ENCRIPTADOR DE SEÑAL MEDIANTE OSCILADORES CAÓTICOS IMPLEMENTADOS EN FPGA

Por:

Ing. Dulce Karolina Vázquez Morales

Tesis sometida como requisito parcial
para obtener el grado de

MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE ELECTRÓNICA

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE).

Octubre de 2016
Santa María Tonantzintla, Puebla

Dirigida por:

Dr. José de Jesús Rangel Magdaleno
Investigador Título
Departamento de Electrónica
INAOE

Encriptador de señal mediante osciladores caóticos implementados en FPGA

Tesis de Maestría

Por:

Ing. Dulce Karolina Vázquez Morales

Asesor:

Dr. José de Jesús Rangel Magdaleno

Instituto Nacional de Astrofísica Óptica y Electrónica

Coordinación de Electrónica

Tonantzintla, Puebla. Octubre 2016

Agradecimientos

A Dios por darme la fuerza en los momentos difíciles.

A mis padres, Isabel e Hilario, por el apoyo durante mi trayectoria académica y por el infinito amor que me brindan.

A mis hermanos, Karen y José Antonio, por las muestras de cariño, apoyo y motivación para seguir adelante.

A mis sobrinos, Luna y Gael, por la alegría, amor y motivación que representan en mí.

A mi amado Daniel por la paciencia, apoyo, amor, comprensión y motivación que me brinda día a día para alcanzar nuevas metas.

A mi asesor el Dr. José de Jesús Rangel Magadaleno por su apoyo, enseñanzas y dedicación en la realización de este proyecto.

Al Consejo Nacional de Ciencia y Tecnología, por la beca otorgada para la realización de mis estudios.

A mis sinodales, Dr. Esteban Tlelo Cuautle, Dr. Juan Manuel Ramírez Cortés y Dr. Israel Cruz Vega. Por el tiempo dedicado a la revisión de esta tesis.

Dedicatoria

Con todo mi amor y cariño para las personas que motivan mi vida y me apoyan incondicionalmente, a mi familia, a mi amado Daniel y a mi pequeña Luna.

Resumen

Actualmente, los sistemas caóticos son comúnmente utilizados en el campo de la ingeniería. Una de las principales estructuras utilizadas es el generador de señales basadas en señales caóticas. Estos generadores tienen un rol importante en el área de comunicación y criptología ya que tienen las características de ser procesos evolutivos complicados y poseer alta aleatoriedad.

En el presente trabajo se desarrolla e implementa una comunicación segura y confiable entre dos o más dispositivos, por medio del uso de una señal caótica del tipo sistema Lorenz como codificador. La codificación se lleva a cabo mediante el uso de FPGAs, los cuales generan oscilaciones caóticas sincronizadas. Dichas oscilaciones se utilizan para enmascarar los datos, encriptándolos. La comunicación entre estos dispositivos se logra conectando módulos inalámbricos XBee a cada uno de ellos. En dicha comunicación, se trasmite un archivo desde una PC a otra usando el protocolo RS232 hacia los FPGAs. Finalmente se realizaron pruebas enviando distintos archivos a los cuales se les realizó un análisis de correlación, tanto al archivo recibido como al canal encriptado, para asegurar la integridad de los datos y su nivel de encriptación.

Abstract

Nowadays, chaotic systems are commonly used in the engineering field. One of the main structures used is the chaos based signal generator. This generators have a very important role in communications and cryptology because they have the characteristic of being evolving processes and having high randomness.

In this work a safe and reliable communication between two or more points is development and implemented, using a Lorenz atractor-type chaotic signal as codifier. This codification is made using FPGAs, which generate synchronized chaotic oscillations, which are used to mask the data for their encryption. The communication between these devices is achieved by connecting them to wireless modules called XBees. In said communication, a file of any kind is sent from one computer to another using the RS232 protocol to the FPGAs. Finally, tests were made by sending different kinds of files and doing correlation tests to the received file as well as the encrypted channel, to ensure the data integrity and its encryption.

Tabla de Contenido

Agradecimientos i

Dedicatoria iii

Resumen v

Abstract vii

Lista de Figuras xi

Lista de Tablas xiii

1. Encriptadores de señal basados en osciladores caóticos **1**

1.1. Introducción 1

1.2. Justificación 3

1.3. Objetivos 3

1.3.1. Objetivo general 3

1.3.2. Objetivos específicos 4

2. Marco Teórico **5**

2.1. Tipos de osciladores caóticos 5

2.1.1. Oscilador Caótico de Lorenz 5

2.1.2. Oscilador Caótico de Rössler 6

2.1.3. Oscilador Caótico de Chua 7

2.2. Sincronización de osciladores caóticos 9

2.3. Métodos de integración 10

2.4. Interfaz y funcionamiento de los módulos inalámbricos Xbee 11

3. Metodología **15**
3.1. Diseño de los observadores 15
3.2. Protocolo de transmisión 17
3.3. Simulación en LabVIEW 21
3.4. Realización en FPGA 22
3.4.1. Diagrama a bloques 23
3.4.2. Máquina de estados 25
3.4.3. Simulaciones 33
3.4.4. Recursos utilizados 34
3.5. Realización en la computadora 34

4. Pruebas experimentales y resultado **41**
4.1. Prueba de integridad y encriptación 41
4.2. Prueba de alcance y ruido 44
4.3. Medición de las señales en Osciloscopio 44
4.4. Resultado 45

5. Conclusiones **49**

6. Trabajo a futuro **51**

Bibliografía **53**

Lista de Figuras

2.1. Atractor de Lorenz. 5
2.2. Atractor de Rössler. 7
2.3. Circuito electrónico de Chua. 7
2.4. Atractor de Chua. 8
2.5. Conexiones mínimas requeridas para el Xbee. 12
2.6. Modos de operación del módulo Xbee. 13
2.7. Ejemplo Comando AT. 13
2.8. Ejemplo direccionamiento 16 bits. 14

3.1. Gráfica de número de iteraciones para sincronizar dos osciladores variando el término observador. 16
3.2. Gráficas de sincronización. 17
3.3. Gráficas de error para distintos valores del vector K. 18
3.4. Grafica de los observadores X, Y y Z. 19
3.5. Panel frontal de la simulación de los dos osciladores caóticos. 21
3.6. Panel frontal de la simulación de los osciladores caóticos sincronizados. 22
3.7. Diagrama a bloques de las ecuaciones del sistema Lorenz. 24
3.8. Diagrama a bloques de las ecuaciones del sistema Lorenz con observadores. 24
3.9. Diagrama a bloques del FPGA transmisor. 25
3.10. Diagrama a bloques del FPGA receptor. 26
3.11. Estados 0, 1 y 2 de la máquina de estados del FPGA transmisor. . . . 27
3.12. Estados 3, 4 y 5 de la máquina de estados del FPGA transmisor. . . . 27
3.13. Estados 6 y 7 de la máquina de estados del FPGA transmisor. 28
3.14. Estados 12 y 13 de la máquina de estados del FPGA transmisor. . . . 28
3.15. Estados 52 y 53 de la máquina de estados del FPGA transmisor. . . . 28

3.16. Estados 54, 55 y 56 de la máquina de estados del FPGA transmisor. . 29
3.17. Estados 57, 58 y 59 de la máquina de estados del FPGA transmisor. . 29
3.18. Estados 60, 61 y 62 de la máquina de estados del FPGA transmisor. . 29
3.19. Estados 0, 1, 2 y 3 de la máquina de estados del FPGA receptor. . . . 31
3.20. Estados 4, 5 y 6 de la máquina de estados del FPGA receptor. 31
3.21. Estados 7 y 8 de la máquina de estados del FPGA receptor. 31
3.22. Estados 9, 10 y 11 de la máquina de estados del FPGA receptor. . . . 32
3.23. Estados 12, 13 y 14 de la máquina de estados del FPGA receptor. . . 32
3.24. Estados 15, 16, 17 y 18 de la máquina de estados del FPGA receptor. 32
3.25. Estados 19 y 20 de la máquina de estados del FPGA receptor. 33
3.26. Simulación en Active de los osciladores maestro y esclavo. 33
3.27. Simulación en Active de la sincronización de los osciladores maestro y esclavo activada por la bandera Sync. 34
3.28. Interfaz de usuario para transmitir. 35
3.29. Diagrama a bloques de la interfaz de usuario para la transmisión. . . 37
3.30. Interfaz de usuario para la recepción. 38
3.31. Diagrama a bloques de la interfaz de usuario para la recepción. 39

4.1. Comparación entre la imagen *Lenna_gray.gif* (256x256) transmitida, recibida y el canal. 43
4.2. Comparación entre la imagen *Lenna.bmp* (256x256) transmitida, recibida y el canal. 43
4.3. Comparación entre la imagen *Lenna.jpg* (512x512) transmitida, recibida y el canal. 43
4.4. Mediciones realizadas con el osciloscopio. 46
4.5. Fotografía del sistema completo. 47

Lista de Tablas

2.1. Recursos utilizados por los métodos de integración para calcular la señal caótica. 11

3.1. Entradas y salidas de la máquina de estados del FPGA transmisor. . 26
3.2. Entradas y salidas de la máquina de estados del FPGA receptor. . . . 30
3.3. Recursos utilizados para transmitir señales caóticas utilizando el FPGA Cyclone II (EP2C35F672C6). 34

Capítulo 1

Encriptadores de señal basados en osciladores caóticos

1.1. Introducción

La teoría del caos es una rama de las matemáticas que trata ciertos tipos de sistemas complejos y dinámicos, los cuales son muy sensibles a las variaciones en las condiciones iniciales. Una mínima variación en dichas condiciones iniciales puede implicar grandes diferencias en el comportamiento futuro, imposibilitando la predicción a largo plazo. Esto sucede aunque estos sistemas son en rigor determinísticos, es decir, su comportamiento puede ser completamente determinado conociendo sus condiciones iniciales.

Los sistemas dinámicos son clasificados por su comportamiento en tres tipos: estables, inestables y caóticos. Los sistemas estables tienen la característica de que cuando dos soluciones con condiciones iniciales suficientemente cercanas siguen siendo cercanas a lo largo del tiempo. Así, un sistema estable tiende a un punto u órbita, llamados atractores o sumideros. A diferencia de los sistemas estables, dos soluciones con condiciones iniciales diferentes en un sistema inestable acaban divergiendo por pequeñas que sean estas diferencias, así un sistema inestable se aleja de los atractores. Cuando el sistema no es inestable y si bien dos soluciones se mantienen a una distancia finita cercana a un atractor del sistema dinámico, se dice que el sistema es caótico. Las soluciones se mueven en torno al atractor de manera irregular y pasado el tiempo ambas soluciones no son cercanas pero suelen ser cualitativamente similares. Por lo tanto, el sistema permanece confinado en una

zona de su espacio de estados, pero sin tender a un atractor fijo [1].

La criptografía se encarga del estudio de los algoritmos, protocolos y sistemas que se utilizan para dotar de seguridad a las comunicaciones, a la información y a las entidades que se comunican [2]. Es una herramienta muy útil cuando se desea tener seguridad informática; puede ser también un medio para garantizar las propiedades de confidencialidad, integridad y disponibilidad de la información. La confidencialidad se trata de dar acceso a la información a un personal autorizado, a través de códigos y técnicas de cifrado. La integridad es garantizar la completitud y corrección de la información.

La encriptación basada en caos ha sido sugerida como una nueva y eficiente forma de tratar con el problema de aplicaciones y comunicaciones embebidas rápidas y de alta seguridad. Por ejemplo, varios trabajos han sido propuestos como el enmascaramiento caótico [3], modulación de conmutación caótica [4], modulación por retroalimentación dinámica de la señal de información [5] y otros. Generalmente, la estrategia de estas técnicas consiste en implementar un generador de llaves de cifrado basado en un generador de datos caóticos para encriptar un texto simple.

La implementación digital de los sistemas caóticos presenta ciertas ventajas para aplicaciones embebidas, tal como la precisión, especialmente para encriptación de datos y comunicación segura entre sistemas embebidos. Recientemente, se han propuesto e implementado varios sistemas caóticos digitales en tecnología FPGA (Field Programmable Gate Array) [6]. Este hardware programable es más importante en el diseño de sistemas digitales debido al excelente poder computacional y flexibilidad de procesado que provee.

Los sistemas caóticos han recibido importante atención en los recientes años, porque tienen características más complicadas de evolución, alta aleatoriedad y un alto rango dinámico, y resultan ser muy útiles en sistemas de comunicación basados en caos.

1.2. Justificación

Desde la existencia de las redes de comunicación, principalmente el internet, se han creado nuevas posibilidades para el flujo de información. Así mismo han surgido amenazas a la seguridad de la información y es por esto que se necesitan nuevas y mejores formas de garantizar la confidencialidad y seguridad de la información electrónica. Por esta razón, en este trabajo se desarrolla e implementa un sistema de encriptación basado en osciladores caóticos, creando una red de comunicación segura entre varios dispositivos. Se ha demostrado en trabajos previos [7],[8] que debido a la naturaleza caótica de este tipo de osciladores, son muy eficaces en la creación de señales aparentemente aleatorias pero que son determinísticas, por lo tanto una de estas señales se utiliza como máscara para los datos obteniendo así un método de cifrado que utiliza pocos recursos en un sistema embebido pero genera una muy buena encriptación de los datos.

En anteriores trabajos, se han desarrollado sistemas de encriptación de señales analógicas, basados en componentes discretos y amplificadores operacionales [9], así mismo se ha desarrollado un encriptador basado en FPGA realizado en un solo dispositivo [10], entre otros. En el presente trabajo, la sincronización y transmisión de datos se lleva a cabo utilizando dos dispositivos físicamente separados, comunicados únicamente por un módulo externo de comunicación inalámbrica llamado Xbee [11].

1.3. Objetivos

1.3.1. Objetivo general

El objetivo de esta tesis es lograr una comunicación que garantice una alta confidencialidad de los datos, además de que se conserve la integridad de la información entre dos o más puntos, por medio del uso de una señal caótica como máscara para los datos, utilizado FPGAs conectados a módulos inalámbricos llamados Xbee. Cada FPGA debe contar con un oscilador caótico y previo a la transmisión de datos se debe alcanzar una rápida sincronización entre los osciladores para la subsecuente comunicación.

1.3.2. Objetivos específicos

- Implementar en VHDL la integración numérica de las ecuaciones diferenciales que componen el sistema caótico atractor de Lorenz.
- Diseñar una estructura digital que gobierne cada paso del protocolo de recepción, enmascaramiento/desenmascaramiento y transmisión de datos.
- Verificar, por medio del uso de la correlación entre las señales e inspección visual, el nivel de encriptación alcanzado por el sistema.
- Implementar en VHDL la red de comunicación inalámbrica haciendo uso de los módulos Xbee por medio de la correcta interfaz entre estos y los FPGAs.
- Programar una interfaz de usuario en el cual se seleccione un archivo y un destino para su transmisión por medio del sistema propuesto, y otra que administre la recepción de dicho archivo y su almacenamiento.
- Realizar pruebas de integridad de los datos transmitidos bajo condiciones de alta separación y alto ruido.

Capítulo 2

Marco Teórico

2.1. Tipos de osciladores caóticos

2.1.1. Oscilador Caótico de Lorenz

Edward Lorenz introduce en 1963 el concepto de atractor de Lorenz, el cual es un sistema dinámico determinista tridimensional no lineal derivado de las ecuaciones simplificadas de rollos de convección que se producen en las ecuaciones dinámicas de la atmósfera terrestre, como se muestra en la figura(2.1), Lorenz llegó a estas ecuaciones al intentar predecir los fenómenos meteorológicos [3, 12].

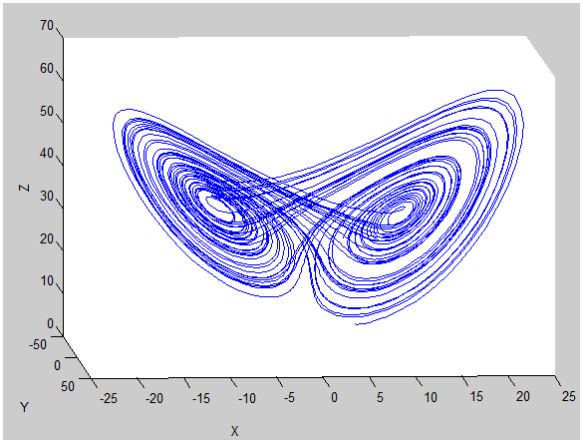


Figura 2.1: Atractor de Lorenz.

El atractor de Lorenz es un sistema de tercer orden dinámico el cual esta dado por:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= rx - xz - y \\ \dot{z} &= xy - bz \end{aligned} \tag{2.1.1}$$

Donde:

- x, y, z son las varibles de estado
- σ es el Número de Prandtl
- r es el Número de Rayleigh
- b es el tamaño de la región espacial apróximada por el sistema Lorenz

Lorenz probó que, para dado un conjunto de valores, este sistema exhibe una trayectoria altamente sensible a las conidiciones inciales, y estas propiedas se relacionan a soluciones caóticas. El conjunto más común de valores son $\sigma = 10$, r = 28, y b = 8/3 [12].

2.1.2. Oscilador Caótico de Rössler

En 1970 es introducido el sistema Rössler como ecuaciones prototipo con el mínimo de ingredientes para un caos en tiempo continuo. Rösler fue inspirado por la geometría de los flujos en tres dimensiones del sistema Lorenz para la creación de un nuevo sistema de ecuaciones caóticas más simples que generan un solo atractor pero conservan la propiedad de ser caóticas, figura (2.2). Estas ecuaciones a diferencia del sistema Lorenz, no poseen una interpretación física [13].

El sistema Rösler está dado por las siguientes ecuaciones:

$$\begin{aligned} \dot{x} &= (y + z) \\ \dot{y} &= x + ay \\ \dot{z} &= b + xz - cz \end{aligned} \tag{2.1.2}$$

Donde:

- x, y, z son las varibles de estado

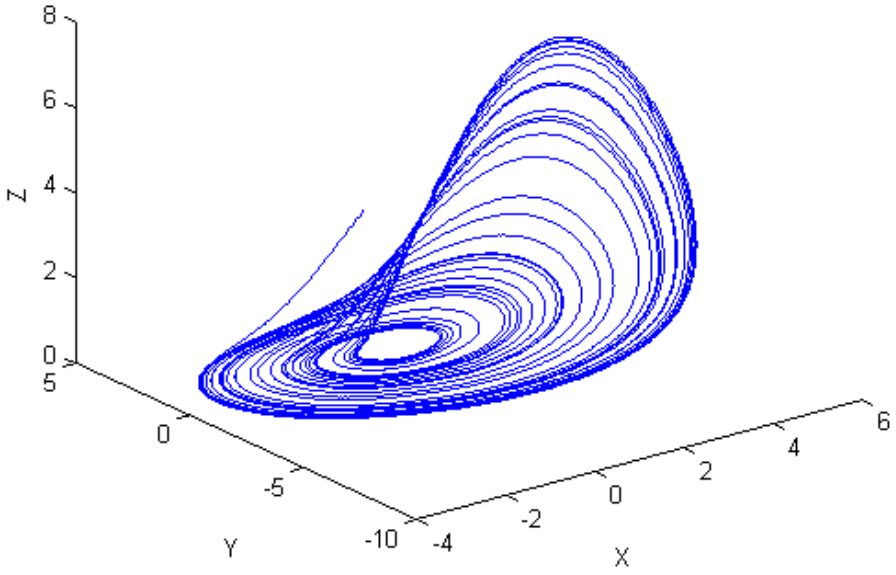


Figura 2.2: Atractor de Rössler.

- a, b, c son constantes

2.1.3. Oscilador Caótico de Chua

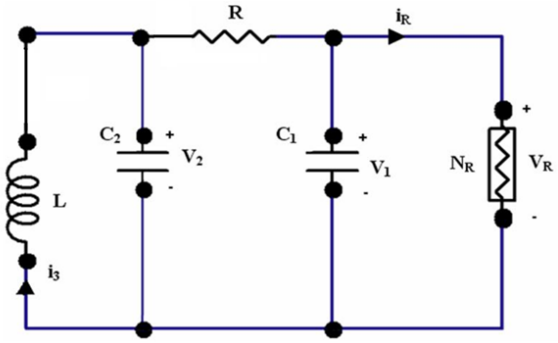


Figura 2.3: Circuito electrónico de Chua.[1]

El circuito de Chua, figura (2.3), inventado en 1983 por Leon O. Chua, es un circuito electrónico simple que exhibe un comportamiento caótico. Produce una forma de onda oscilante que nunca se repite, figura (2.4), a diferencia de un oscilador electrónico ordinario, esto significa a grandes rasgos que el circuito de Chua es un

[1]https://people.eecs.berkeley.edu/ chua/papers/Matsumoto84.pdf

oscilador no periódico [14]. La facilidad de construcción de este circuito lo ha vuelto un ejemplo muy popular de un sistema caótico en la vida real.

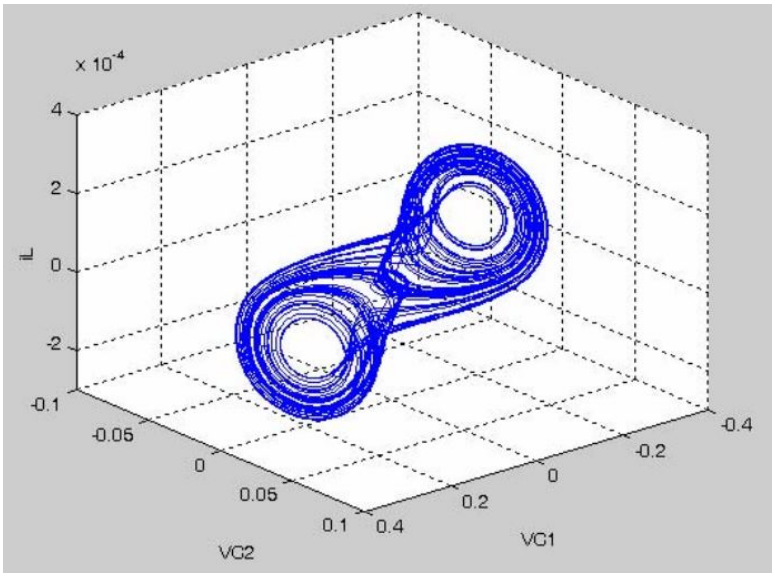


Figura 2.4: Atractor de Chua.[2]

Las ecuaciones del sistema caótico se obtienen haciendo un análisis del circuito de Chua, y las variables de estado son los voltajes a través de los capacitores y la corriente en el inductor [15]. Las ecuaciones de estado adimensionales [16] están dadas por:

$$\begin{aligned} \dot{x} &= \alpha(y - x - g(x)) \\ \dot{y} &= x - y + z \\ \dot{z} &= -\beta y - \gamma z \\ g(x) &= cx + (\frac{1}{2})(d - c)(|x + 1| - |x - 1|) \end{aligned} \tag{2.1.3}$$

Donde:

- x, y, z son las varibles de estado (corriente del inductor y voltajes en los capacitores)
- α, β, γ son constantes que dependen del valor del inductor y los capacitores

[2]http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4381349

- $g(x)$ corresponde a la función de comportamiento del elemento activo (memristor)

2.2. Sincronización de osciladores caóticos

Para llevar a cabo una efectiva transmisión de datos utilizando el método de enmascaramiento con señales caóticas, las señales del transmisor y receptor deben de estar sincronizadas, es decir, los osciladores caóticos deben de tener el mismo valor en sus variables de estado al mismo tiempo.

La sincronización entre sistemas caóticos se ha estudiado ampliamente debido a sus diversas aplicaciones, una forma de sincronización es aplicar las formas Hamiltonianas y la aproximación por observadores. Los detalles pueden ser encontrados en el trabajo de Sira-Ramírez [17]. En ese trabajo se resume que para cualquier sistema caótico se hace uso de un término adicional en las ecuaciones del esclavo u oscilador receptor, llamado observador, el cual consta del error entre el oscilador maestro y esclavo multiplicado por un vector de constantes.

Para sincronizar los osciladores utilizados en este trabajo, se llevó a cabo la misma aproximación, agregando los observadores a cada una de las ecuaciones del sistema Lorenz en el FPGA esclavo. Las ecuaciones quedan de la siguiente forma:

$$\begin{aligned} \dot{x} &= \sigma(y - x) + K1(xm - x) \\ \dot{y} &= (rx - xz - y) + K2(xm - x) \\ \dot{z} &= (xy - bz) + K3(xm - x) \end{aligned} \tag{2.2.1}$$

Donde el término xm es la señal proveniente del FPGA maestro.

En estas ecuaciones, el parámetro de diseño es el valor de las constantes $K1$, $K2$ y $K3$, de las cuales depende si los sistemas llegan a sincronizarse y el tiempo que esto les toma.

2.3. Métodos de integración

Método de integración es cualquier técnica elemental utilizada para calcular el valor numérico de la integral definida de una función. Para generar la señal caótica a partir de las ecuaciones de estado, se tiene que hacer uso de un método de integración numérico. El problema considerado por los métodos de integración numérica, es calcular la solución aproximada del sistema de ecuaciones diferenciales. Existen diversos métodos de integración numérica que varían ampliamente en complejidad.

El método de integración Forward Euler es el más simple para resolver un problema con condiciones iniciales de la forma (2.2.1). Permite obtener la solución deseada muy rápidamente a costa de exactitud. Por otro lado, su implementación en un FPGA es la que menos recursos utiliza y provee la velocidad de procesamiento más alta entre todas las formas de integración numérica.

Un sistema de ecuaciones diferenciales puede ser expresado como:

$$\begin{aligned} y' &= f(x, y) \\ y(a) &= A \end{aligned} \tag{2.3.1}$$

Expandiendo usando series de Taylor, se obtiene:

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(\xi_n) \tag{2.3.2}$$

Si escogemos un valor pequeño para h, el último término puede ser despreciado, obteniendo así la fórmula iterativa del método Forward Euler [18]:

$$y(x_{n+1}) \approx y(x_n) + hf(x_n, y(x_n)) \tag{2.3.3}$$

Un método de integración numérico más elaborado y muy utilizado para resolver problemas con condiciones iniciales es el método Runge-Kutta de 4.º orden. Este método utiliza varias evaluaciones de $f(x, y)$ en una combinación lineal para aproximar $y(x)$. La fórmula iterativa es la siguiente:

$$
\begin{aligned}
K_0 &= F(X_n, Y_n) \\
K_1 &= F(X_n + \frac{h}{2}, Y_n + \frac{h}{2}K_0) \\
K_2 &= F(X_n + \frac{h}{2}, Y_n + \frac{h}{2}K_1) \\
K_3 &= F(X_n + h, Y_n + hK_2) \\
Y_{n+1} &= Y_n + \frac{h}{6}(K_0 + 2K_1 + 2K_2 + K_3)
\end{aligned} \tag{2.3.4}
$$

Donde F es el conjunto de ecuaciones a ser resueltas, X_n son las condiciones iniciales, Y_n es el estado actual de las variables, h es el tamaño de paso y Y_{n+1} es el siguiente estado de las variables. A pesar de que el método de Runge-Kutta es más preciso, utiliza una gran cantidad de recursos como se muestra en la tabla (2.1), en donde se programó ambos métodos de integración para calcular la señal caótica, obteniendo esos resultados. Se aprecia que el método Forward Euler utiliza aproximadamente seis veces menos recursos que el método de integración Runge-Kutta.

Tabla 2.1: Recursos utilizados por los métodos de integración para calcular la señal caótica.

Recursos	**Lorenz FE**	**Lorenz RK**	**Disponible**
Total de elementos logicos	622	3560	33,216
Total de registros	124	124	33,216
Total de pines	5	5	475
Multiplicadores embebidos de 9 bits	8	70	70

2.4. Interfaz y funcionamiento de los módulos inalámbricos Xbee

Una manera de comunicar dos dispositivos de manera inalámbrica es mediante el uso de módulos externos llamados Xbee, estos módulos trabajan utilizando el protocolo de comunicación RS232 tanto en la interfaz como en la trasmisión de información entre el dispositivo y el propio Xbee. La comunicación inalámbrica

entre módulos Xbee utiliza otro protocolo llamado Zigbee, el cual está basado en el estándar para comunicaciones inalámbricas IEEE 802.15.4, creado por Zigbee Alliance. Esto es útil para redes de sensores en ambientes industriales, médicos y domésticos. Las comunicaciones se llevan a cabo en la banda libre de los 2.4 GHz, y a diferencia del Bluetooth, este protocolo utiliza una sola frecuencia.

La figura (2.5) muestra las conexiones mínimas que necesita el módulo Xbee para poder ser utilizado. Luego de esto, se debe configurar según el modo de operación que se desea para la aplicación requerida por el usuario. Asi mismo se requiere una alimentación de 3.3V, tierra y dos líneas de transmisión de datos señaladas como TXD y RXD. La velocidad de transmisión que se utiliza por defecto es de 9,600 baudios, pero puede ser aumentada (teóricamente) hasta 115,200 baudios, siendo esta la velocidad máxima de los módulos inalámbricos Xbee.

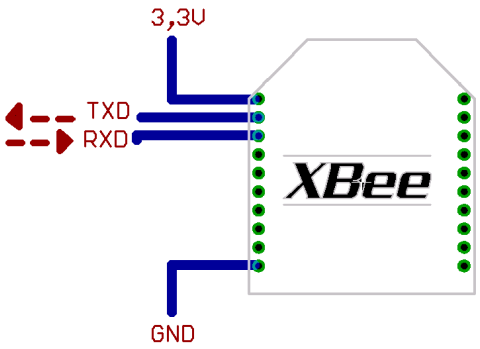


Figura 2.5: Conexiones mínimas requeridas para el Xbee.[3]

Los módulos Xbee tienen 5 estados de operación, como se muestra en la figura (2.6). Los modos principales de operación son el de recibir y transmitir, y se entra a estos modos cuando llega algún paquete de radio frecuencia a través de la antena o cuando se manda información al buffer del pin RXD. La información puede ser enviada ya sea de forma específica a un solo dispositivo o a todos los nodos en una red, para el caso de este trabajo se utilizarála primer forma.

[3]http://olimex.cl/website_MCI/static/documents/XBee-Guia_Usuario.pdf
[4]http://olimex.cl/website_MCI/static/documents/XBee-Guia_Usuario.pdf

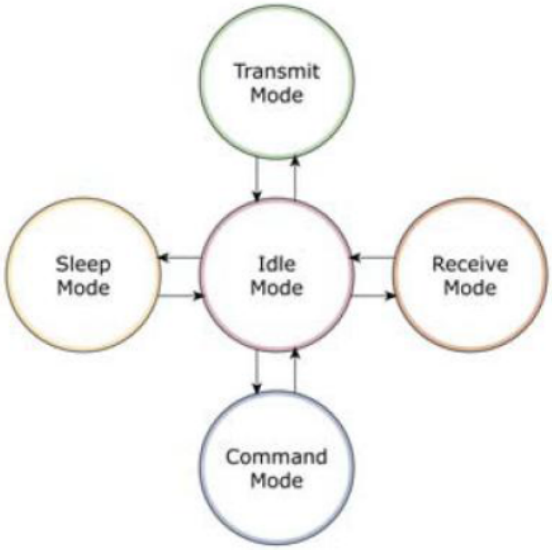


Figura 2.6: Modos de operación del módulo Xbee.[4]

El siguiente modo del cual se hará uso es el modo de comando, este modo permite ingresar comandos al módulo Xbee para configurar, ajustar o modificar parámetros, principalmente la dirección propia o la de destino así como su modo de operación entre otras cosas. Para poder ingresar a este modo, se debe esperar un tiempo dado (por defecto 1 segundo), luego se debe transmitir hacia el pin RXD la cadena de caracteres **"+++"** y esperar otro tiempo similar. Como respuesta el módulo entregará un **"OK"** por el pin TXD.

La figura (2.7) muestra la sintaxis de un comando de configuración. Luego de ingresar al modo de comando, se debe ingresar el comando deseado usando esta sintaxis.

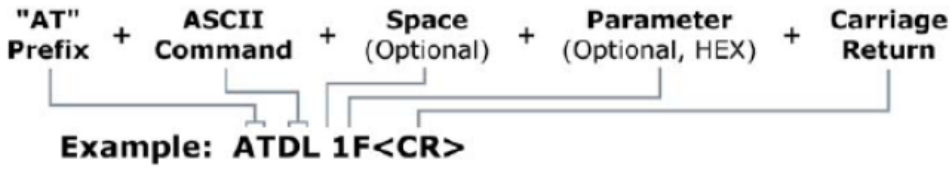


Figura 2.7: Ejemplo Comando AT.

Una vez finalizada la configuración, se ingresa el comando ATCN seguido de un retorno de carro para salir del modo de comando.

Los comandos a utilizar en este trabajo son los siguientes:

- ATMY - Configura la dirección de 16 bits para el módulo.
- ATDL – Ajusta los 32 bits menos significativos para direccionamiento.
- ATBD – Ajusta la tasa de transmisión entre el módulo y su cliente conectado a través de la interfaz serial.
- ATNB – Ajusta la paridad para la comunicación serial.

Para realizar la conexión punto a punto, todos los módulos deben de estar configurados de tal forma que tengan diferentes direcciones, usando el comando MY, y cada que se desea hacer una transmisión, se debe usar el comando DL para asignar el módulo al cual se desea conectar. Un ejemplo se puede apreciar en la figura (2.8).

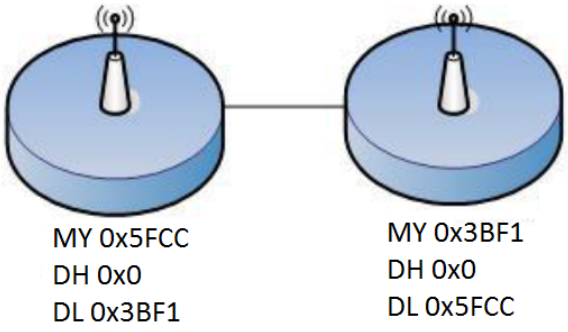


Figura 2.8: Ejemplo direccionamiento 16 bits.[5]

[5]http://olimex.cl/website_MCI/static/documents/XBee-Guia_Usuario.pdf

Capítulo 3

Metodología

3.1. Diseño de los observadores

Como ya se mencionó previamente, para alcanzar una sincronización entre dos osciladores caóticos se necesita un término adicional llamado observador en las ecuaciones del oscilador esclavo. Este término consta de la diferencia entre la variable de estado transmitida y la generada por el receptor multiplicado por un vector de constantes **K**.

Para este trabajo se experimentó con un algoritmo de simulación, el cual procedió a generar un gran número de combinaciones posibles para el valor del vector **K** de forma iterativa, para ser aplicados a las ecuaciones del sistema Lorenz con observadores, luego, se procedió a simular un sistema Lorenz maestro para sincronizar el sistema esclavo a este, almacenando si al cabo de 100 iteraciones se llega a una sincronización estable o inestable y en cuantas iteraciones se logra esto. Se escogió 100 iteraciones para el límite ya que luego de unas pruebas se encontró que existen valores de **K** que lograron una sincronización en aproximadamente 90 iteraciones.

Además, esta simulación se repitió variando de forma aleatoria las condiciones iniciales tanto del sistema maestro como el sistema esclavo, obteniendo así un resultado estadísticamente más confiable que utilizar un valor único de condiciones iniciales. Se tomó una población de 30 condiciones iniciales aleatorias, generando así un alto nivel de confianza en el resultado del número de iteraciones que le lleva al sistema sincronizarse. Un resultado gráfico de esta prueba se puede apreciar en la figura (3.1). En esta se aprecia que al variar las constantes K2 y K3 pertenecientes al término observador,

el número promedio de iteraciones en que los osciladores caóticos se sincronizan varían (error absoluto menor a 0.001), y se dejó constante K1 ya que se encontró con una prueba similar que el mejor valor es 1. El color rojo representa 100 o más iteraciones, mientras que el color azul obscuro representa un valor de 74 iteraciones. El punto con la menor cantidad de iteraciones resultó ser K1 = 1, K2 = 1.9 y K3 = 0.1.

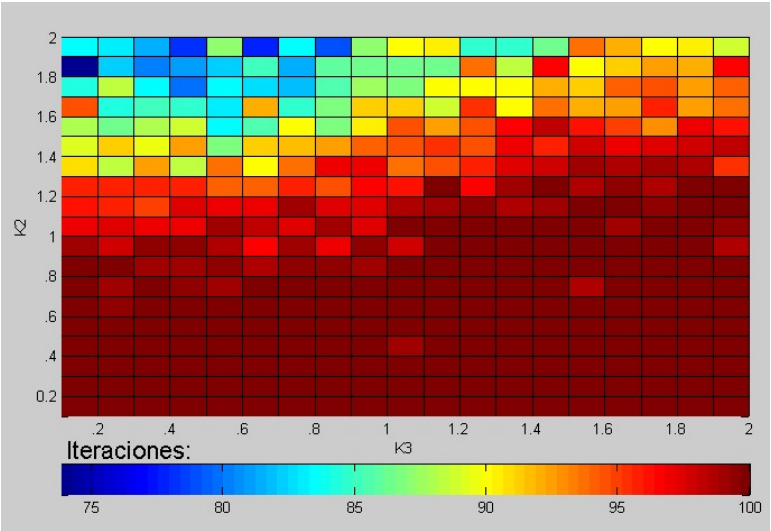


Figura 3.1: Gráfica de número de iteraciones para sincronizar dos osciladores variando el término observador.

Para hacer esto, se analizó la distribución de los valores de las variables de estado para obtener la media y la desviación estándar y poder así generar condiciones iniciales aleatorias apegadas al comportamiento real del sistema. Se encontró que tanto $\dot{x}$ como $\dot{y}$ poseen una media cercana a cero y una desviación estándar de aproximadamente 8.5, mientras que $\dot{z}$ tiene una media de 25 y una desviación estándar de aproximadamente 7.5.

Finalmente se tomaron los valores que producían una sincronización entre los osciladores y se comparó el tiempo de sincronización para encontrar los valores óptimos que producen una sincronización lo más rápido posible. En la figura (3.2) se muestran las gráficas de sincronización luego de encontrar el valor óptimo del vector **K**, las gráficas (a) y (b) muestran el valor de la variable de estado en el oscilador maestro y esclavo, mientras que la (c) muestra el error o diferencia entre estas señales y la (d) grafica una contra la otra, apreciándose la tendencia a formar una línea recta a 45.º, lo cual significa que las señales están sincronizadas.

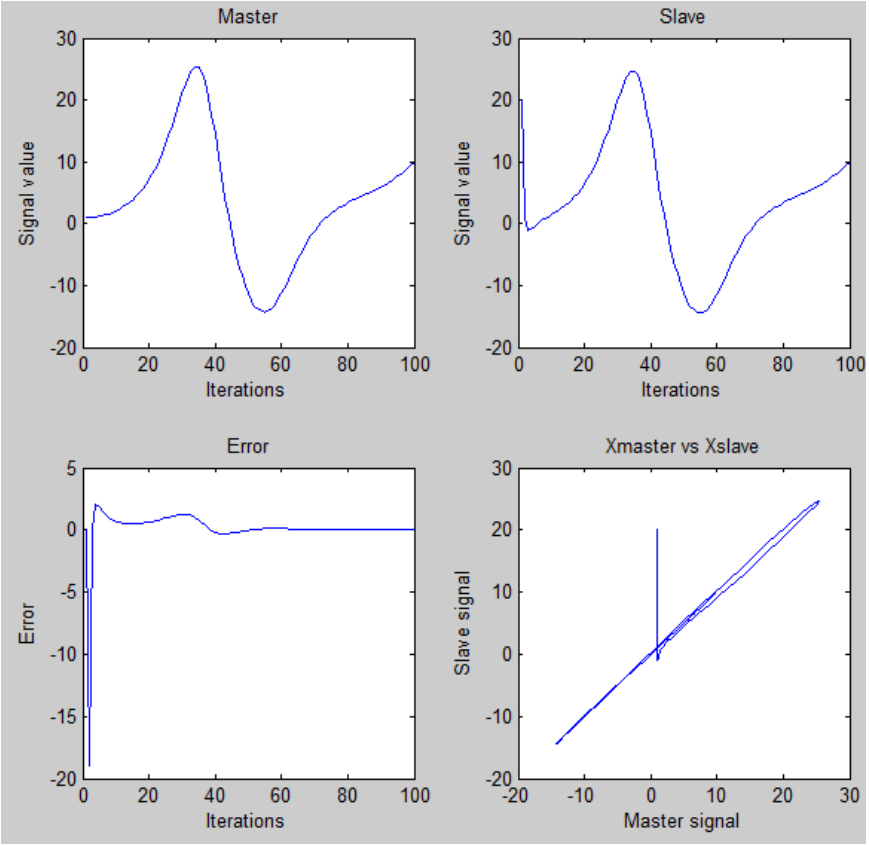


Figura 3.2: Gráficas de sincronización.

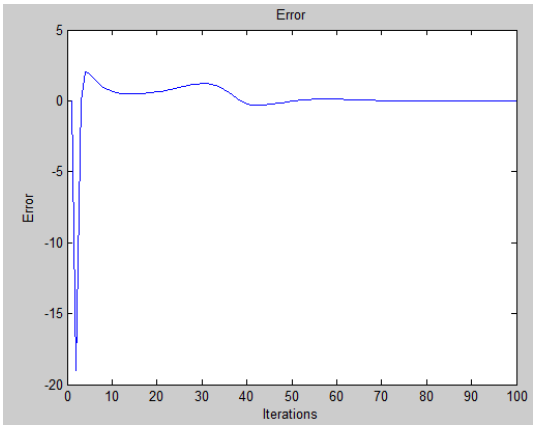
En comparativa, en la figura (3.3) se muestran las gráficas de error para distintos valores del vector **K**, pudiéndose apreciar claramente que los valores escogidos son los más óptimos, ya que la sincronización se alcanza con el menor número de iteraciones.

En la figura (3.4) se muestran las gráficas del error entre las señales X, Y y Z del transmisor y receptor, en dicha gráfica se aprecia que la variable de estado X es la que se sincroniza en menor cantidad de iteraciones.

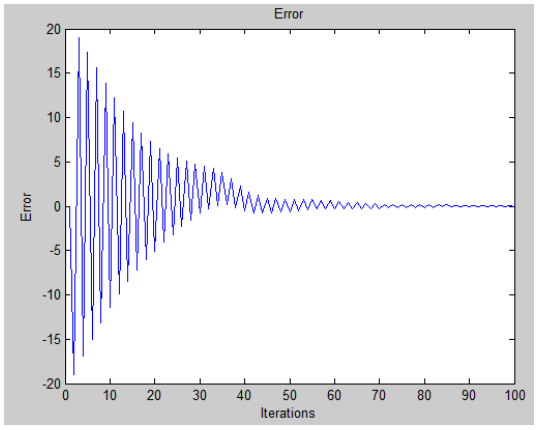
3.2. Protocolo de transmisión

Para llevar a cabo la comunicación, se diseñó el siguiente protocolo:

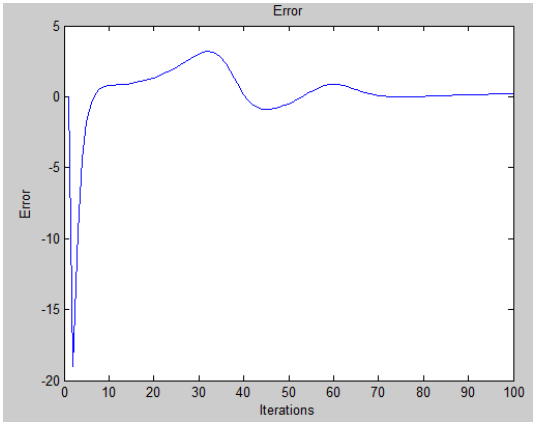
1.- Seleccionar el archivo a transmitir y el dispositivo destino.



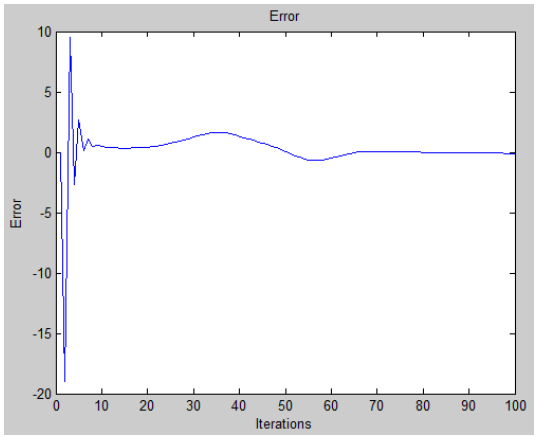
(a) k1 = 1, k2 = 1.9, k3 = 0.1



(b) k1 = 2, k2 =2.9, k3 = 1.7



(c) k1 = 0.5, k2 = 0.5, k3 = 0.5



(d) k1 = 1.5, k2 = 2.4, k3 = 3

Figura 3.3: Gráficas de error para distintos valores del vector K.

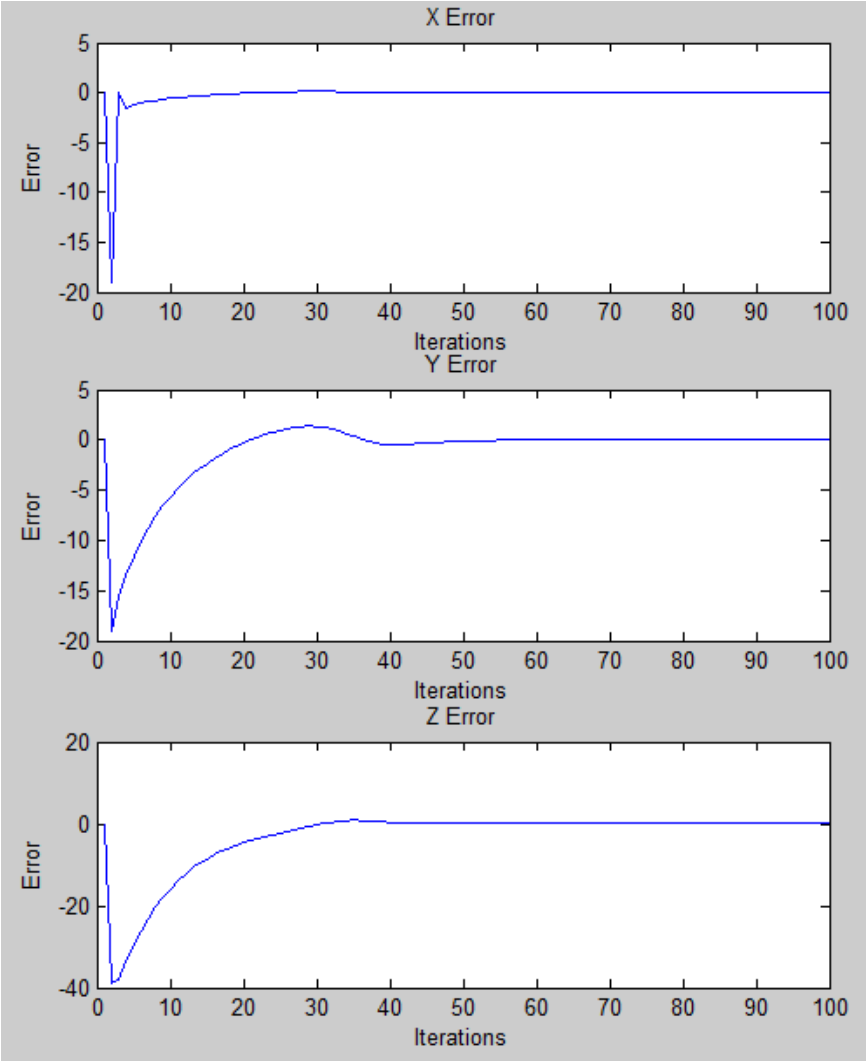


Figura 3.4: Grafica de los observadores X, Y y Z.

2.- Configurar el módulo Xbee con el nuevo dispositivo destino usando el comando ATDL.

3.- Transmitir la señal caótica para iniciar el proceso de sincronización hasta que el receptor le indique al transmisor cuando está sincronizado.

4.- Comenzar la transferencia del archivo enmascarado por la señal caótica, comenzando por el tamaño del archivo para que el receptor pueda calcular el porcentaje de progreso.

5.- Esperar a que finalice la transmisión para luego regresar al estado espera en

ambos dispositivos.

Paso 1. El usuario debe de seleccionar la imagen o cualquier otro tipo de archivo a enviar por medio de una interfaz gráfica en la computadora, además de seleccionar el destino. Al momento de que el usuario indica que se inicie la transmisión, la computadora transmite al FPGA el dispositivo destino para proceder con el siguiente paso.

Paso 2. Al recibir la información del dispositivo destino, el FPGA procede a configurar el módulo Xbee con la siguiente secuencia. Primero, se procede a entrar al modo de comandos enviando la cadena **"+++"** y se espera a que el módulo Xbee conteste con un **"OK"**. Segundo, se utiliza el comando ATDL seguido del identificador del dispositivo destino, esperando nuevamente la confirmación del módulo. Finalmente, se termina con el comando ATCN para salir del modo de comandos y proceder al paso 3.

Paso 3. Luego de configurar el módulo Xbee, se procede a transmitir la señal caótica sin datos para comenzar el proceso de sincronización, enviando un byte a la vez y esperando la confirmación del receptor. Si la confirmación es 0xFA, significa que no se ha alcanzado la sincronización y se necesita enviar otro byte, en cambio si la confirmación es 0xB1, el oscilador esclavo se encuentra totalmente sincronizado. Estas confirmaciones las calcula el FPGA receptor, haciendo una resta entre la señal recibida con la señal calculada, cuando el error es cero por más de 255 iteraciones, se asegura que se ha alcanzado la sincronización tanto de la variable de estado transmitida como de las otras dos.

Paso 4. Una vez alcanzada la sincronización, el FPGA transmisor le envía una señal de confirmación a la computadora para que esta proceda a transmitir el archivo previamente seleccionado. La primera información enviada corresponde al tamaño del archivo para que la computadora receptora pueda mostrar el porcentaje de progreso de transmisión, y el resto corresponde a la totalidad de bytes del archivo, cada uno de estos enmascarado por la señal caótica por medio de la compuerta XOR.

Paso 5. Cuando la cantidad de bytes recibidos alcanza al tamaño del archivo enviado al inicio, la computadora receptora finaliza el proceso guardando el archivo recibido en el disco duro de la computadora y le envía al FPGA receptor una señal de

finalización (caracter "F"), la cual es retrasmitida al FPGA transmisor, haciendo que ambos dispositivos regresen a su estado inicial de espera. La computadora transmisora al finalizar de enviar el archivo regresa a su estado inicial de espera.

3.3. Simulación en LabVIEW

Se decidió hacer una simulación del funcionamiento interno del código para la generación y sincronización de las señales caóticas, como se muestra en la figura (3.5), pudiendo así hacer pequeños ajustes de manera eficiente para encontrar las condiciones iniciales que generan oscilaciones caóticas estables y sostenidas, ya que no todas las condiciones iniciales las generan.

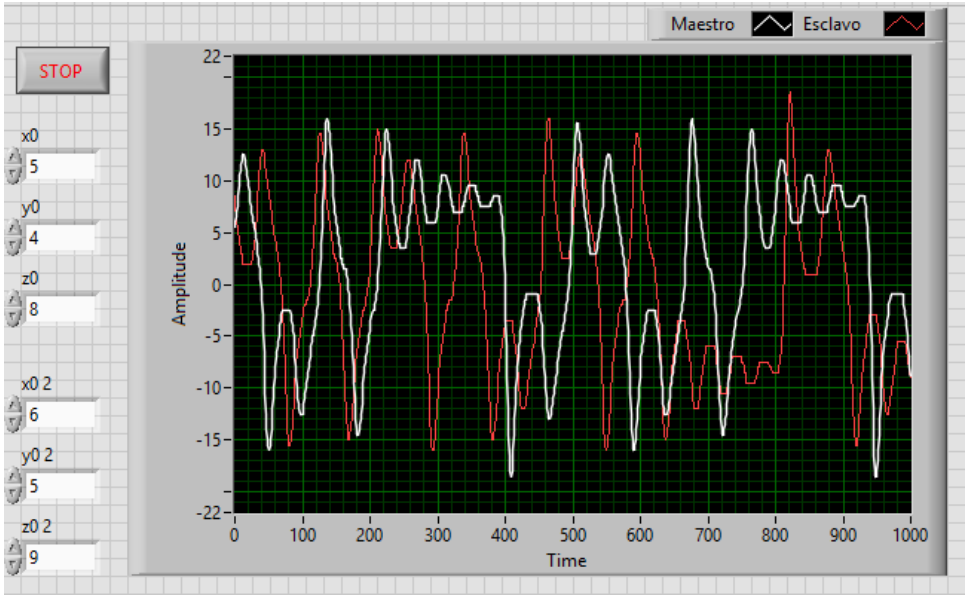


Figura 3.5: Panel frontal de la simulación de los dos osciladores caóticos.

Gracias a esta simulación, se pudo probar todas las combinaciones posibles usando un algoritmo iterativo similar al que se usó para calcular el vector **K** del observador, encontrando así que las mejores condiciones iniciales para las variables de estado son $x_0 = 5$, $y_0 = 4$ y $z_0 = 8$, debido a que estas generan oscilaciones sostenidas y suficientemente caóticas.

Además, se ajustaron ligeramente los valores del vector **K** para reducir el número de iteraciones para la sincronización, ya que debido a la baja precisión de la variable

de estado x (7 bits enteros y 1 bit decimal), utilizar los valores exactos previamente calculados para **K** no aseguran una sincronización exitosa. Los valores finales, luego de ajustarlos a la precisión escogida de punto fijo, resultaron ser [1, 1.90625, 0.09375], como se muestra en la figura (3.6).

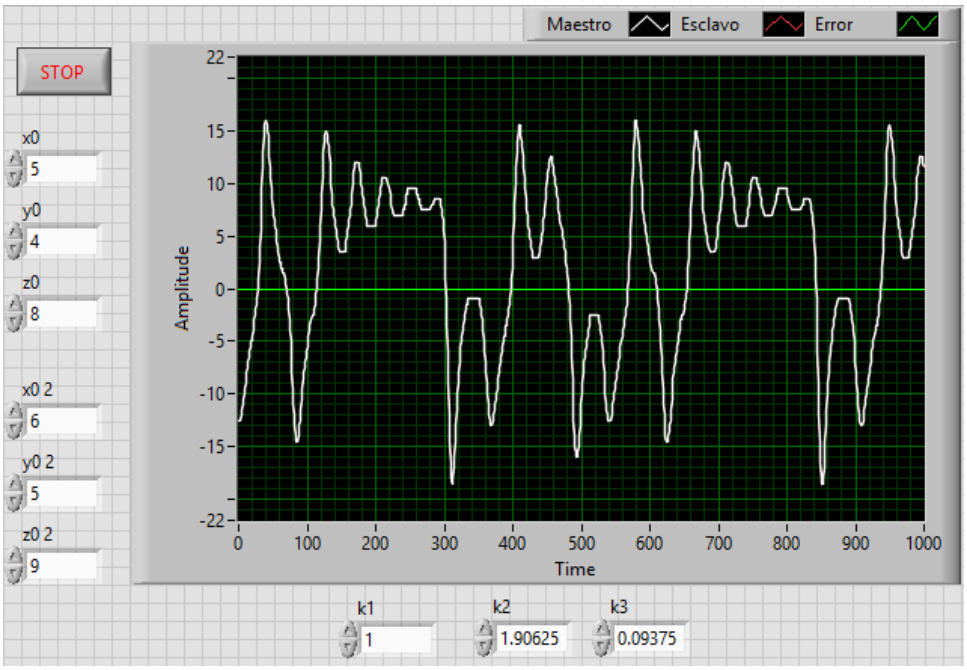


Figura 3.6: Panel frontal de la simulación de los osciladores caóticos sincronizados.

Finalmente, la simulación también sirvió para ver el comportamiento de las dos señales caóticas conforme se sincronizan, pudiendo así encontrar en qué punto exacto se puede asegurar que todas las variables de estado están sincronizadas, siendo este punto, luego de aplicar un factor de seguridad, 255 iteraciones después del último punto con error diferente de cero.

3.4. Realización en FPGA

Teniendo en claro las ecuaciones, el valor del vector **K**, las condiciones iniciales y el tiempo de sincronización, se procedió a implementar en VHDL cada una de las partes que conforman el sistema transmisor y receptor.

Como primer paso, se diseñaron los bloques de transmisión y recepción correspondientes al protocolo RS232, generando las bases de tiempo, las máquinas de estados y

los diagramas a bloques de estos módulos. A continuación, se implementaron las ecuaciones del sistema Lorenz en código VHDL, tanto las ecuaciones del maestro como las ecuaciones del esclavo, que incluyen los observadores. Después, se diseñó el diagrama a bloques del funcionamiento del sistema de encriptación, trasmisión y recepción. Así también, se crearon máquinas de estados que administran cada paso del protocolo de comunicación. Finalmente, para corroborar el correcto funcionamiento del sistema, se utilizó la herramienta de simulación del software Active-VHDL para visualizar si se generan señales caóticas y si se sincronizan correctamente.

3.4.1. Diagrama a bloques

Tomando en cuenta las ecuaciones del sistema Lorenz, se procedió a diseñar el diagrama a bloques que se encarga de realizar la integración numérica por el método Forward Euler, como se aprecia en la figura (3.7), simplificando lo más posible las ecuaciones para hacer uso de la menor cantidad de multiplicadores, además el software de compilación posee un algoritmo de optimización que realizó una simplificación adicional automática, logrando hacer uso de solo 4 elementos multiplicadores embebidos de 9 bits.

De igual forma, se realizó el diagrama a bloques de las ecuaciones incluyendo los términos observadores, pudiéndose apreciar este en la figura (3.8). Las entradas de estos diagramas ($X0$, $Y0$ y $Z0$) provienen de un registro, el cual se encarga de retener el valor de las variables de estado, inicializándose con las condiciones iniciales previamente calculadas y actualizándose con el siguiente valor ($X1$, $Y1$ y $Z1$) en cada iteración. Para el oscilador esclavo, se cuenta con una entrada adicional (Xm), la cual corresponde a la señal caótica generada por el oscilador maestro.

Una vez hecho el bloque que se encarga de generar la señal caótica, se procedió a realizar el diagrama a bloques de la parte que realiza la encriptación, trasmisión y recepción, mostrado en la figura (3.9). Este diagrama consta del bloque de ecuaciones del sistema Lorenz, dos bloques de transmisión y dos de recepción, un par para la interfaz FPGA-Xbee y otro para la interfaz FPGA-PC, además se muestra el uso de bases de tiempo para cumplir con el protocolo de configuración del módulo Xbee. Finalmente, se hace uso de una máquina de estados para controlar la secuencia en la

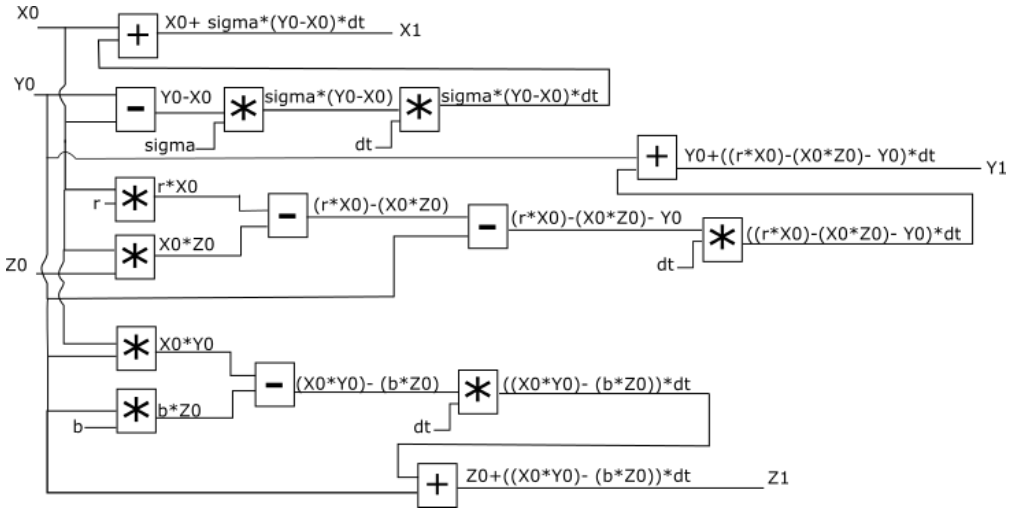


Figura 3.7: Diagrama a bloques de las ecuaciones del sistema Lorenz.

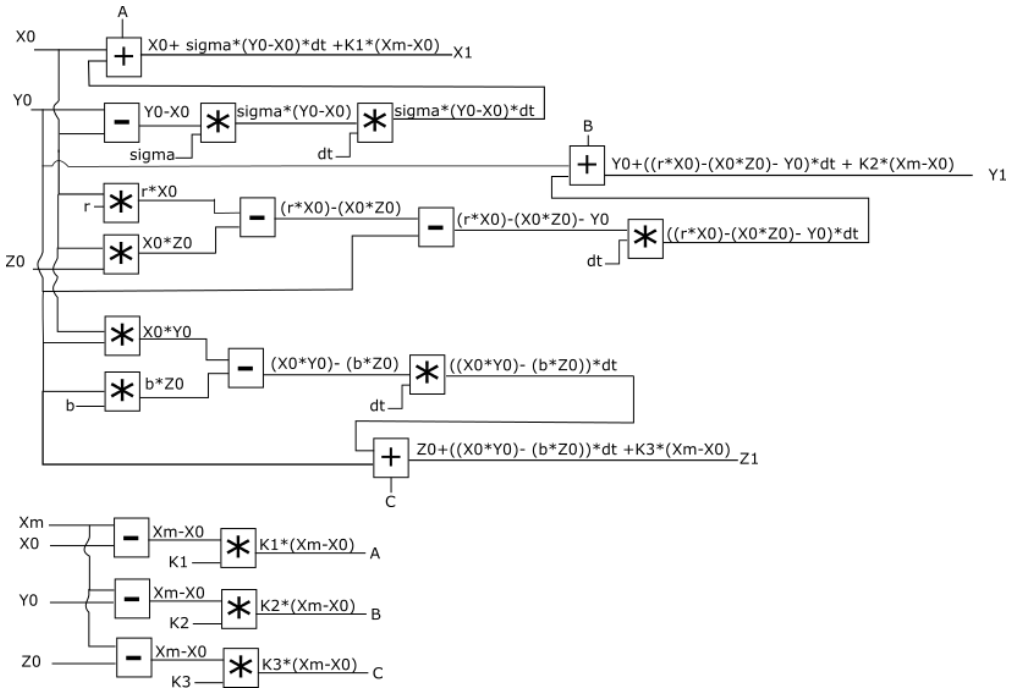


Figura 3.8: Diagrama a bloques de las ecuaciones del sistema Lorenz con observadores.

que cada módulo opera.

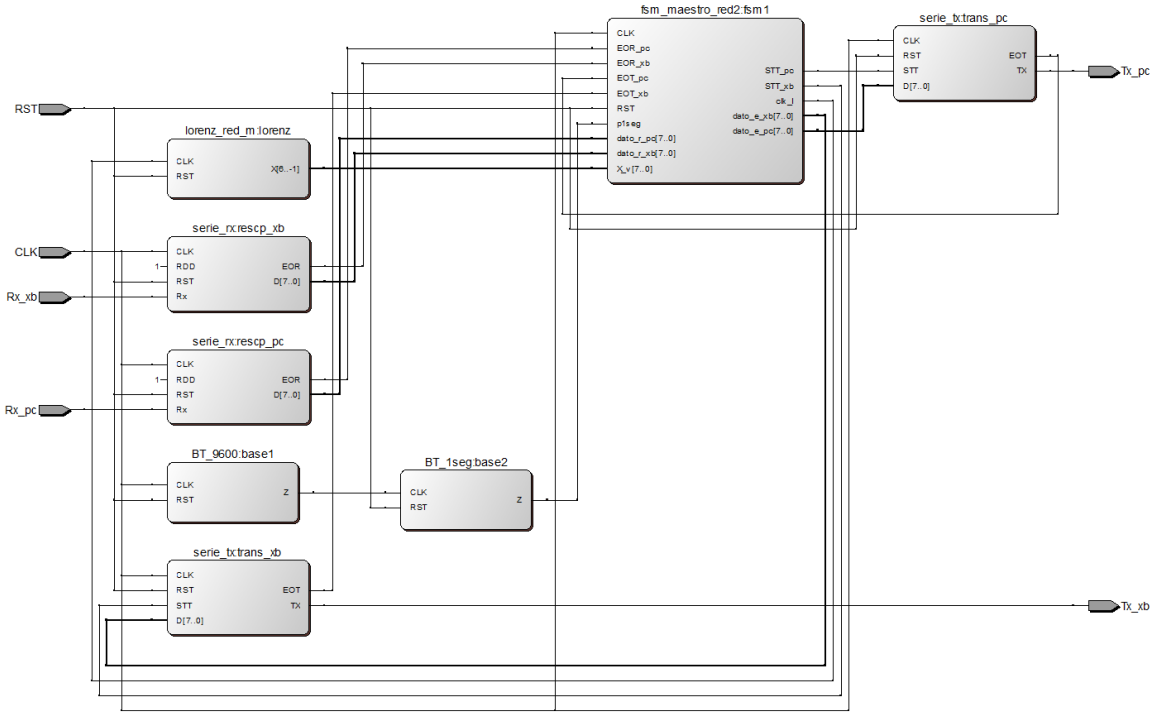


Figura 3.9: Diagrama a bloques del FPGA transmisor.

Debido a que el FPGA receptor no necesita configurar su módulo Xbee, se omiten los bloques de base de tiempo, y se sustituye el bloque de ecuaciones por el que contiene el sistema Lorenz con observadores. Este diagrama se muestra en la figura (3.10).

3.4.2. Máquina de estados

El protocolo de transmisión requiere del uso de máquinas de estados para el sistema transmisor y receptor, ya que el proceso de comunicación consta de varias etapas, las cuales son controladas por varios grupos de estados. Estas etapas son la configuración del módulo Xbee, la sincronización entre los osciladores caóticos y la transmisión de datos encriptados. A continuación se describe la máquina de estados que posee el FPGA transmisor. La tabla (3.1) muestra las entradas y salidas de dicha máquina.

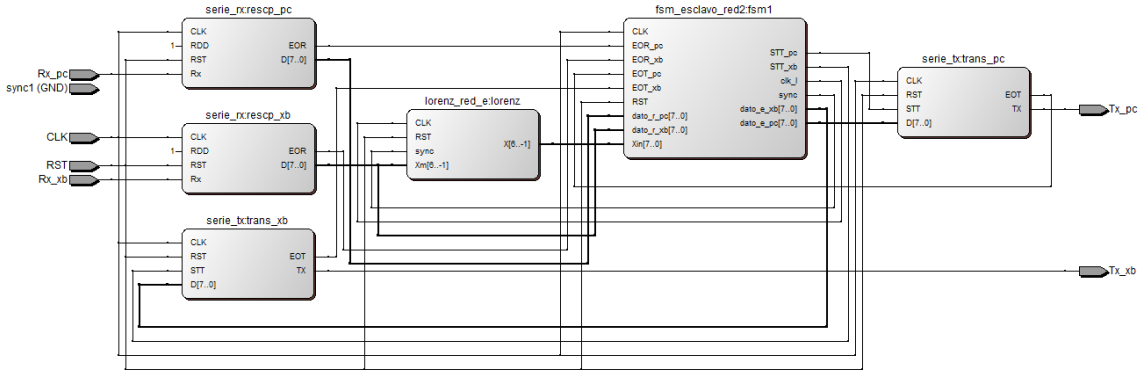


Figura 3.10: Diagrama a bloques del FPGA receptor.

Tabla 3.1: Entradas y salidas de la máquina de estados del FPGA transmisor.

Entradas	**Salidas**
EOR_pc	$a : CLK\_l$
EOT_pc	$b : dato\_e\_pc$
EOR_xb	$c : STT\_pc$
EOT_xb	$d : dato\_e\_xb$
$p1seg$	$e : STT\_xb$
$dato\_r\_xb$	
$dato\_r\_pc$	
X_v	

Las entradas que tienen el prefijo EOR y EOT corresponden las banderas de fin de recepción y fin de transmisión de los bloques de comunicación correspondientes al módulo Xbee y a la PC. La entrada $p1seg$ es una bandera proveniente de una base de tiempo que se activa cada segundo. Las entradas que tienen el prefijo $dato\_r$ corresponden al dato recibido desde el Xbee y la PC en un momento dado. Finalmente, la entrada X_v es la variable de estado proveniente de la solución al sistema de ecuaciones del sistema Lorenz.

La salida CLK_l es una bandera que le indica al bloque de integración numérica cuando generar un nuevo paso en la solución del sistema Lorenz. Las salidas con el prefijo STT corresponden a las banderas de inicio de transmisión de los bloques de comunicación correspondientes al módulo Xbee y a la PC. Finalmente, las salidas $dato\_e$ corresponden a los datos que se van a enviar a través de los bloques de

comunicación.

Todos los estados permanecen en sí mismos mientras no se cumpla la condición de cambio, a excepción de los que cambian con un pulso de reloj (CLK).

Los primeros tres estados, mostrados en la figura (3.11), se encargan de esperar a que llegue un dato desde la PC, el cual según el protocolo, es el identificador del Xbee destino. El primer estado espera a que se detecte un inicio de transmisión de datos, pasando al segundo estado donde se espera a que esta transmisión termine y en el tercero, se almacena el contenido de la entrada *dato_r_pc* en un registro llamado *numero*.

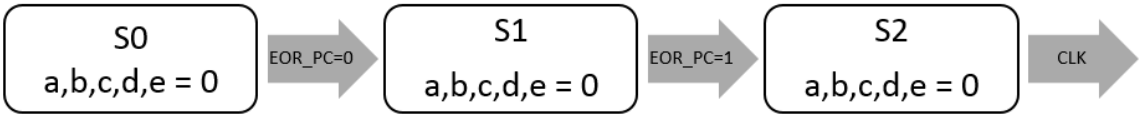


Figura 3.11: Estados 0, 1 y 2 de la máquina de estados del FPGA transmisor.

Los siguientes tres estados, mostrados en la figura (3.12), generan un retardo de poco más de 1 segundo, ya que, para ingresar al modo de comandos del Xbee, se debe de dejar pasar este tiempo antes de enviar la instrucción.

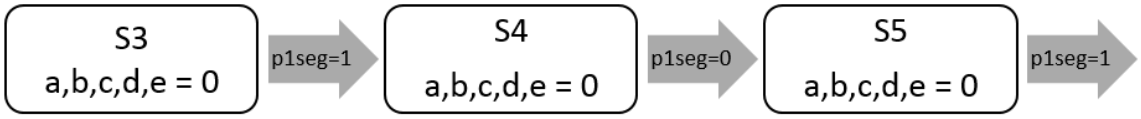


Figura 3.12: Estados 3, 4 y 5 de la máquina de estados del FPGA transmisor.

Los estados 6 y 7, mostrados en la figura (3.13), son estados genéricos para transmitir un caracter hacia el módulo Xbee. Estos se repiten tantas veces como caracteres a enviar para llevar a cabo la configuración.

Los estados 12 y 13, mostrados en la figura (3.14), al igual que los estados 6 y 7, son estados genéricos para manejar la respuesta del módulo Xbee ante los comandos de configuración. Como se explicó previamente en la sección 2.4, luego de enviar un comando el Xbee contesta tres caracteres (O, K, retorno de carro), así que se debe de incluir este par de estados por cada caracter que el Xbee envía de regreso.

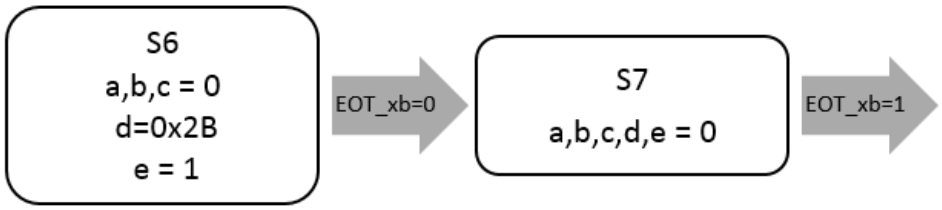


Figura 3.13: Estados 6 y 7 de la máquina de estados del FPGA transmisor.

Figura 3.14: Estados 12 y 13 de la máquina de estados del FPGA transmisor.

Los estados del 6 al 51, son los encargados de llevar a cabo la configuración descrita en el paso 2 de la sección 3.2.

Los estados 52 y 53, mostrados en la figura (3.15), corresponden a la generación de una nueva iteración y su subsecuente transmisión al Xbee destino, esto con el fin de proporcionarle un nuevo dato al sistema receptor para que este compruebe si se ha llevado a cabo o no la sincronización.

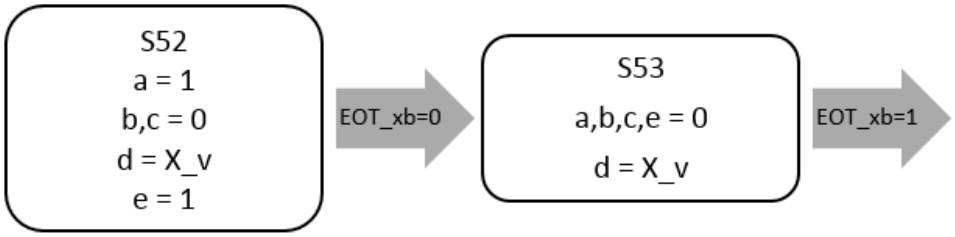


Figura 3.15: Estados 52 y 53 de la máquina de estados del FPGA transmisor.

Los estados 54, 55 y 56, mostrados en la figura (3.16), se encargan de recibir la confirmación del sistema receptor y dependiendo de su valor, ir al estado correspondiente.

Los estados 57, 58 y 59, mostrados en la figura (3.17), se encargan de transmitir la confirmación a la PC para que esta comience a transmitir los datos del archivo

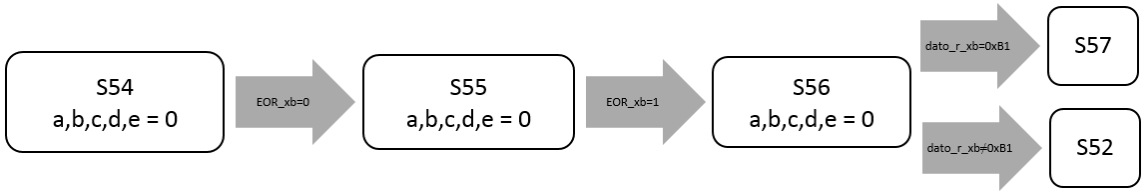


Figura 3.16: Estados 54, 55 y 56 de la máquina de estados del FPGA transmisor.

seleccionado. Si la PC envía un dato, se procede al estado 60, en cambio si el Xbee es el que envía significa que la comunicación ha finalizado y se regresa al estado inicial.

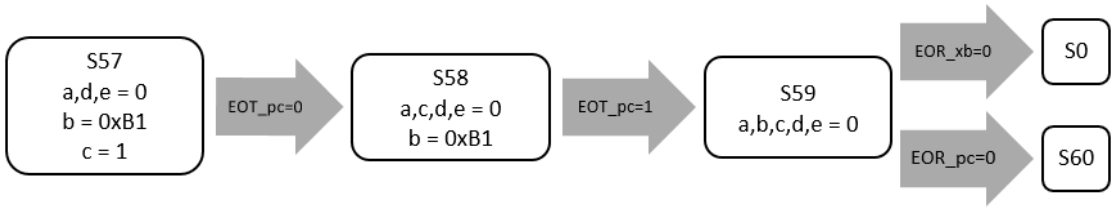


Figura 3.17: Estados 57, 58 y 59 de la máquina de estados del FPGA transmisor.

Los estados 60, 61 y 62, mostrados en la figura (3.18), realizan el enmascaramiento de los datos recibidos desde la PC con la señal caótica generada por el sistema de ecuaciones para transmitirlos hacia el sistema receptor.

Figura 3.18: Estados 60, 61 y 62 de la máquina de estados del FPGA transmisor.

En conjunto, todos los estados mencionados anteriormente conforman la totalidad de la máquina de estados correspondiente al FPGA transmisor, siendo estos los que proporcionan el control de recepción, configuración, enmascaramiento y transmisión de datos.

La tabla (3.2) enlista las entradas y salidas de la máquina de estados que posee el FPGA receptor. Como se aprecia en la tabla, algunas de las entradas y salidas tienen una función similar a la máquina de estados anterior. La entrada Xin es la

variable de estado proveniente de la solución al sistema de ecuaciones del sistema Lorenz con observadores. La entrada con_flg representa la bandera de salida de un contador de 8 bits, cuyas entradas son con_clk y con_rst, este contador se encuentra integrado al bloque de esta máquina de estados y es el encargado de medir el número de iteraciones donde el error es igual a cero.

Por su parte, la salida *sync* es una bandera que le indica al bloque de integración numérica si añadir o no los observadores al sistema de ecuaciones que genera la señal caótica.

Tabla 3.2: Entradas y salidas de la máquina de estados del FPGA receptor.

Entradas	**Salidas**
EOR_pc	$a : sync$
EOT_pc	$b : CLK\_l$
EOR_xb	$c : dato\_e\_pc$
EOT_xb	$d : STT\_pc$
Xin	$e : dato\_e\_xb$
$dato\_r\_xb$	$f : STT\_xb$
$dato\_r\_pc$	$g : con\_clk$
con_flg	$h : con\_rst$

Los estados iniciales, 0, 1, 2 y 3, mostrados en la figura (3.19), son los encargados de estar a la espera de que se reciba un dato proveniente del módulo Xbee, ante lo cual generan una nueva iteración y comprueban si el dato recibido coincide con el dato generado internamente. Sí los dos datos son iguales, los osciladores ya se encontraban sincronizados debido a una transmisión previa y se procede a ir directamente al enmascaramiento y transmisión de datos, en caso contrario, se lleva a cabo el proceso de sincronización.

En los estados 4, 5 y 6, mostrados en la figura (3.20), se procede a enviar la confirmación negativa de sincronización al FPGA maestro y se está a la espera de un nuevo dato.

En los estados 7 y 8, mostrados en la figura (3.21), se procesa el dato recibido desde el FPGA maestro y se compara con el estado actual de la señal caótica generada internamente. Sí los dos datos son iguales, se procede a activar el reloj del contador

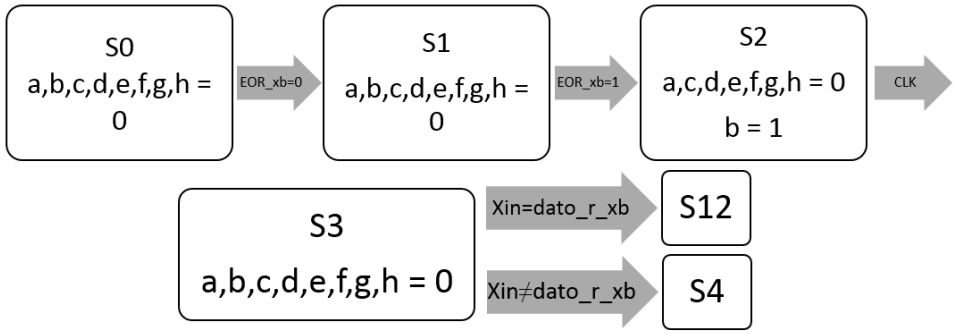


Figura 3.19: Estados 0, 1, 2 y 3 de la máquina de estados del FPGA receptor.

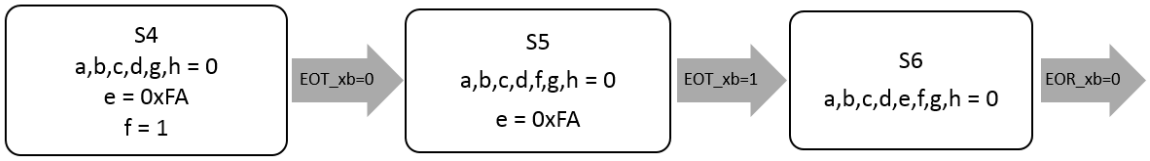


Figura 3.20: Estados 4, 5 y 6 de la máquina de estados del FPGA receptor.

interno, en caso contrario, el contador se reinicia. Esto nos permite llevar la cuenta de cuantos datos sincronizados se han recibido.

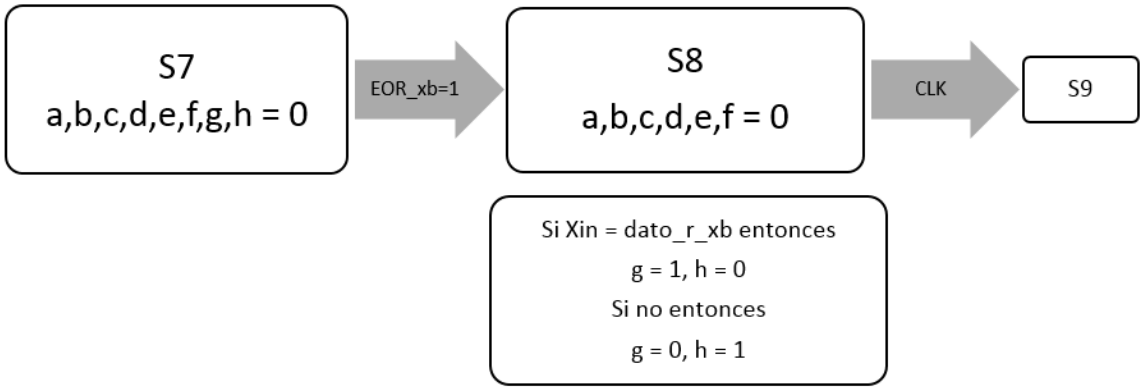


Figura 3.21: Estados 7 y 8 de la máquina de estados del FPGA receptor.

En los estados 9, 10 y 11, mostrados en la figura (3.22), se calcula una nueva iteración de la señal caótica haciendo uso de los observadores para buscar la sincronización de los sistemas y se lee la bandera de salida del contador. Al ser un contador de 8 bits, la bandera pasa de 0 a 1 luego de 255 pasos sin haber sido reiniciado, asegurando así que los sistemas estén completamente sincronizados antes

de proceder a la etapa de comunicación

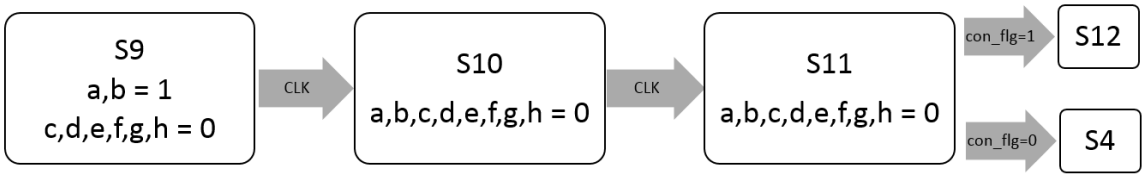


Figura 3.22: Estados 9, 10 y 11 de la máquina de estados del FPGA receptor.

En los estados 12, 13 y 14, mostrados en la figura (3.23), se envía la señal de confirmación afirmativa de sincronización al FPGA transmisor y se entra en el estado de espera de datos enmascarados o la señal de que la trasmisión se ha completado para posteriormente ir al estado de reinicio.

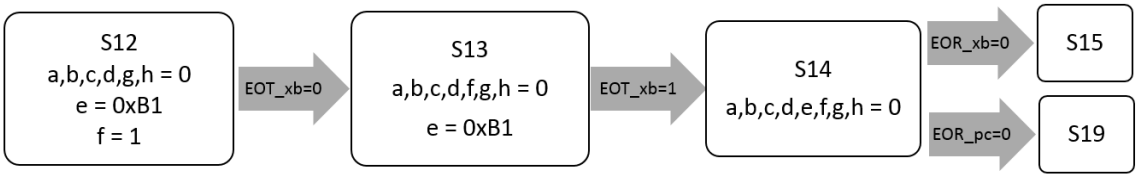


Figura 3.23: Estados 12, 13 y 14 de la máquina de estados del FPGA receptor.

En los estados 15, 16, 17 y 18, mostrados en la figura (3.24), se recibe el dato enmascarado y se desenmascara utilizando la señal caótica generada internamente sin observador, para luego transmitirlo hacia la PC y calcular la nueva iteración de la señal caótica.

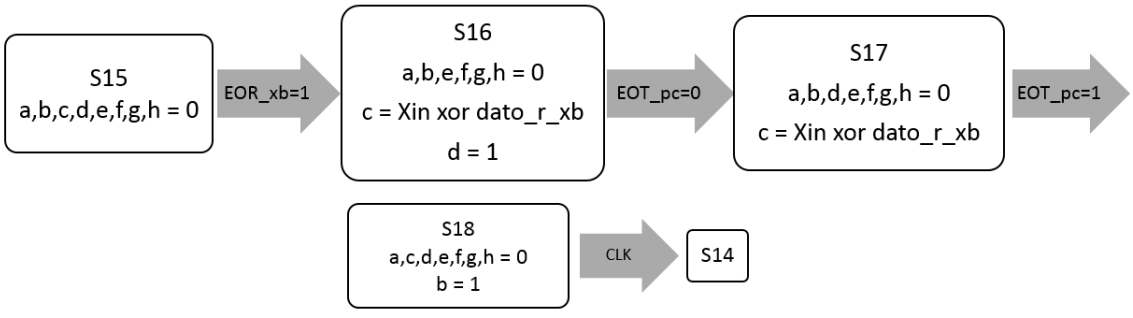


Figura 3.24: Estados 15, 16, 17 y 18 de la máquina de estados del FPGA receptor.

Cuando se recibe la señal de finalización de transmisión proveniente de la PC, se entra a los estados 19 y 20, mostrados en la figura (3.25), en donde se retransmite

esta señal hacia el FPGA transmisor y se regresa al estado inicial.

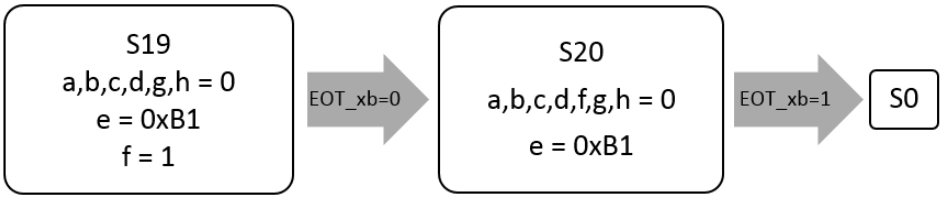


Figura 3.25: Estados 19 y 20 de la máquina de estados del FPGA receptor.

El conjunto de todos los estados mencionados corresponden a la máquina de estados del FPGA receptor, cumpliendo así el protocolo de comunicación establecido.

3.4.3. Simulaciones

En la figura (3.26) se muestra la primera simulación realizada, en esta se visualizó el comportamiento de las señales caóticas generadas por el oscilador maestro (Xm) y el oscilador esclavo (Xs), iniciando con condiciones iniciales diferentes para ver la tendencia a sincronizarse, dejando pasar un gran número de iteraciones para inspeccionar visual y matemáticamente la existencia de un comportamiento caótico. Para comprobar esto, se exportó el resultado de esta simulación al software Matlab y se compararon tanto la señal del maestro y el esclavo como la señal del maestro y la solución al sistema de ecuaciones que conforman el sistema Lorenz, encontrando una igualdad entre las señales.

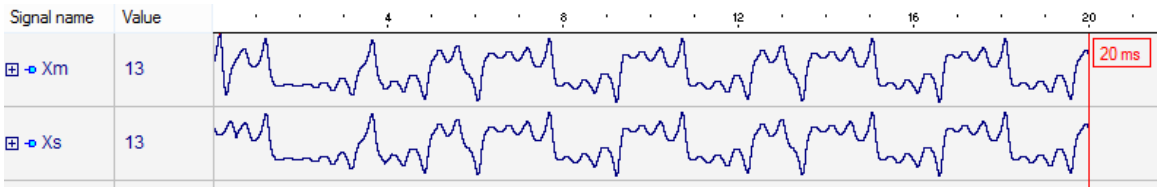


Figura 3.26: Simulación en Active de los osciladores maestro y esclavo.

En la figura (3.27) se aprecia la segunda simulación, en la cual los observadores no se agregan a las ecuaciones del oscilador esclavo hasta pasado cierto tiempo, pudiéndose apreciar que antes de esto, la señal del maestro y la señal del esclavo son

completamente diferentes, pero en cuanto este término es agregado, las señales se sincronizan rápidamente. La ejecución de este término está controlada por una bandera llamada Sync la cual se aprecia en la figura.

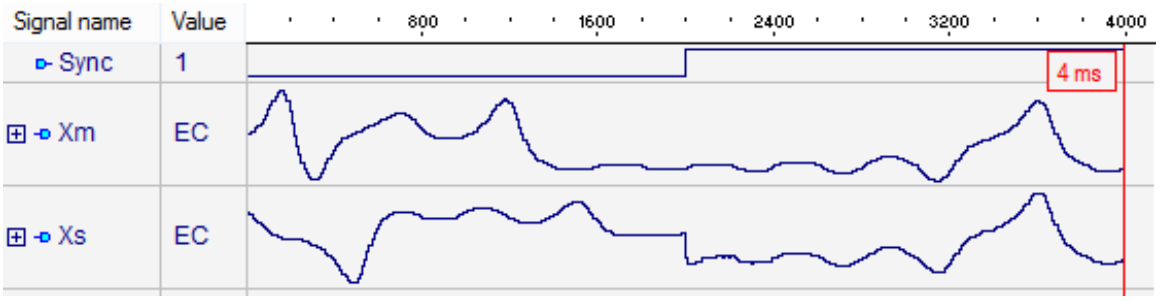


Figura 3.27: Simulación en Active de la sincronización de los osciladores maestro y esclavo activada por la bandera Sync.

3.4.4. Recursos utilizados

La tabla (3.3) muestra los recursos utilizados tanto por el FPGA maestro como el esclavo. En esta se puede notar que la cantidad de recursos utilizados es menor al 2 % en cuestión de registros y elementos lógicos y del 6 % en multiplicadores, lo cual permite integrar este sistema a futuros sistemas embebidos más complejos.

Tabla 3.3: Recursos utilizados para transmitir señales caóticas utilizando el FPGA Cyclone II (EP2C35F672C6).

Recursos	Maestro	Esclavo	Disponible
Total de elementos lógicos	761	854	33,216
Total de registros	226	117	33,216
Total de pines	6	7	475
Multiplicadores embebidos de 9 bits	4	4	70

3.5. Realización en la computadora

Como se estableció previamente en el protocolo de comunicación, para empezar el usuario debe escoger la imagen (o cualquier otro tipo de archivo) a transmitir y el dispositivo destino. Para esto, se creó una interfaz de usuario en el software LabVIEW,

mostrada en la figura (3.28). El otro parámetro que aparece, es el puerto donde está conectado el FPGA, en este caso, se utiliza un puerto serie emulado por un cable adaptador Serial-USB, conectado al FPGA gracias a la placa de desarrollo y a la computadora por el puerto USB.

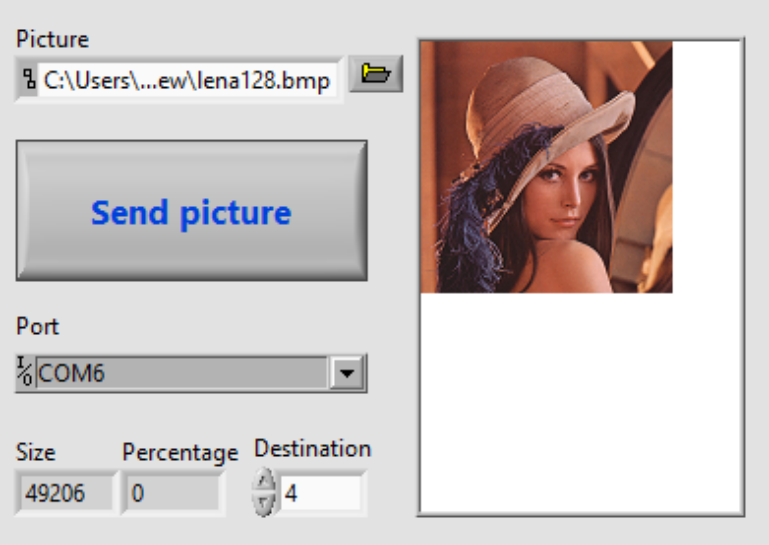


Figura 3.28: Interfaz de usuario para transmitir.

En la figura (3.29 (a)), se muestra el primer ciclo del programa, en el cual se lee la imagen seleccionada, se muestra en un cuadro de imagen y se mide su tamaño, preparando tanto el tamaño como la información en una cadena de caracteres separados por un retorno de carro para su posterior transmisión.

En la figura (3.29 (b)), se muestra el segundo ciclo, en donde se configura y abre el puerto serie para transmitir el identificador del dispositivo destino, para luego esperar la respuesta del FPGA de que se ha llevado a cabo la sincronización y se encuentra a la espera de que comience la transmisión de datos o que haya ocurrido un error al abrir el puerto serie.

En la figura (3.29 (c)), se muestra el último ciclo, en donde se procede a transmitir byte a byte la totalidad de la cadena de caracteres previamente configurada, además de hacer la división entre el índice actual y el tamaño total de la cadena para así calcular el porcentaje de progreso. Una vez finalizado este ciclo, se cierra el puerto serie y el programa regresa al primer ciclo, a la espera de la siguiente transmisión.

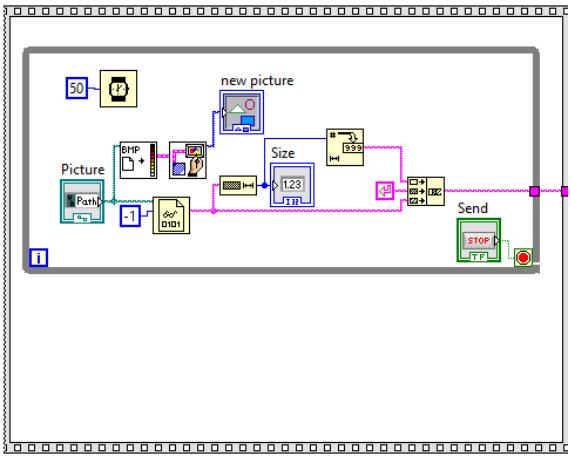
La contraparte de la interfaz anterior, es la correspondiente a la recepción de la imagen, mostrada en la figura (3.30), la cual solo incluye la selección de la dirección para almacenar el archivo recibido y el puerto donde se encuentra conectado el FPGA. Así como en el programa anterior, se utiliza un cable convertidor para comunicar la FPGA con la computadora. Como indicadores, aparecen el tamaño de la imagen, el porcentaje de progreso y la imagen recibida.

En la figura (3.31 (a)), se muestra el primer ciclo del programa, en el cual primero se configura, abre y vacía el puerto serial, haciendo esto último para evitar datos espurios. Luego, se espera a que llegue un dato para proceder al siguiente ciclo o que haya ocurrido un error, el cual cancela la ejecución de los siguientes ciclos.

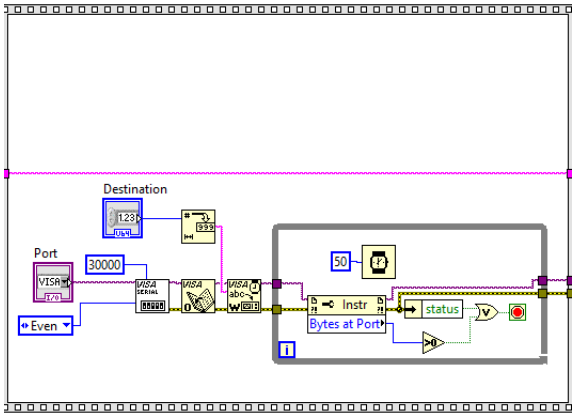
En la figura (3.31 (b)), se muestra el segundo ciclo, el cual lee la cadena de caracteres recibida byte a byte, hasta que encuentra un retorno de carro, procediendo a quitar este carácter y convertir los bytes recibidos a una constante numérica, correspondiente al tamaño del archivo a recibir.

En la figura (3.31 (c)), se muestra el tercer ciclo, el cual al igual que el anterior, lee byte a byte la cadena recibida, con la diferencia de que esta corresponde al archivo. Además, se hace la división entre la cantidad de bytes recibidos y el tamaño del archivo previamente convertido. Cuando ambos valores son iguales, el ciclo termina y da paso al último bloque.

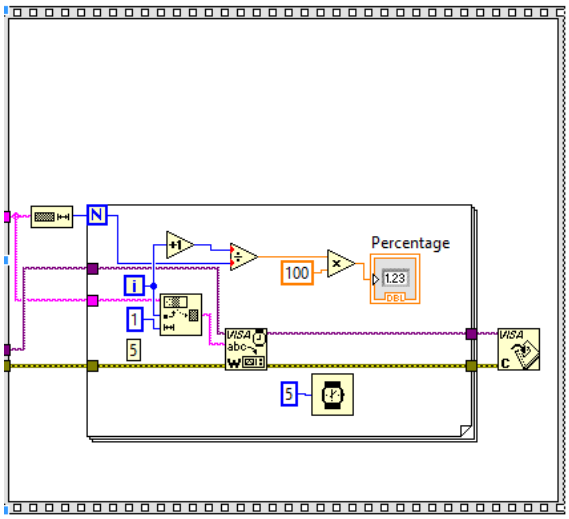
En la figura (3.31 (d)), se muestra el último bloque, el cual envía la señal de finalización al FPGA, cierra el puerto serie, almacena en la dirección previamente seleccionada la imagen y la muestra en un cuadro de imagen, finalizando así el protocolo de comunicación.



(a) Bloque de selección.



(b) Bloque de configuración Xbee.



(c) Bloque de transmisión.

Figura 3.29: Diagrama a bloques de la interfaz de usuario para la transmisión.

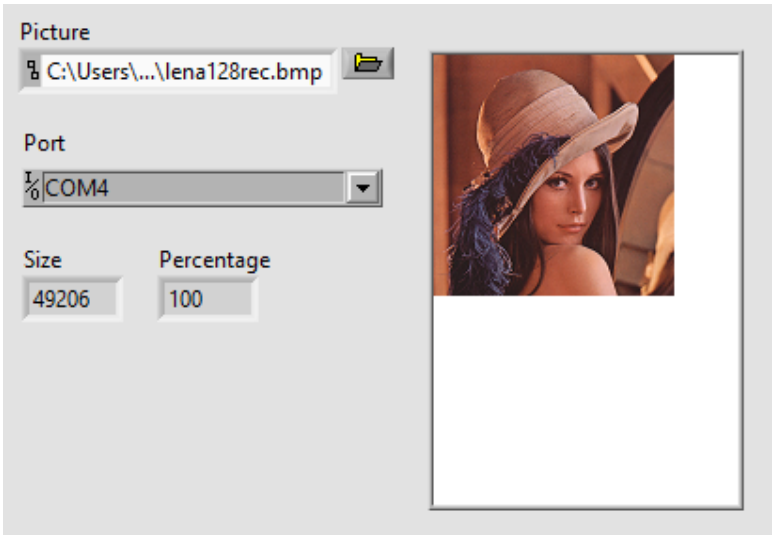
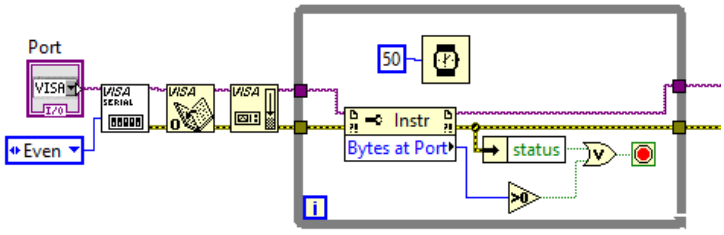
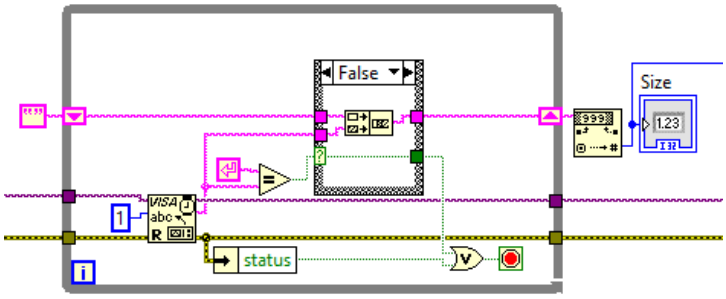


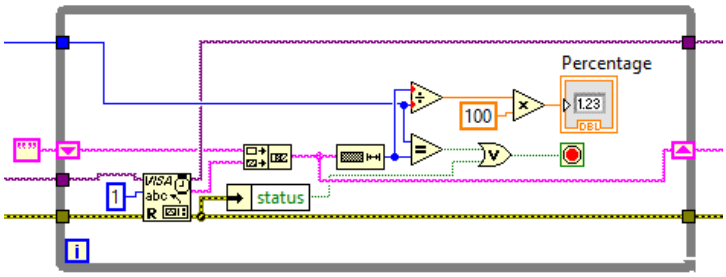
Figura 3.30: Interfaz de usuario para la recepción.



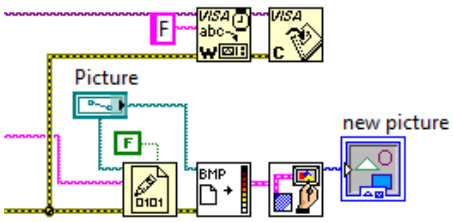
(a) Bloque de configuración y espera.



(b) Bloque de captura de tamaño de archivo



(c) Bloque de lectura.



(d) Bloque de almacenamiento y visualización.

Figura 3.31: Diagrama a bloques de la interfaz de usuario para la recepción.

Capítulo 4

Pruebas experimentales y resultado

Para demostrar la integridad y el correcto funcionamiento del sistema, se realizaron diversas pruebas bajo diferentes condiciones y se analizaron los resultados obtenidos para así poder comparar el resultado de este trabajo con otros trabajos previos.

4.1. Prueba de integridad y encriptación

Teniendo el sistema funcionando, se procedió a verificar la integridad de los datos y que tan alta es la codificación alcanzada al usar las oscilaciones caóticas. Para hacer esto, se realizó un análisis de correlación entre los datos enviados y los datos recibidos, dando como resultado 1, es decir, llegó el 100 % de los datos sin pérdidas. Este análisis se realizó usando el software Matlab, para ello se cargó en memoria la imagen transmitida y la imagen recibida, y luego se hizo uso del comando corr().

El comando corr() de Matlab calcula el coeficiente de relación lineal de Pearson, el cual es una medida de la relación lineal ente dos variables aleatorias cuantitativas. También este coeficiente se define como un índice que puede utilizarse para medir el grado de relación de dos variables [19]

En el caso de que se esté estudiando dos variables aleatorias X y Y sobre una población; el coeficiente de correlación de Pearson se simboliza con la letra $\rho_{x,y}$, siendo la expresión (4.1.1) que nos permite calcularlo.

$$\rho_{x,y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_x)(Y - \mu_y)]}{\sigma_X \sigma_Y} \tag{4.1.1}$$

Donde:

- σ_{XY} es la covarianza de (X,Y)
- σ_X es la desviación típica de la variable X
- σ_Y es la desviación típica de la variable Y

De manera análoga podemos calcular este coeficiente sobre un estadístico muestral, denotado como a r_{xy} (4.1.2).

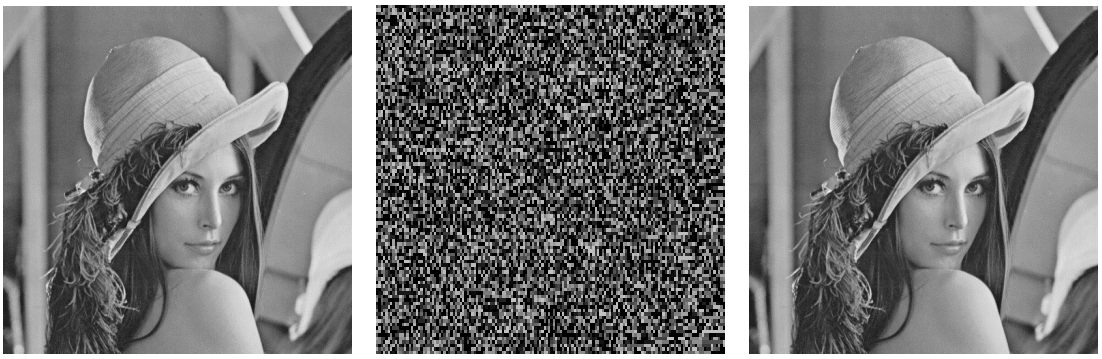
$$\mathbf{r}_{x,y} = \frac{\sum x_i y_i - n\overline{xy})}{(n-1)_{S_x S_y}} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \tag{4.1.2}$$

Para medir el nivel de encriptación alcanzado, se capturó el contenido del canal de comunicación en un archivo, el cual fue de igual forma cargado en memoria y comparado con la información original utilizando el mismo análisis estadístico de correlación. Este análisis dio como resultado un valor de correlación de 0.0042, lo que quiere decir que menos del 0.5 % de la señal original es discernible en el canal ya que el resto está sumergida en ruido caótico.

En la figura (4.1) se muestra la imagen *Lenna_gray.gif* (256x256) transmitida, el canal y la imagen recibida. Como se puede apreciar, el contenido del canal es de aspecto ruidoso y es imposible a simple vista distinguir el más mínimo detalle de la imagen original.

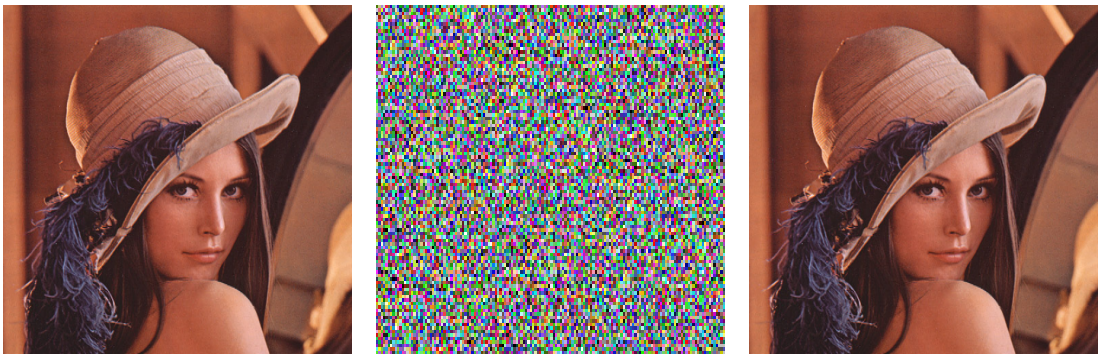
Así también en la figura (4.2) se muestra la imagen *Lenna.bmp* (256x256) a color transmitida, el canal y la imagen recibida. El contenido del canal, Al igual que en la prueba anterior, presenta aspecto ruidoso y es imposible discernir la imagen original.

Finalmente, en la figura (4.3) se muestra la imagen *Lenna.jpg* (512x512) a color, junto con el contenido del canal de transmisión, obteniendo una encriptación similar a las dos pruebas anteriores.



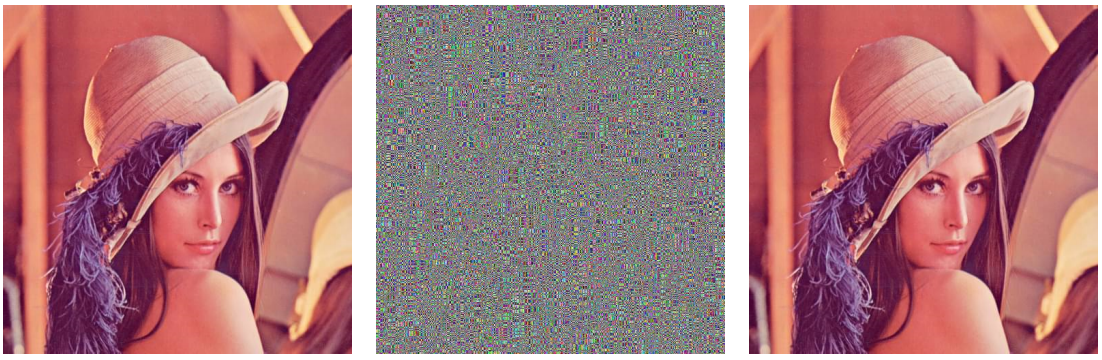
(a) Imagen transmitida (b) Canal (c) Imagen recibida

Figura 4.1: Comparación entre la imagen *Lenna_gray.gif* (256x256) transmitida, recibida y el canal.



(a) Imagen transmitida (b) Canal (c) Imagen recibida

Figura 4.2: Comparación entre la imagen *Lenna.bmp* (256x256) transmitida, recibida y el canal.



(a) Imagen transmitida (b) Canal (c) Imagen recibida

Figura 4.3: Comparación entre la imagen *Lenna.jpg* (512x512) transmitida, recibida y el canal.

Estas pruebas demuestran que independientemente del tamaño y tipo de imagen, la codificación es exitosa en cada caso.

4.2. Prueba de alcance y ruido

La siguiente prueba que se realizó fue la prueba de alcance, la cual es una prueba de distancia y sirvió para conocer el comportamiento de los módulos inalámbricos cuando estos están alejados entre sí.

La hoja de datos de los módulos Xbee [11] menciona que el rango efectivo de comunicación sin obstáculos es de 100 m, y en un ambiente con obstáculos o ruido, este rango se reduce a 30 m. Así pues, se realizó una prueba a una distancia de 50 m para corroborar esta información, transmitiendo una imagen y haciendo un análisis de correlación entre la información transmitida y recibida, obteniendo un valor de correlación igual a 1, lo que significa una integridad del 100 % de los datos.

La última prueba realizada consistió en colocar una fuente de ruido entre ambos módulos Xbee y realizar una transmisión bajo esta condición. La fuente de ruido provino de un motor eléctrico de corriente alterna colocado en la trayectoria directa entre los módulos de comunicación inalámbrica, obteniendo como resultado un valor de correlación igual a 1, por lo que se demuestra la confiabilidad de estos dispositivos.

4.3. Medición de las señales en Osciloscopio

Una manera de visualizar las señales caóticas cuando ya está implementado el sistema, es haciendo uso de un convertidor digital-analógico y un osciloscopio. En el caso de este trabajo, se utilizó un convertidor del tipo red R-2R, ya que los inconvenientes asociados a este tipo de convertidor son despreciables debido a que se están utilizando resistencias discretas de buena precisión y una baja cantidad de bits. El osciloscopio utilizado para visualizar la señal tiene una resolución de 10 bits y es capaz de leer 40M muestras por segundo.

Se implementó el sistema en dos FPGAs, modificando el código de estos para copiar el contenido de una variable de estado a un conjunto de salidas no utilizadas

normalmente. A estas salidas se les conectó de forma paralela el convertidor digital-analógico para poder así llevar a cabo las mediciones de las distintas variables de estado, mostradas en la figura (4.4).

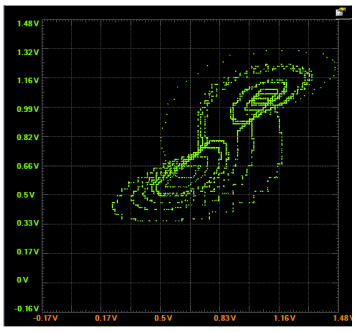
En los diagramas (a), (b) y (c), se comparan las variables de estado de un solo oscilador, en ellas se aprecia la naturaleza caótica de este y la forma en que un sistema Lorenz se comporta, es decir, los dos enrollamientos.

En cambio, en los diagramas (d), (e) y (f), se muestran los diagramas de fase entre las variables de estado X provenientes de los dos osciladores cuando estos no están sincronizados. Como se puede apreciar, no se muestra una relación definida entre estas señales.

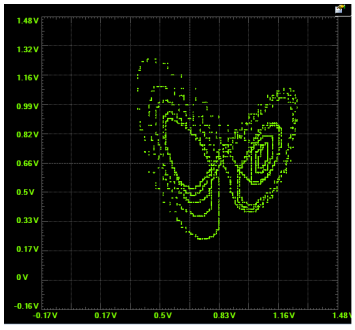
Finalmente, en los diagramas (g), (h) e (i), los osciladores se encuentran totalmente sincronizados ya que la gráfica forma una línea recta a 45°.

4.4. Resultado

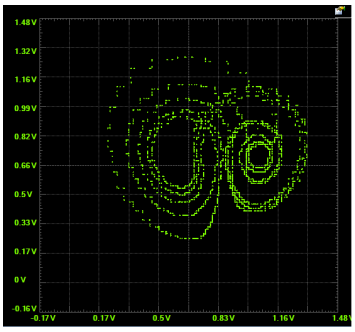
Una vez realizadas todas las pruebas individuales, se procedió a implementar una red de cuatro dispositivos del tipo estrella, es decir, un dispositivo central que administra el flujo de información y tres dispositivos receptores. Cada uno consta de un identificador distinto y cuando se desea transmitir información, solo el oscilador del dispositivo seleccionado es sincronizado con el del transmisor. La figura (4.5) muestra las fotografías del sistema completo, en las cuales se aprecian los cuatro dispositivos funcionando, cada uno de estos constando de un FPGA, un módulo inalámbrico Xbee y una interfaz de usuario.



(a) Diagrama de fase X,Y

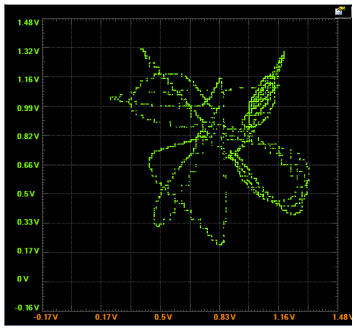


(b) Diagrama de fase X,Z

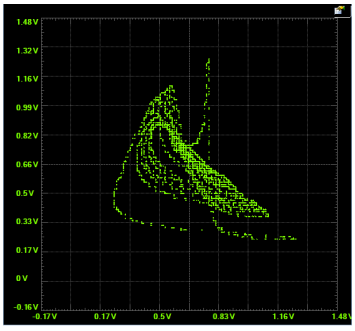


(c) Diagrama de fase Z,Y

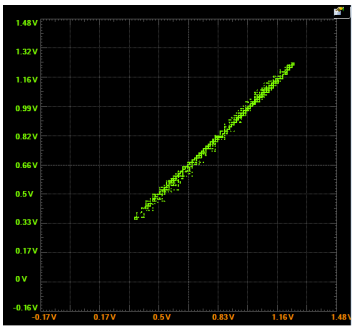
(d) Diagrama de fase X1,X2 no sincronizados



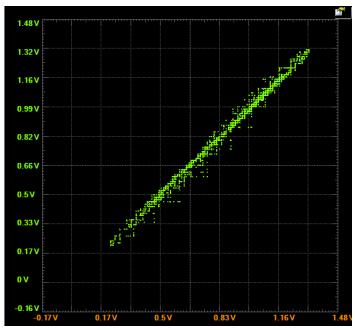
(e) Diagrama de fase Y1,Y2 no sincronizados



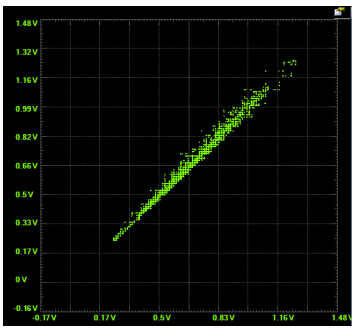
(f) Diagrama de fase Z1,Z2 no sincronizados



(g) Diagrama de fase X1,X2 sincronizados

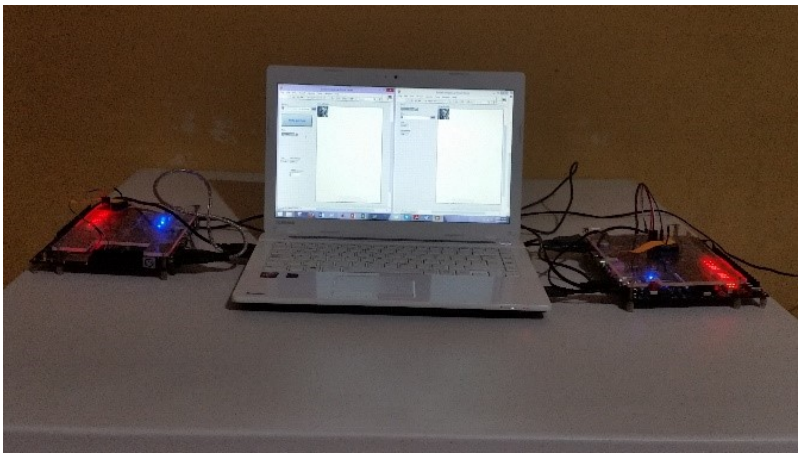


(h) Diagrama de fase Y1,Y2 sincronizados

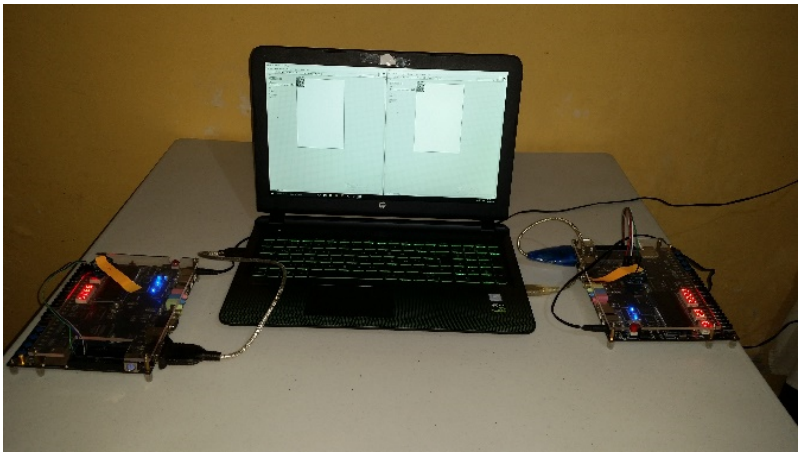


(i) Diagrama de fase Z1,Z2 sincronizados

Figura 4.4: Mediciones realizadas con el osciloscopio.



(a) Fotografía 1



(b) Fotografía 2

Figura 4.5: Fotografía del sistema completo.

Capítulo 5
Conclusiones

En este trabajo se presentó la realización de una comunicación de alta confidencialidad, por medio de FPGAs con módulos inalámbricos, basándose en osciladores caóticos sincronizados para realizar un enmascaramiento de los datos. Los osciladores inician no sincronizados y se utilizó el método de observadores para llevar a cabo una rápida sincronización.

Implementar el método de integración Forward Euler en VHDL dio como resultado un bajo consumo de recursos en comparación al método de integración Runge-Kutta, permitiendo al sistema ser expandido a otros sistemas de oscilaciones caóticas si así se desease.

Se diseñaron máquinas de estados para los dos sistemas desarrollados, transmisor y receptor, las cuales gobiernan el proceso de comunicación que se planteó inicialmente. Gracias a estas, se obtuvo una comunicación exitosa, la cual engloba el enmascaramiento, transmisión, recepción y desenmascaramiento de los datos.

Al realizar las pruebas de correlación entre el canal de comunicación y los datos transmitidos, se encontró que el canal no corresponde casi en absoluto a la información que se está transmitiendo. Además, inspeccionando visualmente estos dos conjuntos es imposible distinguir la más mínima relación.

Así mismo, se desarrolló una interfaz de usuario que facilita el uso del sistema desarrollado en FPGA, permitiendo seleccionar rápidamente la imagen a ser transmitida y el dispositivo al cual transmitírsela.

Para asegurar que el sistema es confiable bajo condiciones adversas, se llevaron a cabo las pruebas de distancia y ruido, obteniendo como resultado una integridad del 100 % de los datos, demostrando así que los módulos inalámbricos XBee son apropiados para esta aplicación. Si se quisiera tener un mayor alcance o un sistema más complejo, se podría fácilmente utilizar otro módulo de comunicación o una red más grande de dispositivos.

Finalmente, basándonos en la imagen del canal y las pruebas realizadas al sistema se demuestra que el protocolo de comunicación de datos es seguro y confiable. Este sistema puede ser implementado para diferentes aplicaciones de comunicaciones, proporcionando un método de transmitir cualquier tipo de información. Además, se observó que al utilizar un oscilador del tipo sistema de Lorenz, la sincronización es bastante rápida. Los recursos que necesita cada FPGA son mínimos, comparados con el total disponible, pudiendo así agregar más funcionalidad a futuro.

Capítulo 6

Trabajo a futuro

Como trabajo a futuro se propone:

1.- Implementar en VHDL la integración numérica de las ecuaciones diferenciales que componen a otros tipos de sistemas caóticos de dos o más enrollamientos, tales como el sistema Rössler, circuito de Chua, funciones saturadas, entre otras.

2.- Utilizar otros métodos de integración numérica para encontrar la solución de los sistemas de ecuaciones, por ejemplo Runge-Kutta, método trapezoidal, método de Simpson, etc.

3.- Diseñar un sistema en donde un solo FPGA pueda ser maestro y esclavo, pudiendo implementar otras topologías de red.

Bibliografía

[1] D Chillingworth. Chaos: An introduction to dynamical systems (kathleen t. alligood, tim d. sauer, and james a. yorke). *SIAM REVIEW*, 40:732–732, 1998.

[2] José Pastor and Miguel Ángel Sarasa López. *Criptografía digital: fundamentos y aplicaciones*. Publicaciones Universitarias Universidad de Zaragoza, 1998.

[3] Kevin M Cuomo, Alan V Oppenheim, and Steven H Strogatz. Synchronization of lorenz-based chaotic circuits with applications to communications. *IEEE Transactions on circuits and systems II: Analog and digital signal processing*, 40(10):626–633, 1993.

[4] Herve Dedieu, Michael Peter Kennedy, and Martin Hasler. Chaos shift keying: modulation and demodulation of a chaotic carrier using self-synchronizing chua's circuits. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(10):634–642, 1993.

[5] V Milanovic and ME Zaghloul. Improved masking algorithm for chaotic communications systems. *Electronics Letters*, 32(1):11–12, 1996.

[6] MS Azzaz, C Tanougast, S Sadoudi, and A Dandache. Real-time fpga implementation of lorenz's chaotic generator for ciphering telecommunications. In *Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA'09. Joint IEEE North-East Workshop on*, pages 1–4. IEEE, 2009.

[7] Yuhua Xu, Bing Li, Yuling Wang, Wuneng Zhou, and Jian-an Fang. A new four-scroll chaotic attractor consisted of two-scroll transient chaotic and two-scroll ultimate chaotic. *Mathematical Problems in Engineering*, 2012, 2012.

[8] Roger Chiu, Juan H Garcia-Lopez, Rider Jaimes-Reategui, Edgar Villafaña-Rauda, Carlos E Castañeda-Hernandez, Guillermo Huertacuellar, and Didier Lopez-Mancilla. Implementation of chaotic systems for secure communications in embedded dsp system.

[9] Yong-fang Zhang Xin-Guo Zhang Parag Gupta Li Xiong, Yan-Jun Lu. Design and hardware implementation of a new chaotic secure communication technique.

[10] E Tlelo-Cuautle, VH Carbajal-Gomez, PJ Obeso-Rodelo, JJ Rangel-Magdaleno, and Jose Cruz Nuñez-Perez. Fpga realization of a chaotic communication system applied to image processing. *Nonlinear Dynamics*, 82(4):1879–1892, 2015.

[11] Oyarce Andrés. Guía de usuario xbee series 1. 2010.

[12] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[13] Otto E Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[14] T Matsumoto. A chaotic attractor from chua's circuit. *IEEE Transactions on Circuits and Systems*, 31(12):1055–1058, 1984.

[15] Michael Peter Kennedy. Robust op amp realization of chua's circuit. *FREQUENZ.*, 46(3):66–80, 1992.

[16] Chai Wah Wu. *Synchronization in coupled chaotic circuits and systems*, volume 41. World Scientific, 2002.

[17] Hebertt Sira-Ramirez and César Cruz-Hernández. Synchronization of chaotic systems: A generalized hamiltonian systems approach. *International Journal of bifurcation and chaos*, 11(05):1381–1395, 2001.

[18] Lawrence F Shampine, Richard C Allen, and Steven Pruess. *Fundamentals of numerical computing*, volume 1. Wiley New York, 1997.

[19] Harald Cramér. *Mathematical Methods of Statistics (PMS-9)*, volume 9. Princeton university press, 2016.