaotic Communication System Applied to Image Processing", *Nonlinear Dynamics*, (artículo sometido a revisión, 21 de Abril del 2015).

Capítulo 6

Conclusiones y Trabajos futuros

6.1 CONCLUSIÓN GENERALES

En este trabajo de tesis se llevó a cabo un estudio bibliográfico del estado del arte sobre osciladores caóticos, por eso la elección de dos de los sistemas caóticos más sencillos de implementar, se presentó de manera general una introducción a los métodos numéricos, y una breve explicación de los métodos de Euler y Runge-Kutta que fueron los utilizados para resolver los sistemas caóticos en esta tesis, también se mostraron algunas aplicaciones de los métodos numéricos en general, enfocadas a la solución de ecuaciones diferenciales. Se hizo referencia a trabajos importantes donde se utiliza la sincronización de sistemas caóticos para aplicaciones específicas, toda esta teoría fue la base para el desarrollo de esta tesis, y sus principales aportaciones.

Además se mostró el diseño de multi-enrollamientos de los osciladores caóticos basados en SNLF y el circuito de Chua, se mostraron los resultados de las simulaciones en el software matemático MATLAB, resolviendo ambos sistemas mediante el método de Euler y Runge-Kutta de orden 4, a simple vista no se notó la diferencia de utilizar el método de Euler y Runge-Kutta, sin embargo se sabe que el método de Runge-Kutta es más preciso que el método de Euler, aunque sea más complejo de implementar. También se mostró una forma alternativa de simular el oscilador caótico basado en el circuito de Chua utilizando la herramienta Simulink.

Asimismo se presentaron las características de la tarjeta FPGA a utilizar y su tarjeta de adquisición de datos HSMC, se llevó cabo las descripción de la metodología empleada para realizar la implementación en el dispositivo FPGA-DSP Cyclone III Edition de Altera del oscilador caótico de Chua mediante la herramienta DSP-builder y Simulink. Además se mostró otra metodología para la implementación en FPGA para los osciladores caóticos basados en SNLF y el circuito de Chua a través del diseño de arquitecturas mediante código VHDL, ambos tipos de osciladores resueltos mediante el método de Euler y Runge-Kutta, se mostraron los resultados en co-simulación y experimentales, los cuales fueron satisfactorios ya que fueron similares a los obtenidos en las simulaciones de MATLAB, estas señales son factibles para aplicaciones para la seguridad de sistemas de comunicación, ya que con estos diseños se puede llevar al Hardware en forma de chip, y sustituir otros osciladores caóticos ya implementados de manera analógica, lo cual mejorará la portabilidad y el costo en Hardware.

Igualmente se presentó el desarrollo de la implementación en diseño mediante código VHDL para un FPGA de la sincronización de dos osciladores caóticos en un sistema maestro-esclavo a través de formas Hamiltonianas y observadores, también se mostró una compacta aplicación que demostró mediante la co-simulación entre Active-Matlab que las señales generadas de dos osciladores caóticos, siendo estas sincronizadas, se pueden utilizar para encriptar imágenes mediante la propiedad de aditividad, y así poder ser transmitida hacia el receptor o receptores. Se demostró que al utilizar señales caóticas de seis enrollamientos se mejora notablemente la robustez del sistema ya que es cuando la correlación entre la imagen limpia y encriptada tiende más a "0".

En el presente trabajo se ha diseñado una metodología para el diseño e implementación de dos tipos de osciladores caóticos en una tarjeta FPGA a través de métodos numéricos, con dicha metodología se han obtenido resultados aceptables los cuales fueron clave para el desarrollo de una aplicación a nivel de co-simulación y lo que conlleva a que se empiecen a desarrollar más aplicaciones, o simplemente a implementar otros sistemas caóticos en tarjetas FPGA.

6.2 COMENTARIOS Y RECOMENDACIONES PARA TRABAJOS FUTUROS

Debido a los excelentes resultados obtenidos tanto en simulación como en la implementación, en este trabajo de tesis surgen las siguientes recomendaciones como trabajos futuros:

- Aplicación de los diseños de dichos sistemas en esquemas criptográficos más complejos en donde se exija mayor robustez del sistema.
- Aumento de frecuencia y optimización de los sistemas caóticos diseñados para llevarlos a nivel de microondas.
- Comparación de resultados utilizando distintos dispositivos FPGA.
- Diseño e implementación de otros sistemas caóticos utilizando la misma metodología presentada, realizando comparación de resultados para probar que sistema es más robusto.
- Incorporación de los diseños implementados en un sistema de transmisión completo, acaparando desde el transmisor hasta el receptor.
- Probar la robustez de los sistemas caóticos diseñados en un nuevo esquema de comunicación caótica llamado CDSK.

Referencias Bibliográficas

[1] E. N. Lorenz, "Deterministic Nonperiodic Flow", *Journal of the Atmospheric sciences*, vol. 20, pp. 130-141, Marzo 1963.

[2] A. Ali-Pacha, N. Hadj-Said, A. M'Hamed, and A. Belghoraf, "Lorenz's Attractor Applied to the Stream Cipher (Ali-Pacha Generator)," *Chaos, Solitons & Fractals*, vol. 33, pp. 1762-1766, Agosto 2007.

[3] A. Ali-Pacha, N. Hadj-Said, B. Belmeki, and A. Belgoraf, "Chaotic behavior for the secrete key of cryptographic system," *Chaos, Solitons & Fractals*, vol. 48, pp. 1549-1552, Marzo 2007.

[4] R. Bernardini and G. Cortelazzo, "Tools for Designing Chaotic Systems for Secure Random Number Generation," *Fundamental Theory and Applications*, vol. 48, pp. 552-564, Mayo 2001.

[5] J. M. Amigó, J. Szczepanski, and L. Kocarev, "Theory and Practice of Chaotic", *Physics Letters A*, vol. 366, pp. 211-216, Febrero 2007.

[6] M. Yaobin, C. Liu, and L. Wenbo, "Design and FPGA Implementation of a Pseudo-Random Bit Secuence Generator Using Spatiotemporal Chaos," in *Proceedings of the International Conference on communications, circuits and systems*, vol. 3, pp. 2114-2118, Guilin, China, Junio 2006.

[7] E. A. Ibáñez, "La Pedagogía del Caos: ¿una aplicación lícita de la teoría homónima," Universidad Católica de Santa Fe, Argentina.

[8] L. Kocarev, K. S. Halle, K. Eckert, L. O. Chua and U. Parlitz, "Experimental demonstration of secure communications via chaotic synchronization", *International Journal of Bifurcation and Chaos*, vol. 2, no. 3, pp. 709-713, Septiembre 1992.

[9] S. Banerjee and J. Kurths, "Chaos and Cryptography: A new dimension in secure communications", *The European Physical Journal Special Topics*, vol. 223, no. 8, pp. 1441-1445, Junio 2014.

[10] G. Alvarez and S. Li, "Some Basic Cryptographic requirements for Chaos-based Cryptosystems", International Journal of Bifurcation and Chaos, vol. 16, no. 8, pp. 2129-2151, Agosto 2006

[11] J. S. Bay, *Fundamentals of Linear State Space Systems*, 1ª. Ed. McGraw-Hill Series in Electrical Engineering, 1998.

[12] T. Kapitaniak, *Chaotic Oscillators: Theory and applications*. 2ª. Ed. Singapure: World Scientific Publishing, 1992.

[13] F. J. Rivera, E. Tepichín and C. G. Treviño, "Transmisor inalámbrico de audio y video", in 18º Congreso Nacional de Instrumentación, 2002.

[14] J. Jensen, "An improved square-wave oscillator circuit", in *Solid-State Circuits Conference. Digest of Technical Papers. 1957 IEEE International*, vol. 0, pp. 25, 25, Febrero 1957.

[15] S. Wolf and R. F. M. Smith, *Guía para Mediciones Electrónicas y Prácticas de Laboratorio*, 2ª. Ed. México: Prentice-Hall Hispanoamericana, 1992.

[16] L. E. Avendaño, *Sistemas Electrónicos Analógicos, un Enfoque Matricial*, 1ª. Ed. Colombia: Universidad Tecnologica de Pereira, 2006.

[17] J. J. Brophy, *Basic electronics for scientists*, 3ª. Ed. New York, USA: McGraw-Hill Inc., 1977.

[18] A. Cornejo, Biblioteca Virtual eumed.net,"Complejidad y Caos: Guía para la administración del siglo XXI", Abril 2015. [En línea]. Disponible: http://www.eumed.net/cursecon/libreria/2004/aca/aca.html

[19] I. Gomes da Silva, "Aspectos de Sincronización en un sistema caótico", Tesis doctoral, Universitat de Les Illes Balears, Mallorca, Noviembre 2006.

[20] C. Ramírez, "Diseño de una celda lineal a tramos en tecnología CMOS basada en transistores de compuerta cuasi-flotante", Tesis de Maestría en CITEDI-IPN, Tijuana, México, Enero 2014.

[21] J. Lü and G. Chen, "Generating multiscroll chaotic attractors: theories, methods and applications", *International Journal of Bifurcation and Chaos*, vol. 16, no. 4, pp. 775-858, 2006.

[22] E. Tlelo-Cuautle, M. A. Duarte, and J. M. García, "Modalado y Simulación de un Oscilador Caótico usando MatLab," *IEEE Latin America Transactions*, vol. 4, no. 2, pp. 95-98, Mayo 2007.

[23] S. C. Chapra and R. P. Canale, *Métodos numéricos para ingenieros*, 5ª. Ed. México, D. F.: McGraw-Hill Interamericana, 2007.

[24] A. Ochoche, "Almost Runge-Kutta methods of orders up to five", *WSEAS Transactions on Mathematics*, vol. 10, pp. 159-168, Mayo 2011.

[25] E. H. C. Julyan and O. Piro, "The dynamics of Runge-Kutta methods", *International Journal of Bifurcation and Chaos*, vol. 2, pp. 1-8, 1992.

[26] A. Ochoche, "Improving the modified Euler Method for Better Performance on Autonomous Initial Value Problems", *Leonardo Journal of Sciences*, no. 12, pp. 57-66, Junio 2008.

[27] E. Tlelo, A. D. Pano, V. H. Carbajal, and M. Sanchez, "Experimental Realization of a Multiscroll Chaotic Oscillator with Optimal Maximum Lyapunov Exponent", *The Scientific World Journal*, vol. 2014, pp. 1-16, Abril 2014.

[28] L. de la Fraga and E. Tlelo, "Optimizing the maximum Lyapunov exponent and phase space portraits in multi-scroll chaotic oscillators", *Nonlinear Dynamics*, vol. 76, no. 2, pp. 1503-1515, Enero 2014.

[29] I. I. Blechman, "The generalized self-balancing principle", *Asia-Pacific Vibration Conference*, Kitakyshu, Japan, pp. 509-514, 1993

[30] I. I. Blechman, "Self-synchronization of mechanical vibroexciters: control of vibrations, generalized rotors, self-balancing principle", *The active control vibration, IUTAM Symposium*, London, Inglaterra, pp. 169-173, 1994.

[31] I. I. Blechman, P. S. Landa and M. G. Rozenblum, "Synchronization and chaotization in interacting dynamical systems", *Applied Mechanical Reviews*, vol. 48, no. 11, pp. 733-752, Noviembre 1995.

[32] H. Nijmeijer, I. I. Blechman, A. L. Fradkov, and A. Y. Pogromsky, "Self-synchronization and controlled synchronization", *Control of Oscillations and Chaos*, vol. 1, pp. 36-41, Agosto 1997.

[33] L. M. Pecora, T. L. Carroll, G. A. Johnson, D. J. Mar and J. F. Heagy, "Fundamentals of Synchronization in Chaotic Systems, Concepts, and Applications", *Chaos*, vol. 7, no. 4, pp. 520-543, Diciembre 1997.

[34] L. M. Pecora and T. L. Carroll, "Synchronization in Chaotic Systems", *Physical Reviews Letters*, vol. 64, no. 8, pp. 821-824, Febrero 1990.

[35] L. M. Pecora and T. L. Carroll, "Synchronizing Chaotic Circuits", *IEEE Transactions Circuits and Systems*, vol. 38, no. 4, pp. 453-456, Abril 1991.

[36] M. Wada, Y. Nishio and A. Ushida, "Chaotic itinerancy phenomena on coupled n-double scrolls chaotic circuits", *Proceedings of the IEEE International Symposium on Circuits and Systems, 1999. ISCAS '99.* , vol. 5, pp.487-490, June 1999.

[37] C. E. Jiménez Acosta, "Diseño de un sistema de comunicaciones seguras por medio de la sincronización de señales caóticas", Tesis de Maestría en CITEDI-IPN, Tijuana, México, Octubre 2005.

[38] J. D. Reyes de la Cruz, "Acotación de conjuntos compactos invariantes y la solución de problemas de sincronización y diseño de controladores para algunos sistemas caóticos", Tesis de Maestría en CITEDI-IPN, Tijuana, México, Diciembre 2009.

[39] S. Boccaletti, J. Kurths, G. Osipov, D. L. Valladares and C. S. Zhou, "The synchronization of chaotic systems", *Physics reports*, vol. 366, no. 1, pp. 1-101, Enero 2002

[40] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D-U Hwang, "Complex networks: Structure and dynamics", *Physics reports*, vol. 424, no. 4, pp. 175-308, Enero 2006

[41] S. Boccaletti, C. Grebogi, Y-C Lai, H. Mancini and D. Maza, "The control of chaos: theory and applications", *Physics reports*, vol. 329, no. 3, pp. 103-197, Marzo 2000.

[42] L. Kocarev and U. Parlitz, "General approach for chaotic synchronization with applications to communication", *Physical Review Letters*, vol. 74, no. 25, pp. 5028-5031, Junio 1995.

[43] L. Kocarev and U. Parlitz, "Generalized synchronization, predictability, and equivalence of unidirectionally coupled dynamical systems", *Physical Review Letters*, vol. 76, no. 11, pp. 1816-1819, Marzo 1996.

[44] K. M. Cuomo and A. V. Oppenheim, "Circuit Implementation of Synchronized Chaos with Applications to Communications", *Physical Review Letters*, vol. 71, no. 1, pp. 65-68, Julio 1993.

[45] H. Nijmeijer and M. Y. Mereels, "An observer looks at synchronization", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol.44, no.10, pp.882-890, October 1997.

[46] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography*. 5ª Ed. New York, USA: CRC Press, 1996.

[47] H. Sira-Ramirez and C. Cruz-Hernandez, "Synchronization of chaotic systems: a generalized Hamiltonian systems approach", *International Journal of Bifurcation and Chaos*, vol. 11, pp. 1381-1395, May 2001.

[48] J.M. Muñoz-Pacheco, "Synchronization of PWL function-based 2D and 3D multi scroll chaotic systems," *Nonlinear Dynamics*, vol. 70, no. 2, pp. 1633-1643, 2012.

[49] R. Trejo-Guerra, E. Tlelo-Cuautle, C. Sánchez-López, J.M. Muñoz-Pacheco and C. Cruz Hernandez, "Realization of multiscroll chaotic attractors by using current-feedback operational amplifiers", *Revista Mexicana de Física*, vol. 56, no. 4, pp. 268-274, August 2010.

[50] V. H. Carbajal-Gomez, E. Tlelo-Cuautle, R. Trejo-Guerra and J.M. Muñoz-Pacheco, "Simulating the synchronization of multi-scroll chaotic oscillators," *IEEE International Symposium on Circuits and Systems (ISCAS), 2013*, pp.1773-1776, 19-23 May 2013.

[51] G. Conde and A. Ramirez, "Estudio de dos circuitos caóticos", *Revista boliviana de Física*, vol. 13, no. 13, pp. 58-74, 2007.

[52] Altera, "DSP Builder Handbook: Introduction to DSP Builder". [En línea]. Fecha de consulta: Abril 2015. Disponible: http://www.altera.com/literature/hb/dspb/hb_dspb_intro.pdf

[53] M.A. Murillo, C. Cruz, F. Abundiz, R. M. López, O. R. Acosta Del Campo, "A RGB image encryption algorithm based on total plain image characteristics and chaos", *Signal Processing*, vol. 109, pp. 119-131, Abril 2015.