BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Electrónica

Título de la tesis:

DISEÑO ANALÓGICO EMBEBIDO DE UN GENERADOR DE NÚMEROS ALEATORIOS USANDO SISTEMAS HIPERCAÓTICOS.

T E S I S

QUE PARA OBTENER EL TITULO DE:

Lic. en Ingeniería en Mecatrónica

PRESENTA:

Diego Ruiz Sánchez de la Vega

ASESOR:

Dr. Jesús Manuel Muñoz Pacheco

JUNIO, 2023

Agradecimientos

Me gustaría agradecer de manera especial a mi familia, es por ellos que debo la fuerza poder concluir con mis estudios y con la realización de esta tesis. En especial quiero agradecer mi madre Gloria Sanchez de la Vega Escalante, gracias a ella, por su esmero y dedicación, así como entrega y sacrificio. A ellos dedico este trabajo.

También agradezco en gran medida el apoyo y asesoría del Dr. Jesús Manuel Muñoz Pacheco, su conocimiento y paciencia hizo posible la realización de este trabajo. Así mismo, agradezco al jurado de tesis por sus valiosas observaciones que contribuyeron a mejorar este trabajo.

Así como también agradezco a todos mis compañeros de la facultad con quienes compartí trabajo y esfuerzo, me permitieron convertirme en mejor estudiante y persona.

Los autores agradecen al Laboratorio Nacional de Supercómputo del Sureste de México perteneciente a la red de laboratorios nacionales CONACYT, por los recursos computacionales, el apoyo y la asistencia técnica.

Resumen

El presente trabajo de tesis tiene como objetivo implementar un generador de números aleatorios basado en sistemas caóticos e hipercaóticos utilizando Field-Programmable Analog Arrays (FPAA's). Se realiza una investigación exhaustiva en cuanto a los generadores de números aleatorios basados en sistemas caóticos y se determina que los sistemas caóticos son una fuente adecuada de entropía para generar números aleatorios de alta calidad. Por otro lado, los FPAA's son una herramienta útil para implementar sistemas caóticos debido a su flexibilidad y capacidad de reconfiguración en tiempo real.

En este trabajo de tesis se utiliza el sistema hipercaótico de Lorenz y el sistema caótico de Chua como generadores de entropía en el modelo propuesto para el generador de numeros aleatorios. Mediante simulaciones numéricas de los sistemas caóticos e hipercaóticos en Matlab/Simulink, la secuencia de bits obtenida fue analizada por las pruebas estadísticas de la NIST.

Finalmente, se implementa el TRNG utilizando hardware analógico embebido por medio de la tarjeta “Anadigm SingleApex Development Board” (ASDB). El generador de números aleatorios proporciona una secuencia de numeros aleatorios, aprobando exitosamente las pruebas estadísticas NIST-FIPS descritas a detalle en el presente trabajo.

En conclusión, se demostró que los sistemas caóticos son una fuente adecuada de entropía para la generación de números aleatorios de alta calidad y que los FPAA's son una herramienta útil para la implementación de sistemas caóticos debido a su adaptabilidad y sencilla reconfiguración, permitiendo al diseñador un análisis en tiempo real. La implementación de los sistemas caótico e hipercaótico y el generador de números aleatorios en el FPAA demuestran su viabilidad. Se sugiere que futuros trabajos que estén centrados en la implementación de otros sistemas caóticos o hipercaóticos evalúen la posibilidad de ser implementados en FPAA’s.

Lista de imágenes

Nomenclatura

ASDB	Anadigm SingleApex Development Board
FIPS	Estándar de procesamiento de información federal.
NIST	Instituto Nacional de Estándares y tecnología.
PRNG	Generador de numeros pseudoaleatorios.
RNG	Generador de numeros aleatorios.
SCC	Sistema caótico de Chua.
SCCE	Sistema caótico de Chua escalado.
SED	Sistemas de ecuaciones diferenciales.
SHL	Sistema hipercaótico de Lorenz.
SHLE	Sistema hipercaótico de Lorenz escalado.
TRNG	Generador de numeros aleatorios verdadero.

Lista de tablas

Índice

Capítulo 1

Introducción

1. Introducción

La generación de números aleatorios es una tarea fundamental en muchas áreas de la estadística [1], las matemáticas [2] y la simulación numérica [3], en general cualquier área en la cual se requiera generar una secuencia de numeros los cuales no presenten alguna correlación o patrón discernible. Podemos clasificar a los generadores de numeros aleatorios en dos tipos, los generadores de numeros pseudoaleatorios (PRNG) y los generadores de numeros aleatorios verdaderos (TRNG), el primero es algoritmo ejecutado por un programa el cual genera una secuencia de numeros aleatorios los cuales aparentan ser aleatorios, sin embargo, esta es una secuencia determinista y dependiente de un valor inicial llamado "seed", el segundo utiliza un proceso físico por el cual son capaces de generar una secuencia de numeros aleatorios, estos numeros son considerados como verdaderamente aleatorios debido a la inherente impredecibilidad del proceso físico [4].

La mayoría de los generadores de números aleatorios son PRNG debido a su sencilles en la elaboración e implementación, además en la práctica suelen ser más rápidos, pero presentan algunas limitaciones importantes. En particular, pueden generar secuencias de números que son periódicas o que no tienen una distribución uniforme. Además, estas secuencias pueden ser predecibles si se conoce suficiente información sobre el algoritmo y el estado inicial, lo cual es perjudicial principalmente en aplicaciones de los RNG relacionadas a la seguridad.

Existen distintos modelos para la elaboración de TRNG's para los cuales se utilizan distintos comportamientos de la naturaleza con comportamiento impredecible, algunos ejemplos de estos modelos pueden ser en base al ruido [24], en base al comportamiento aleatorio intrínseco de la mecánica cuántica [25], en base a la extrema sensibilidad de los sistemas caóticos, entre otros.

En los últimos años, se ha prestado cada vez más atención al uso de sistemas caóticos para la generación de números aleatorios, algunos ejemplos pueden ser los trabajos realizados por Sezgin Kacar [6] o por Fatih Ozkaynak [7], en los cuales se utilizan las propiedades exhibidas por los sistemas dinámicos caóticos.

1.1 Sistemas dinámicos.

Los sistemas dinámicos son un área de estudio en la matemática y la física que se enfoca en el análisis de sistemas que cambian con el tiempo. El estudio de los sistemas dinámicos se remonta a Isaac Newton, quien utilizó las ecuaciones diferenciales para describir el movimiento de los planetas alrededor del sol debido a la atracción gravitacional que existe entre ellos [9]. Desde entonces, el estudio de los sistemas dinámicos ha evolucionado y se ha aplicado a una amplia variedad de disciplinas, como la biología, la química, la economía, la ingeniería o en general cualquier disciplina en la cual ocurra una evolución.

La evolución de los sistemas dinámicos no solo se reduce a los temas en los que se relaciona, sino también a las herramientas y técnicas para poder analizar y resolver las ecuaciones diferenciales que a estos sistemas describen, entre los cuales se incluyen, transformadas de Laplace, soluciones por series de potencias, variación de parámetros, métodos algebraicos, entre otros [8]. El nacimiento de las tecnologías computacionales en el siglo pasado ha permitido a los investigadores poder hacer avances en el análisis de sistemas dinámicos más complejos, esto debido al poder de la simulación que estas nuevas tecnologías ofrecen.

La gran parte de los sistemas dinámicos estudiados son representados por medio de un finito sistema de ecuaciones diferenciales o SED por sus siglas, las cuales pueden ser lineales o no lineales

$$\begin{aligned} \dot{x_1} &= f_1(t, x_1, x_2, \dots, x_n, u_1, \dots, u_p) \\ \dot{x_2} &= f_2(t, x_1, x_2, \dots, x_n, u_1, \dots, u_p) \\ &\vdots \\ \dot{x_n} &= f_n(t, x_1, x_2, \dots, x_n, u_1, \dots, u_p) \end{aligned}$$

Donde $\dot{x_i}$ denota la derivada de x_i con respecto a la variable *t,* y $u_1, \dots, u_p$ son las variables de entrada. Con regularidad el SED de primer orden se expresa de manera reducida por medio de vectores definidos como

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \end{bmatrix}, \qquad \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} f_1(t, \boldsymbol{x}, \boldsymbol{u}) \\ f_2(t, \boldsymbol{x}, \boldsymbol{u}) \\ \vdots \\ f_n(t, \boldsymbol{x}, \boldsymbol{u}) \end{bmatrix}$$

por lo tanto, podemos reescribir el SED como

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{1.1}$$

En un sistema dinámico, el presente estado está completamente determinado por los estados previos del sistema [8].

La ecuación (1.1) es llamada la *ecuación de estados o espacio de estados*, se refiere a $\boldsymbol{x}$ como el estado y a $\boldsymbol{u}$ como la entrada [10]. En algunas ocasiones se asocia la ecuación (1.1) con otra ecuación

$$\boldsymbol{y} = \boldsymbol{h}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{1.2}$$

la cual define un vector *n*-dimensional de salida. Existen casos especiales en los cuales el sistema no depende explícitamente ni de la entrada ni del tiempo, es decir,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{1.3}$$

en cuyo caso al sistema se le denomina como autónomo o invariante en el tiempo.

Existen conceptos de gran relevancia en el estudio de los sistemas dinámicos, uno de estos es el concepto del *punto de equilibrio*. Un punto $\boldsymbol{x} = \boldsymbol{x}^*$ en el espacio de estados se dice que es un punto de equilibro si siempre que el sistema inicie en el estado $\boldsymbol{x}^*$ entonces el sistema se mantendrá en el estado $\boldsymbol{x}^*$ para cualquier estado siguiente. Para un sistema autónomo (1.3) se puede hallar los puntos de equilibrio del sistema hallando las raíces de la ecuación [10]

$$f(\boldsymbol{x}) = 0 \tag{1.4}$$

1.2 Sistemas lineales y no lineales.

Un sistema lineal es aquel que satisface la propiedad de superposición, lo que significa que su respuesta a una combinación lineal de entradas es igual a la suma de las respuestas

a cada entrada individual [11]. Matemáticamente, un sistema lineal se puede expresar mediante una ecuación diferencial lineal

$$a_n(t)\frac{d^n y}{dx^n} + a_{n-1}(t)\frac{d^{n-1} y}{dx^{n-1}} + \cdots + a_0 y(t) = b_m(t)\frac{d^m x}{dt^m} + b_{m-1}(t)\frac{d^{m-1} x}{dt^{m-1}} + \cdots + b_0 x(t) \qquad 1.5$$

donde $y(t)$ representa la respuesta del sistema, $x(t)$ es la entrada, y las funciones $a_i(t)$ y $b_i(t)$ son coeficientes que pueden depender o no del tiempo "t" [12].

Al contrario que los sistemas lineales, los sistemas no lineales son aquellos que no satisfacen la propiedad de superposición, lo que significa que su respuesta a una combinación no lineal de entradas no es igual a la suma de las respuestas a cada entrada individual [13]. Matemáticamente, un sistema no lineal se puede expresar mediante una ecuación diferencial no lineal que no puede ser escrita en la forma estándar de una ecuación diferencial lineal (1.5).

1.3 Sistemas caóticos.

Existen muchas posibles definiciones para el caos, en realidad, no existe un consenso general dentro de la comunidad científica para saber qué es exactamente un sistema caótico dinámico [8], sin embargo, si existe un consenso de que características un sistema caótico debe contener para ser considerado como tal:

- Deterministas y no lineales.
- Densidad de orbitas periódicas.
- Extrema sensibilidad a las condiciones iniciales.

Un sistema determinista si su evolución futura está completamente determinada por sus condiciones iniciales y por las leyes matemáticas que describen su comportamiento. Esto significa que, si conocemos con precisión el estado del sistema en un momento dado y las ecuaciones que rigen su evolución, podemos predecir su comportamiento futuro con absoluta certeza [13].

En un sistema determinista, cualquier perturbación o variación en las condiciones iniciales tendrá un efecto predecible en la evolución futura del sistema. En otras palabras,

pequeños cambios en las condiciones iniciales pueden tener un efecto significativo en la evolución del sistema a largo plazo [13], esta característica de los sistemas deterministas es llevado al extremo por los sistemas caóticos en donde la más mínima variación en las condiciones iniciales tiene cambios significativos en la evolución del sistema.

El estudio de los sistemas caóticos comenzó con el trabajo pionero de Henri Poincaré en la teoría de las órbitas planetarias. En 1961, Edward Lorenz descubrió la primera ecuación caótica en el contexto de la meteorología, al observar que pequeñas variaciones en los datos iniciales podían llevar a grandes diferencias en las predicciones del clima [16], tras esto comparo los resultados con las observaciones dadas por Henry Poincaré y concluyo que este comportamiento es caótico por naturaleza. Tiempo después repitió el mismo experimento con mínimas variaciones en las condiciones iniciales y con los cuales valido sus observaciones. Estos resultados fueron publicados en 1963 [18] junto con el SED no lineales, las cuales ahora son conocidas como el sistema de Lorenz

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \sigma(y-x) \\ -xz + \gamma x - y \\ xy - \beta z \end{bmatrix} \tag{1.6}$$

donde las variables "x", "y" y "z" son las variables de estado, el espacio de estados es controlado por los parámetros σ, β y γ. La figura 1.1 muestra la gráfica de la primera variable de estado "x" con respecto al tiempo, la figura 1.2 la gráfica de la variable de estado "y" con respecto a "x" la llamada “mariposa de Lorenz”, obtenidas tras la simulación en Matlab del sistema de Lorenz con los valores $\sigma = 10, \beta = 8/3$ y $\gamma = 28$ y condiciones iniciales $x_0 = 1, y_0 = 0$ y $z_0 = 0$.

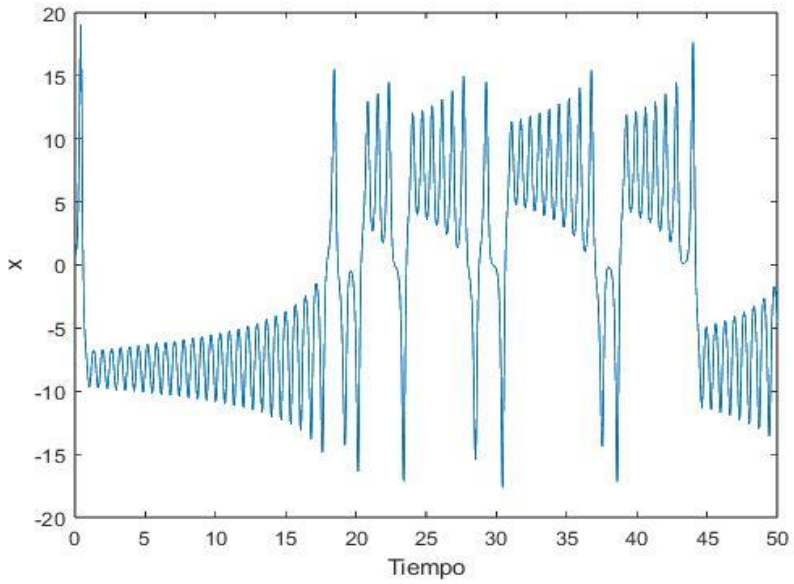


Figura 1.1. Variable de estado "x" con respecto al tiempo del sistema Lorenz.

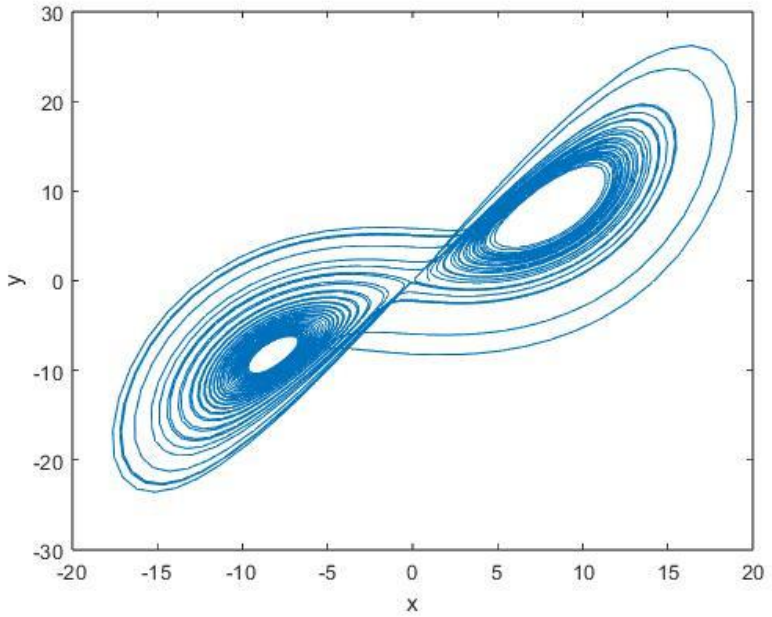


Figura 1.2. Variable de estado "x" con respecto a la variable de estado "y" del sistema Lorenz.

Desde entonces, se ha encontrado que los sistemas caóticos son comunes en una amplia variedad de disciplinas científicas, desde la física [14] y la biología [17] hasta la economía [15] y las ciencias sociales [1].

Los sistemas caóticos pueden modelarse matemáticamente utilizando ecuaciones diferenciales no lineales (1.6). Estas ecuaciones se caracterizan por tener términos no lineales que dan lugar a comportamientos no periódicos e impredecibles. Los sistemas

caóticos también suelen tener atractores extraños, que son conjuntos de puntos que atraen las trayectorias del sistema, pero que no son puntos fijos o periódicos.

El estudio de los sistemas caóticos ha llevado a la comprensión de una serie de fenómenos interesantes, como la multi-estabilidad [8], que se observa en muchos sistemas caóticos, y la sensibilidad a las condiciones iniciales, que ha sido utilizada en la criptografía para generar números aleatorios [1].

1.4 Matriz de bloques de circuitos analógicos programables

En muchas ocasiones cuando se está implementando un sistema electrónico es necesario ajustar o rediseñar el sistema propuesto, si bien existen dispositivos embebidos capaces de ser reprogramables y facilitar este ajuste o rediseño por medio de su programación, estos suelen ser dispositivos digital, como lo son los FPGA's o los microcontroladores, cuando se habla de dispositivos analógicos un ajuste suele ser el cambio de los valores en los capacitores, resistencias o en general algún dispositivo electrónico activo, un rediseño de un circuito analógico podría implicar un cambio de una gran parte del circuito. En general estas acciones de ajuste o rediseño suele ser una tarea laboriosa en lo que a circuitos analógicos se refiere, tomando en cuenta que los sistemas caóticos son sistemas autónomos no lineales su diseño es más complejo, es por esto por lo que la utilización de un hardware embebido analógico toma una gran importancia para el desarrollo de este trabajo.

Los FPAA's (Field-Programmable Analog Arrays) son dispositivos electrónicos programables que permiten la creación de circuitos analógicos personalizados [21]. Son similares a los FPGA's (Field-Programmable Gate Arrays) pero en lugar de permitir la programación de circuitos digitales, los FPAA's permiten la programación de circuitos analógicos y como tal nos permiten realizar las mismas funciones que con los

dispositivos analógicos podemos hacer, como lo son la integración, derivación, suma y resta, comparación, amplificación, entre otros. Gracias a estas características los FPAA's son adecuados para la implementación de circuitos analógicos complejos como lo pueden ser los sistemas caóticos.

Estos sistemas analógicos reconfigurables están ganando una relevancia significante en todas las aplicaciones de la industria de semiconductores. Los FPAA's son excelentes para el diseño de circuitos analógicos que requieren el procesamiento analógico en tiempo real como lo pueden ser las comunicaciones, control de procesos, sistemas biomédicos, entre otros; además, debió de su reconfigurabilidad se genera una importante reducción de costos y del tiempo de desarrollo sobre el diseño de circuitos analógicos convencionales.

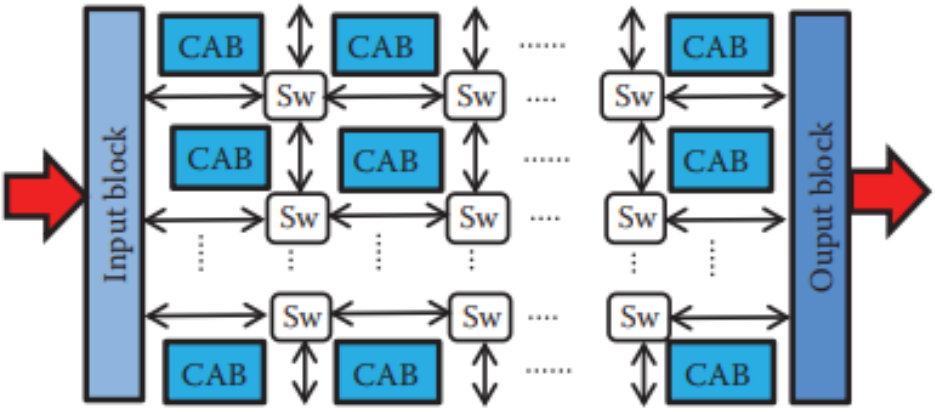


Figura 1.3. Arquitectura general de un FPAA [19].

El modelo del FPAA AN231E04 que se utilizara en esta tesis consiste en una matriz de 2x2 bloques analógicos configurables (CAB's), los cuales están conectados con recursos programables interconectados y 7 celdas de entradas y/o salidas analógicas con elementos activos [21]. La inclusión de una "look.up table" de 8x256 bits permite la síntesis de formas de onda y el manejo de varias funciones no lineales más complejas, en la figura 1.4 se observa la arquitectura de un FPAA AN231E04.

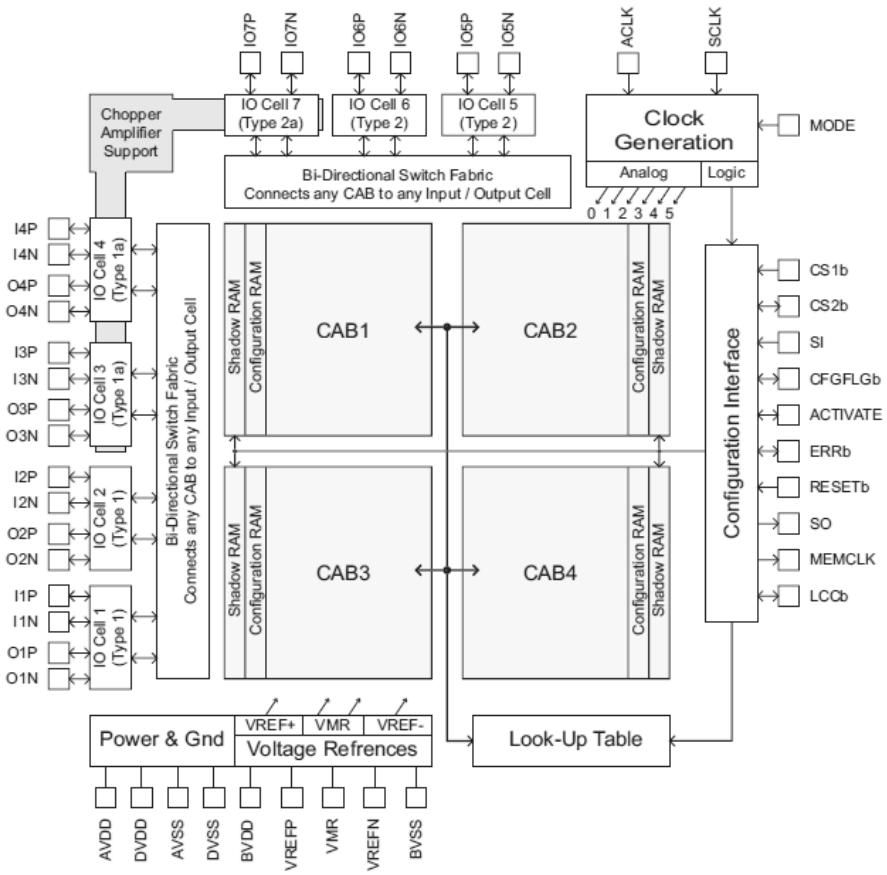


Figura 1.4. Arquitectura de un FPAA AN231E04 (Figura obtenida de la hoja de datos del dispositivo).

El software ofrecido por el desarrollador y distribuidor del AN231E04 llamado AnadigmDesigner2 proporciona un entorno de diseño gráfico con el cual podemos diseñar y simular circuitos analógicos, además con él se programa la tarjeta Anadigm SingleApex Development Board, la cual en este trabajo abreviaremos como ANDB, la cual contine 4 FPAA's, un PIC32 y otros dispositivos que nos permiten crear sistemas embebidos. El ambiente de trabajo de AnadigmDesigner2 (figura 1.5) consiste en bloques los cuales representan el FPAA sobre el cual se programará y dentro de este se pueden implementar CAM's, que por sus siglas en ingles significan Módulos Analógicos Configurables, cada uno de estos tiene una función de transferencia equivalente la cual se encuentra en la documentación dada por el desarrollador.

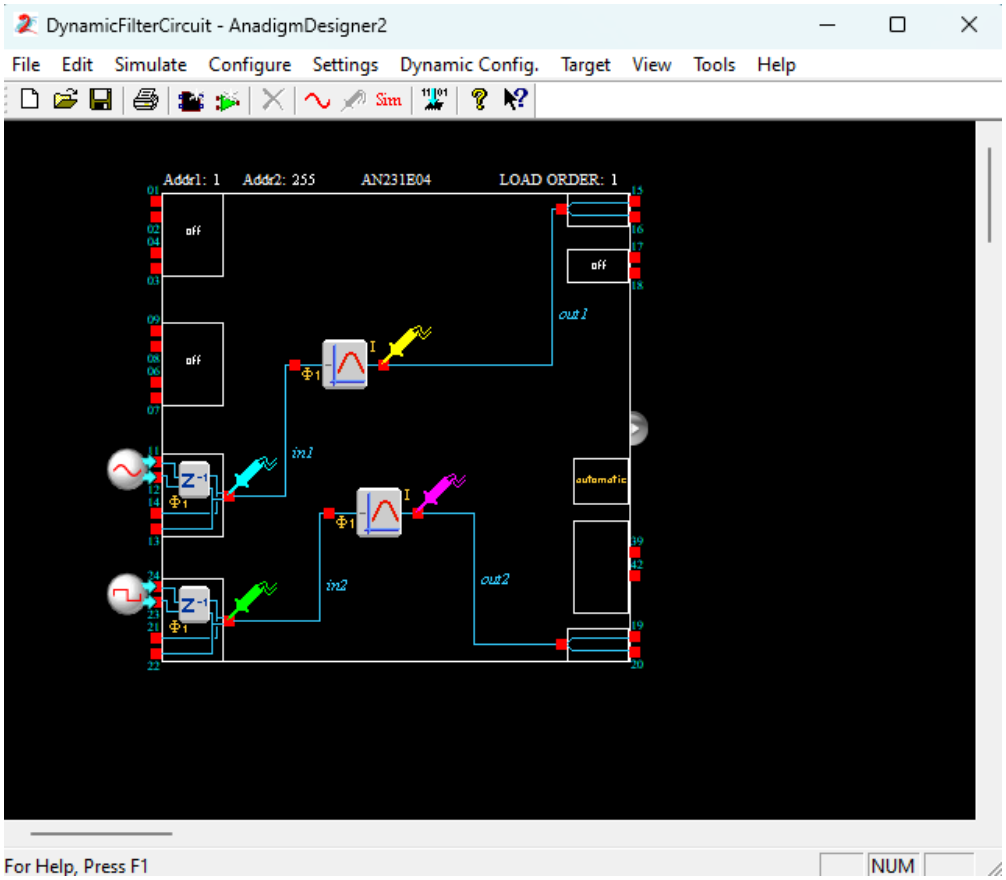


Figura 1.5. Ambiente de programación del software AnadigmDesigner2.

1.5 Generadores de números aleatorios

Los generadores de números aleatorios (RNG, por sus siglas en inglés) son herramientas fundamentales en la computación y la estadística, utilizados en una gran variedad de aplicaciones, desde juegos de azar hasta simulaciones numéricas complejas. Sin embargo, generar números verdaderamente aleatorios es un desafío, y muchos RNG pueden presentar patrones no deseados o predecibles, lo que puede comprometer la seguridad o la precisión de los resultados [23].

Existen dos tipos principales de RNG: los generadores de números pseudoaleatorios (PRNG) y los generadores de números verdaderamente aleatorios (TRNG). Los PRNG se basan en algoritmos matemáticos que generan secuencias de números aparentemente aleatorios, pero la exacta implementación de estos conceptos en las computadoras

convencionales es imposible [22], en realidad, estos numeros aleatorios son algoritmos determinísticos, es decir, son completamente predecibles conocidas las condiciones iniciales; entre algunos métodos que existen para desarrollar PRNG están los métodos basados en congruencias lineales, en desplazamiento y rotación, y en algoritmos criptográficos, siendo estos últimos los preferidos en criptografía. Por otro lado, los TRNG se basan en fuentes físicas de aleatoriedad, como el ruido térmico, la radiación cósmica, sistemas caóticos, que producen valores que son verdaderamente aleatorios e impredecibles.

Aunque los PRNG son más comunes y fáciles de implementar, pueden presentar limitaciones en algunas aplicaciones críticas de seguridad, como en criptografía, donde la generación de claves secretas y la autenticación de mensajes dependen de números aleatorios impredecibles. En este sentido, los TRNG son la opción preferida, ya que garantizan la aleatoriedad de los números generados.

Es importante tener en cuenta que incluso los TRNG pueden presentar limitaciones, como la posibilidad de que la fuente de aleatoriedad sea influenciada por factores externos o la posibilidad de que se presenten patrones en la secuencia generada. Por esta razón, se utilizan técnicas de post-procesamiento, como lo pueden hacer el hashing o el llamado filtro de Von-Neumann, y una evaluación estadística para garantizar la calidad de los números generados, en el caso de esta tesis la evaluación será llevada a cabo por las pruebas estadísticas NIST.

1.6 Pruebas estadísticas NIST

El Instituto de Estándares y Tecnología (NIST), es una agencia no reguladora que promueve la innovación mediante el fomento de la ciencia, los estándares y la tecnología de la medición. El NIST Test Suite es un paquete estadístico que consta de 15 pruebas las cuales fueron desarrolladas para probar a aleatoriedad de secuencias binarias (de tamaño arbitrario) producidas por hardware o software generador de números aleatorios o pseudoaleatorios. Estas pruebas se enfocan en una variedad de diferentes tipos de no aleatoriedad que podrían existir en una secuencia.

Algunas de estas pruebas desarrolladas por NIST fueron evaluadas y aprobadas por los Estándares Federales de Procesamiento de la Información (FIPS), tomando esto en cuenta el modelo del TRNG desarrollado en esta tesis utilizara las siguientes pruebas:

- Prueba de frecuencia del Monobit.
- Prueba de frecuencia en bloque.
- Prueba de rachas.
- Prueba de sumas acumuladas.

Estas pruebas serán utilizadas para evaluar la calidad y aleatoriedad del generador de numeros aleatorios. La aprobación de estas pruebas, lo cual consiste en la evaluación de un valor designado como *P*, indica que se cumple con los estándares requeridos por FIPS para el uso de entornos gubernamentales y de seguridad en los Estados Unidos.

1.7 Objetivo general

Implementación en hardware embebido analógico de un generador de números aleatorios basado en sistemas caóticos e hipercaóticos.

1.7.1 Objetivos específicos

- Investigar y estudiar la teoría de los sistemas caóticos para entender la diferencia entre los sistemas caóticos e hipercaóticos.
- Desarrollar la metodología para el TRNG y simularla de forma numérica en el software Matlab/Simulink.
- Implementar los sistemas caóticos e hipercaóticos en FPAA's.
- Implementar el TRNG usando los sistemas hipercaóticos.
- Validar los numeros aleatorios usando las pruebas estadísticas NIST-FIPS.

1.8 Orden de la tesis

El trabajo se organiza de la siguiente manera.

- El capítulo 1 hace una introducción a los temas relacionados a la tesis, describiendo las definiciones necesarias para su comprensión, entre ellas, los sistemas dinámicos lineales y no lineales. También introduce el concepto sobre los generadores de números aleatorios y hace la diferenciación entre los PRNG y TRNG explicando sus ventajas y desventajas, así como, la naturaleza de ellos. Los sistemas caóticos e hipercaóticos, la base del modelo TRNG desarrollado en la tesis, es presentado en este capítulo, los cuales serán implementados en FPAA's. Finalmente se explica la base del análisis estadístico de NIST que se hará para probar el modelo TRNG sugerido y los objetivos de la tesis.
- Un análisis más extenso sobre los sistemas caóticos se desarrolló en el capítulo 2, así como, la introducción de los sistemas hipercaóticos. La intención de este capítulo será la de fortalecer el conocimiento sobre los sistemas caóticos e hipercaóticos para su posterior implementación. Al final de este capítulo se realizarán las simulaciones numéricas de distintos sistemas caóticos e hipercaóticos en el software Matlab usando el integrador ODE45.
- En el capítulo 3 se desarrolla la metodología detallada utilizada para el TRNG, incluyendo la base con la que el generador funcionara, así como su descripción por etapas, además, se llevó a cabo el desarrollo en Simulink de las etapas descritas con anterioridad, desde la simulación hasta la obtención de la secuencia de bits aleatorios. Finalmente se presentan y describen las pruebas estadísticas NIST y más específicamente las pruebas NIST-FIPS, las cuales evaluaran y validaran la secuencia de bits obtenida por simulación como aleatoria, cada prueba es descrita al detalle incluyendo su función y significado.
- La implementación sobre los FPAA, así como la programación en la tarjeta con ASDB, es presentado en el capítulo 4, en donde se muestran todas las evidencias de su implementación. También se llevaron a cabo la recopilación y pruebas sobre la secuencia de bits obtenida, de la misma forma con la que se evalúa la simulación del TRNG, por medio de las pruebas NIST-FIPS.

- Finalmente, los resultados de la tesis son resumidos en el capítulo 5, en donde también se agregaron los apéndices con los programas utilizados, así como, las configuración y parámetros usado en la tarjeta ASDB.

Capítulo 2

Sistemas caóticos e hipercaóticos

2. Sistemas caóticos e hipercaóticos.

Desde el momento en el que los físicos han intentado describir las leyes de la naturaleza ha sufrido una especial ignorancia sobre el desorden que en esta habita [16], irregularidades como ocurren en la medición del clima, el oleaje del mar, las fluctuaciones en la población o incluso en las oscilaciones del pulso de un corazón, en todos ellos existen momentos de comportamiento caótico.

En 1960 el meteorólogo y matemático Edward Lorenz desarrollo una maquina capaz de pronosticar el clima; cada minuto la maquina imprimía una lista de numeros los cuales, si eras capaz de comprender su significado, eran capaces de predecir el clima. Sus colegas se juntaban con sus estudiantes para observar la máquina de Lorenz y hacer apuestas sobre qué valor imprimiría después. Un año después en 1961, Lorenz seguía trabajando con su máquina, durante sus simulaciones, decidió redondear el valor que introducía en el sistema para ahorrar tiempo en los calculo, para su asombro, su máquina había imprimido un resultado significativamente diferente ante un insignificante cambio en las condiciones iniciales, si bien en un inicio Lorenz supuso que esto era un error en su máquina, tras varias simulaciones encontró que este resultado se repetía, y que presentaba resultados muy distintos ante mínimas variaciones en los valores iniciales. Este hallazgo puso el manifiesto el llamado “efecto mariposa” el cual expresa la extrema sensibilidad que el comportamiento de un sistema puede tener a las condiciones iniciales.

En 1963 Lorenz publicó su trabajo en un artículo llamado “Deterministic Nonperiodic Flow” [18] en el Journal of atmospheric Sciences, en el que se describen los conceptos fundamentales del caos. El trabajo de Lorenz sentó las bases para el estudio y comprensión de los sistemas caóticos, con este, los científicos hallaron una gran cantidad de sistemas con comportamiento simulares lo que revoluciono muchos campos como lo son las matemáticas, física, economía, entre otras, y que lo sigue haciendo hasta hoy en día.

Como se mencionó en la introducción, no existe un consenso general dentro de la comunidad científica para la caracterización completa del caos, dependiendo de la literatura se puede dar una definición u otra, en lo que se relaciona a esta tesis la definición de un sistema caótico será:

Caos *es un sistema denso en sus orbitas periódicas determinista no lineal cuyo comportamiento exhibe una extrema sensibilidad a las condiciones iniciales*[1].

Para comprender mejor la definición anterior es necesario tener noción del significado que a ella componen. A continuación, se hará una definición de cada una de estas.

Definición: Densidad. Supongamos que *X* es un conjunto e *Y* es un subconjunto. Se dice que, *Y* es denso en *X* si, para cualquier punto $x \in X$, existe un punto *y* en el subconjunto *Y* arbitrariamente cercano a *x* [8].

Equivalentemente, Y es denso en X si para cualquier $x \in X$, podemos encontrar una secuencia de puntos $\{y_n\} \in Y$ que convergen en *x*. Por ejemplo, el conjunto de los numeros racionales es denso en el conjunto de los numeros reales, por otro lado, el conjunto de los numeros enteros no puede ser considerado como denso en el de los reales.

Para la siguiente definición será necesario definir lo que es la *iteración* de una función.

Definición: Función iterada. La iteración es el proceso de repetir el mismo proceso múltiples veces, entonces la iteración de una función es el proceso de la aplicación de esta función múltiples veces, se escribe como sigue, para una función F, $F^2(x)$ es la segunda iteración de *F*, es decir, $F^2(x) = F(F(x))$; la tercera iteración de *F* es $F^3(x) = F(F(F(x)))$ y, en general, $F^n(x)$ es la n-ésima iteración de F.

Definición: Orbita. Dado $x_0 \in \mathbf{R}$, definimos como *orbita* de x_0 bajo a *F* como una secuencia de puntos formados por $x_0, x_1 = F(x_0), x_2 = F^2(x_0), \dots, x_n = F^n(x_0), \dots$. El punto x_0 es conocido como la "semilla" de la órbita o "seed" en inglés. Esta definición es de suma importancia cuando definamos lo que son los puntos de equilibrio.

Definición: Extrema sensibilidad a las condiciones iniciales. Un sistema dinámico *F* es *extremadamente sensible a las condiciones iniciales* si existe un valor $\beta > 0$ tal que, para cualquier *x* y cualquier $\epsilon > 0$, existe un valor *y* entre ϵ y *x*, así como, un valor *k* tal que la distancia entre $F^k(x)$ y $F^k(y)$ es por lo menos β.

Esta definición nos intenta expresar que sin importar con que valor de *x* empieza el sistema ni que tan pequeña se elige la región cercana a *x*, siempre podemos hallar un valor de *y* cuya trayectoria eventualmente se separa de la trazada por *x* por lo menos una

[1]*Definición dada por el autor*

distancia β. En la figura 2.1 se presenta la simulación cada una de las variables de estado del sistema caótico de Lorenz y su evolución con respecto al tiempo de color naranja, de color azul se graficó la misma variable de estado, sin embargo, el valor de la condición inicial fue cambiado de 0.1 a 0.1001, nótese que esta diminuta variación genero un gran cambio en la evolución del sistema tras solo unas oscilaciones.

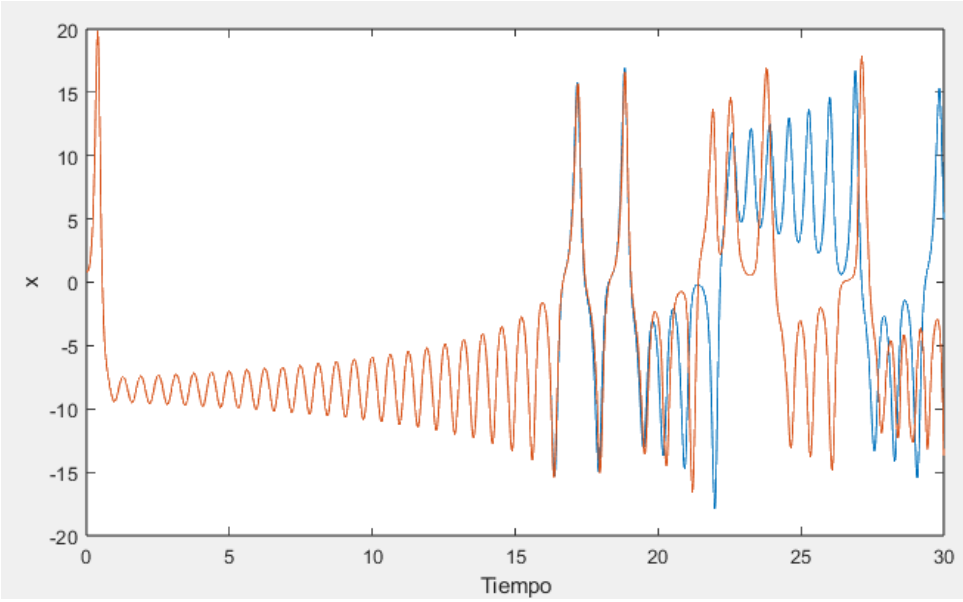


Figura 2.1. Simulación de la variable de estado *x* del sistema caótico de Lorenz usando dos condiciones iniciales **(a)** x(0)=0.1 (en naranja) y **(b)** x(0)=0.1001 (en rojo).

2.1 Sistema caótico de Chua

El sistema caótico de Lorenz ha sido estudiado en detalle debido principalmente a ser el primer sistema caótico con un conjunto de ecuaciones diferenciales, además, suele ser el sistema caótico que más atrae debido a su simetría sobre el eje z, sin embargo, existen una gran cantidad de sistemas caóticos entre los cuales está el conocido sistema caótico de Chua (abreviado SCC por sus siglas).

Durante los años 1980, un circuito relativamente sencillo se volvió popular entre la comunidad científica debido a su comportamiento caótico, gracias a su naturaleza como circuito electrónico, este comportamiento podía ser observado por medio de un osciloscopio. El diseño de este circuito se les atribuye a L. Chua, un ingeniero

electrónico y profesor, en ese momento, de la Universidad de California en Berkeley, y el científico japones T. Matsumoto; este consiste en un circuito RLC de simplemente 4 elementos activos y un diodo no lineal [9].

El circuito de Chua (figura 2.2) puede ser modelado por medio de un sistema de 3 ecuaciones diferenciales

$$\begin{aligned} C_1 \frac{dV_R}{dt} &= \frac{1}{R}(V_{C2} - V_R) - f(V_R) \\ C_2 \frac{dV_{C2}}{dt} &= i_L - \frac{1}{R}(V_{C2} - V_R) \\ L \frac{di_L}{dt} &= -V_{C2} - i_L R_s \end{aligned} \qquad 2.1$$

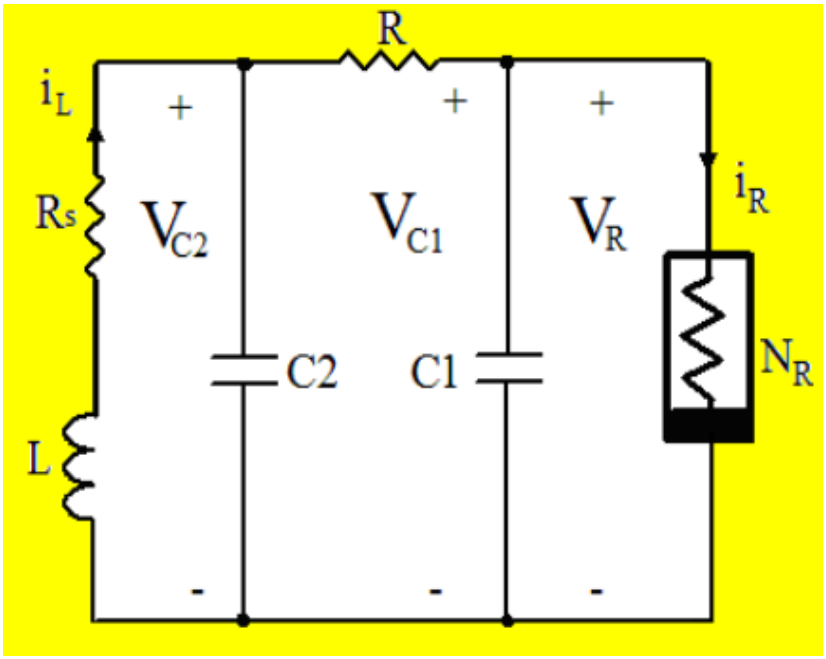


Figura 2.2. Circuito de Chua [27].

El dispositivo N_R es conocido como "el diodo de Chua", $f(V_R)$ representa la corriente que pasa sobre el diodo Chua. La ecuación característica de diodo de Chua es no lineal y de pendiente negativa para lo cual existen distintas propuestas las cuales se muestran

en la tabla 2.1 [27]. Con lo que respecta a esta tesis la ecuación característica no lineal del diodo de Chua utilizara su forma cubica definida por la ecuación (2.3).

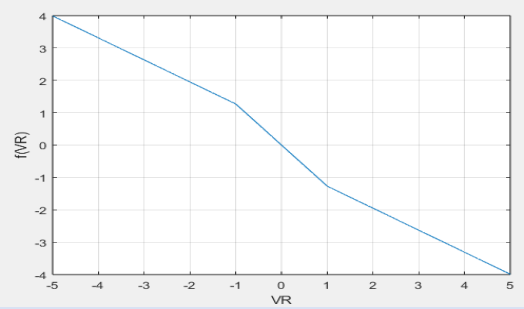
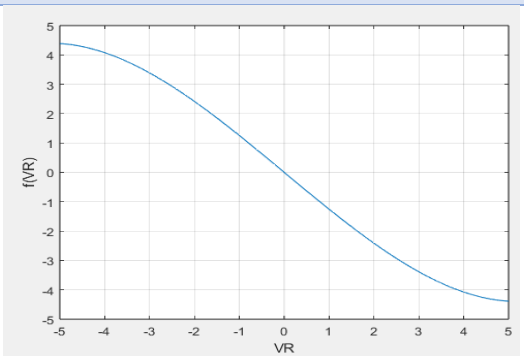
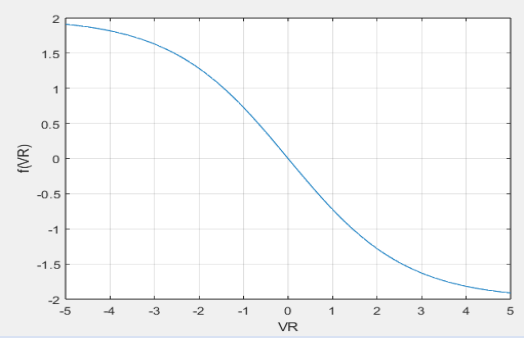
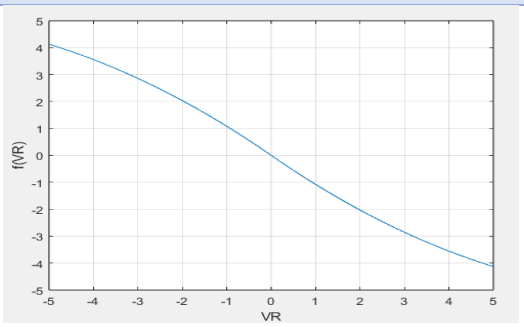
Definición de la función no lineal	Grafica asociada
$f(V_R) = -0.68V_R - 0.295(\lvert V_R + 1\rvert - \lvert V_R - 1\rvert)$ (2.2)	
$f(V_R) = -1.27V_R + 0.0157\,V_R^3$ (2.3)	
$f(V_R) = -2tanh(0.38V_R)$ (2.4)	
$f(V_R) = -\frac{8}{7}V_R + \frac{4}{63}V_R\lvert V_R\rvert$ (2.5)	

Tabla 2.1. Definiciones para la ecuación característica del diodo no lineal de Chua.

De forma general adimensional, el circuito de Chua puede ser descrito por la ecuación en espacio de estados

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}\alpha(y - x - f(x))\\ x - y + z\\ -\beta y\end{bmatrix} \tag{2.6}$$

Y presenta un comportamiento caótico para los parámetros α=10 y β=14.87.

2.2 Puntos de equilibrio del SCC.

Existen distintos tipos de *orbitas*, el más relevante es el llamado punto de equilibro. Un punto de equilibrio es un punto x_0 que satisface $F^n(x_0) = x_0$, de forma que la órbita de un punto de equilibro es la secuencia $x_0, x_0, x_0, \ldots$ [8]. Para un sistema dinámico los puntos de equilibro se encontrar al hallar las raíces del sistema como fue mencionado en la introducción y su solución está dada por (1.4).

Los puntos de equilibrio para el SCC dado por (2.6) se calculan por

$$\alpha\big(y - x - f(x)\big) = 0 \tag{2.7}$$

$$x - y + z = 0 \tag{2.8}$$

$$-\beta y = 0 \tag{2.9}$$

Por la ecuación (2.9) sabemos qué y debe ser 0, sustituyendo en (2.7) tenemos

$$x + f(x) = 0 \tag{2.10}$$

Las raíces de la ecuación (2.10) serán los puntos de equilibro para *x*, este valor y el punto de equilibro *y* = 0 se sustituyen en (2.8) para hallar nuestro conjunto de puntos de equilibrio. Para el caso en donde la función no lineal está definida por (2.3), las raíces de la ecuación (2.10) se obtienen por

$$-0.27x + 0.0157x^3 = 0 \tag{2.11}$$

La ecuación (2.11) no regresa tres soluciones, sustituyéndolas en (2.8) hallamos el conjunto de puntos de equilibro del SCC.

$$EP_1 = (0,0,0) \qquad 2.12$$
$$EP_2 = (-0.8294,0,0.4879)$$
$$EP_3 = (0.8294,0,-0.4879)$$

Los puntos de equilibrio tienen un impacto importante sobre cualquier sistema dinámico y nos indica en donde y de que forma el sistema se comportara, por ejemplo, supongamos que x_1 es un punto de equilibrio en un sistema dinámico unidimensional (como sola una variable dependiente) f, y $f'(x_1) = a > 1$, entonces la orbita de cualquier punto cercano x a x_1 se separa de ese punto [9]. Para el caso de los sistemas caóticos el punto de equilibro funciona como un atractor de la órbita, sin embargo, los sistemas caóticos suelen presentar múltiples puntos de equilibrio, como lo muestra la sección anterior para el SCC, generando que la órbita del sistema transite entre estos puntos de equilibrio. Además, los exponentes de Lyapunov para el SCC son [27]: λ_1=0.11, $\lambda_2 = 0, \lambda_3 = -0.73$.

2.3 Simulación numéricas de sistemas caóticos.

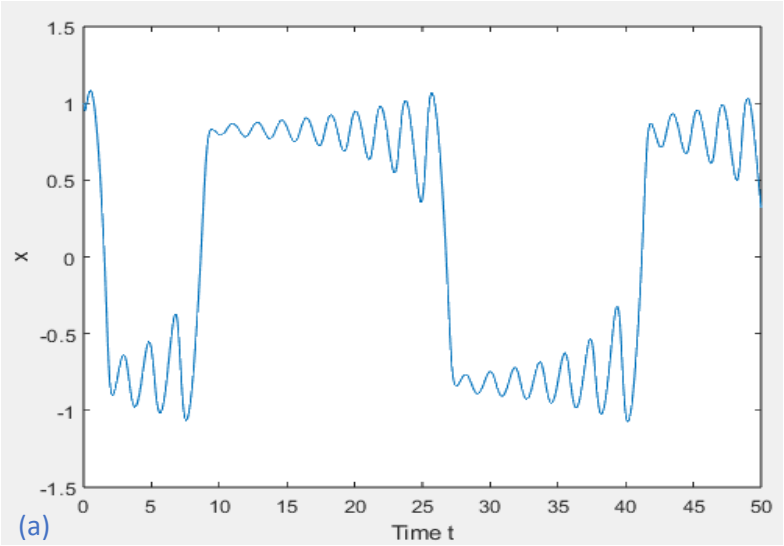
Los sistemas caóticos, como la mayoría de los sistemas dinámicos no lineales, no suelen tener una solución analítica a su sistema de ecuación diferenciales, la solución del sistema de ecuación diferenciales suele ser resuelta por métodos numéricos. Con la gran capacidad de los dispositivos de cómputo actuales, es muy sencillo poder simular sistemas dinámicos y observar su respuesta en un muy corto periodo de tiempo.

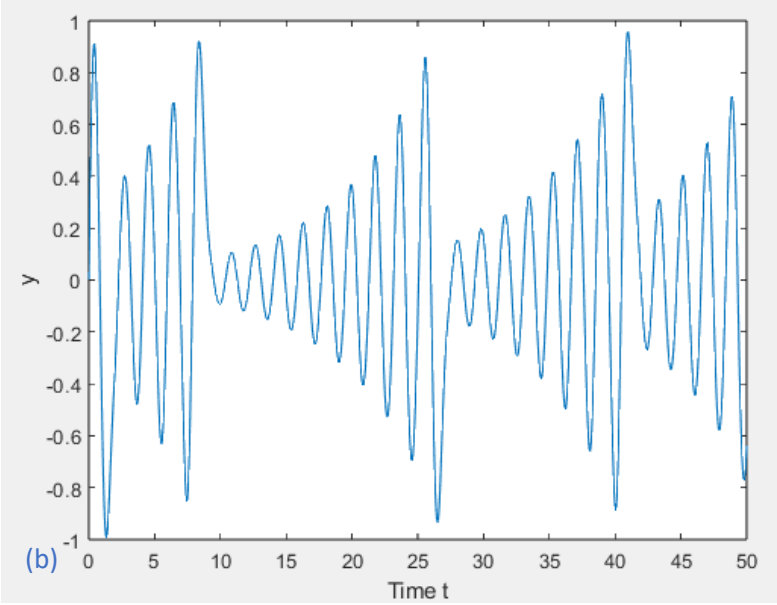
Las simulaciones numéricas realizadas en esta tesis fueron realizadas con una programación en el software Matlab, usando el integrador ODE45, el cual nos permite integrar un SED en un periodo de tiempo con unas condiciones iniciales dadas.

El integrador ODE45 es un método de un solo paso, es decir, soluciona el sistema de ecuaciones basándose solamente en el estado anterior. A diferencia de otras librerías que el software Matlab proporciona como lo pueden ser ODE15s o ODE23t, ODE45 es un

integrador versátil que permite simular gran parte de los sistemas de ecuaciones con relativa sencilles.

En lo que se relaciona a este trabajo, se necesitara de la simulación de dos sistemas caóticos en específico, el SCC (2.6) y el sistema hipercaótico definido por (2.14), la simulación de las variables de estado con respecto al tiempo correspondiente a sistema de Chua por medio del integrador ODE45 se observan en la figura 2.3, en el cual se utiliza la función no lineal cubica definida por (2.3) con valores iniciales (1, 0, 0) y el periodo de la simulación fue de 0 a 50 segundos.





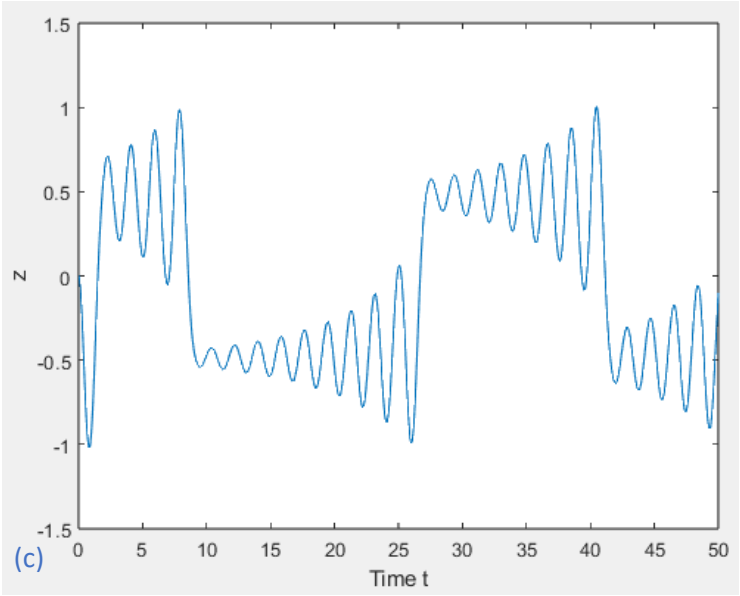
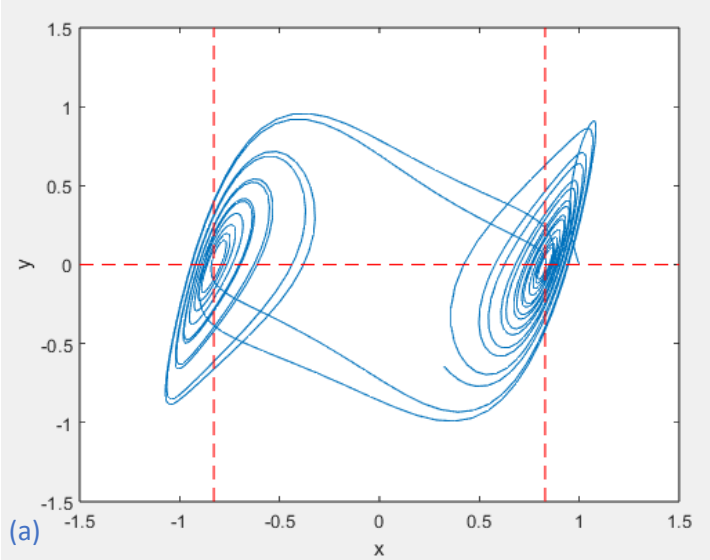


Figura 2.3. Simulación con periodo de 50 segundos del sistema caótico de Chua, respuesta con respecto al tiempo de las variables (a) x, (b) y (c) z con condiciones iniciales (1,0,0).

Cada variable de estado del sistema de Chua puede ser grafica una contra la otra generando planos llamados “diagramas de fase”, lo relevante de esto es que en estos se puede observar las orbitas que toma el sistema y su atracción a los puntos de equilibrio calculados en (2.12), los cuales resaltados por medio de líneas rojas punteadas perpendiculares a su valor en su respecto eje.



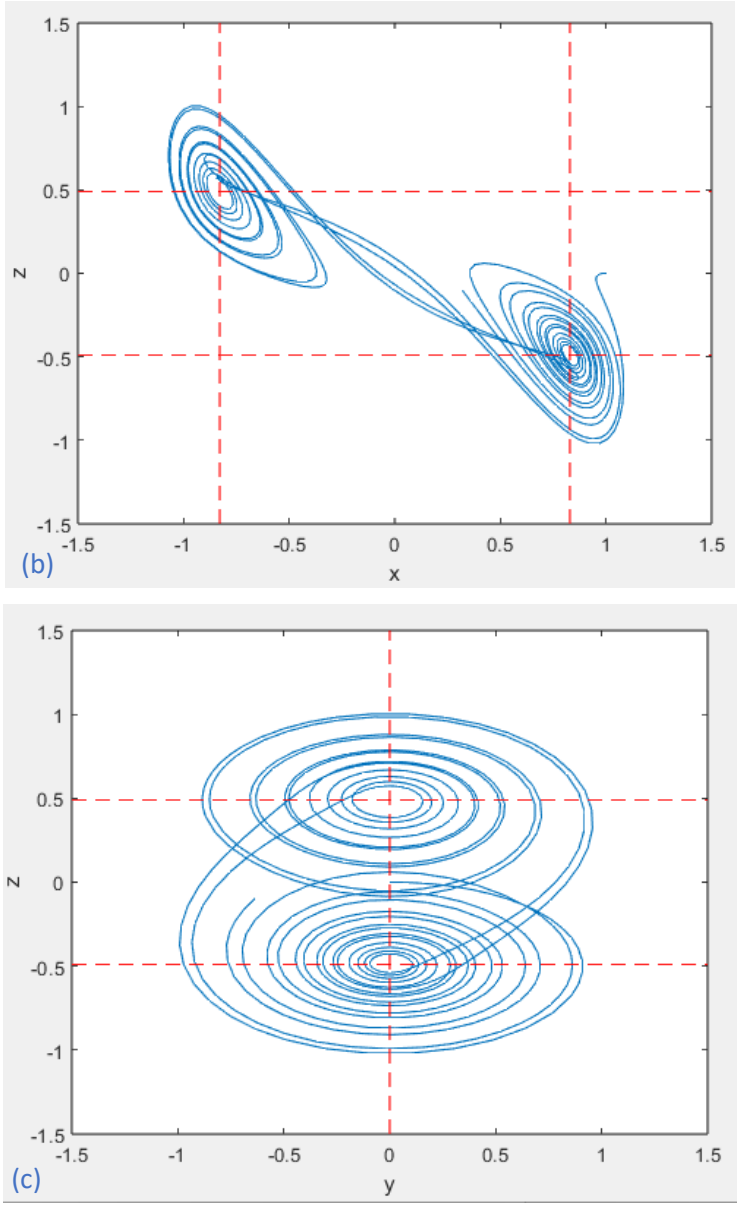


Figura 2.4. Graficas de los diagramas de fase con periodo de simulación de 50 segundos del sistema caótico de Chua con condiciones iniciales (1,0,0) de (a) y contra x, (b) z contra x (c) z contra y. Se resalta con líneas rojas los puntos de equilibrio $(\pm 0.8294, 0, \pm 0.4879)$.

Los códigos utilizados para las simulaciones en Matlab se encuentran en el Aprendice de este trabajo.

Por medio de la simulación también se pudo realizar el diagrama de fase 3D del SCC como se muestra en la figura 2.5.

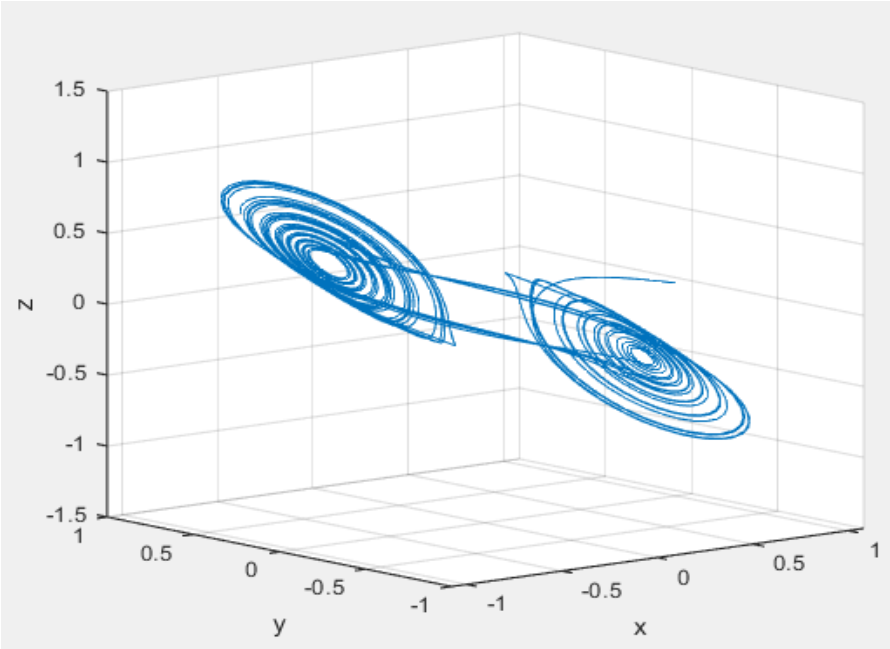


Figura 2.5. Diagrama de fase 3D del sistema caótico de Chua.

2.4 Exponentes de Lyapunov.

En un sistema dinámico caótico, las condiciones iniciales exactas pueden ser difíciles de conocer o medir con precisión, incluso pequeñas variaciones en las condiciones iniciales pueden tener un impacto significativo en el comportamiento futuro del sistema. El termino de numero de Lyapunov es introducido para poder cuantificar como se amplifican las perturbaciones del sistema a medida que este evoluciona en el tiempo (el exponente de Lyapunov es simplemente el logaritmo natural de este número) [9], por ejemplo, un numero de Lyapunov igual a 2 (exponente de Lyapunov aproximadamente igual a 0.69315) para la órbita de un punto de equilibrio x_1 significaría que la distancia entre la órbita de x_1 y la órbita de un punto cercano x se duplica en promedio cada iteración. De forma reciproca un numero de Lyapunov igual a $\frac{1}{2}$ (exponente de Lyapunov aproximadamente igual a -0.69315) de punto de equilibrio x_1 significaría que la distancia

entre la órbita de este punto y la de un punto cercano x se reduce a la mitad en promedio cada iteración.

La importancia del concepto de los exponentes de Lyapunov es que puede ser aplicadas a orbitas no periódicas como lo son los sistemas caóticos, cuya característica principal es la extrema sensibilidad a las condiciones iniciales.

Definición: Exponente de Lyapunov. Sea f un sistema continuo dentro de los reales. El número de Lyapunov $L(x_1)$ de la órbita $\{x_1, x_2, x_3, \dots\}$ está definido por

$$L(x_1) = \lim_{n\to\infty} (|f'(x_1)| \dots |f'(x_n)|)^{\frac{1}{n}}$$

Si este límite existe entonces el **exponente de Lyapunov** está dado por

$$\lambda(x_1) = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} \ln|f'(x_i)|}{n}$$

Es importante remarcar que el exponente de Lyapunov esta indefinido para algunas orbitas. En particular, una órbita que contenga al punto x_i con $f'(x_i) = 0$ causa a que el exponente de Lyapunov sea indefinido.

Para sistemas caóticos los exponentes de Lyapunov son de la siguiente manera:

Un exponente positivo, un exponente negativo y, un exponente cero. Mientras que para sistemas hipercaóticos los exponentes de Lyapunov son: dos o más exponentes positivos, un exponente negativo y, un exponente cero.

2.5 Sistemas hipercaóticos.

Los sistemas hipercaóticos representan una clase especial de sistemas dinámicos caóticos que exhiben un *comportamiento aún más complejo y caótico* que los sistemas caóticos tradicionales. Estos sistemas presentan *múltiples exponentes de Lyapunov positivos*, lo que implica una alta sensibilidad a las condiciones iniciales y una amplia variedad de trayectorias caóticas en su espacio de fase. La dimensión mínima para un sistema hipercaótico continuo es 4 [26].

Los antecedentes de los sistemas hipercaóticos se remontan al estudio de los sistemas dinámicos caóticos en general. A medida que los investigadores profundizaban en la teoría del caos, se dieron cuenta de que algunos sistemas exhibían una mayor complejidad y comportamiento caótico en comparación con otros. Esto llevó a la identificación de los sistemas hipercaóticos como una categoría específica de sistemas dinámicos caóticos.

El primer sistema hipercaótico fue propuesto por Otto E. Rössler [28], este expandio el conocimiento que se tenía sobre los sistemas caóticos y abrió las puertas a un nuevo tipo de sistemas, otros sistemas hipercaóticos fueron propuestos en base a sistemas caóticos ya existentes, sin embargo, el sistema hipercaótico de relevancia en este trabajo será una de las variantes del sistema hipercaótico de Lorenz [29] definido por

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} ay - ax - ew \\ xz - hy \\ b - xy - cz \\ ky - dw \end{bmatrix} \qquad 2.14$$

El sistema dinámico defino por (2.14) presenta un comportamiento hipercaótico para los parámetros $a = 5, b = 20, c = 1, d = 0.1, k = 0.1, e = 20.6$ y $h = 1$, con valores para los exponentes de Lyapunov iguales a [26]: λ_1=0.24, $\lambda_2 = 0.23$, $\lambda_3 = 0$ y $\lambda_4 = -7.56$ que, como se explica anteriormente, dado que dos de sus exponentes de Lyapunov son positivos el sistema se considera hipercaótico.

En general, para el SCC y el SHL los exponentes de Lyapunov son:

	λ_1	λ_2	λ_3	λ_4
Sistema caótico de Chua	*-0.73*	*0*	*0.11*	
Sistema hipercaótico de Lorenz	*-7.56*	*0*	*0.23*	*0.24*

Tabla 2.2. Exponentes de Lyapunov de los sistemas caóticos e hipercaóticos usados.

La simulación del SHL fue realizada de forma similar a las simulaciones para el sistema caótico de Chua, pero al ser un sistema hipercaótico continuo, este tiene 4 variables de estado por lo que en total tendremos 4 graficas con respecto al tiempo, 6 diagramas de

fase y 4 diagramas de fase 3D del sistema. Los resultados son presentados en las figuras siguientes.

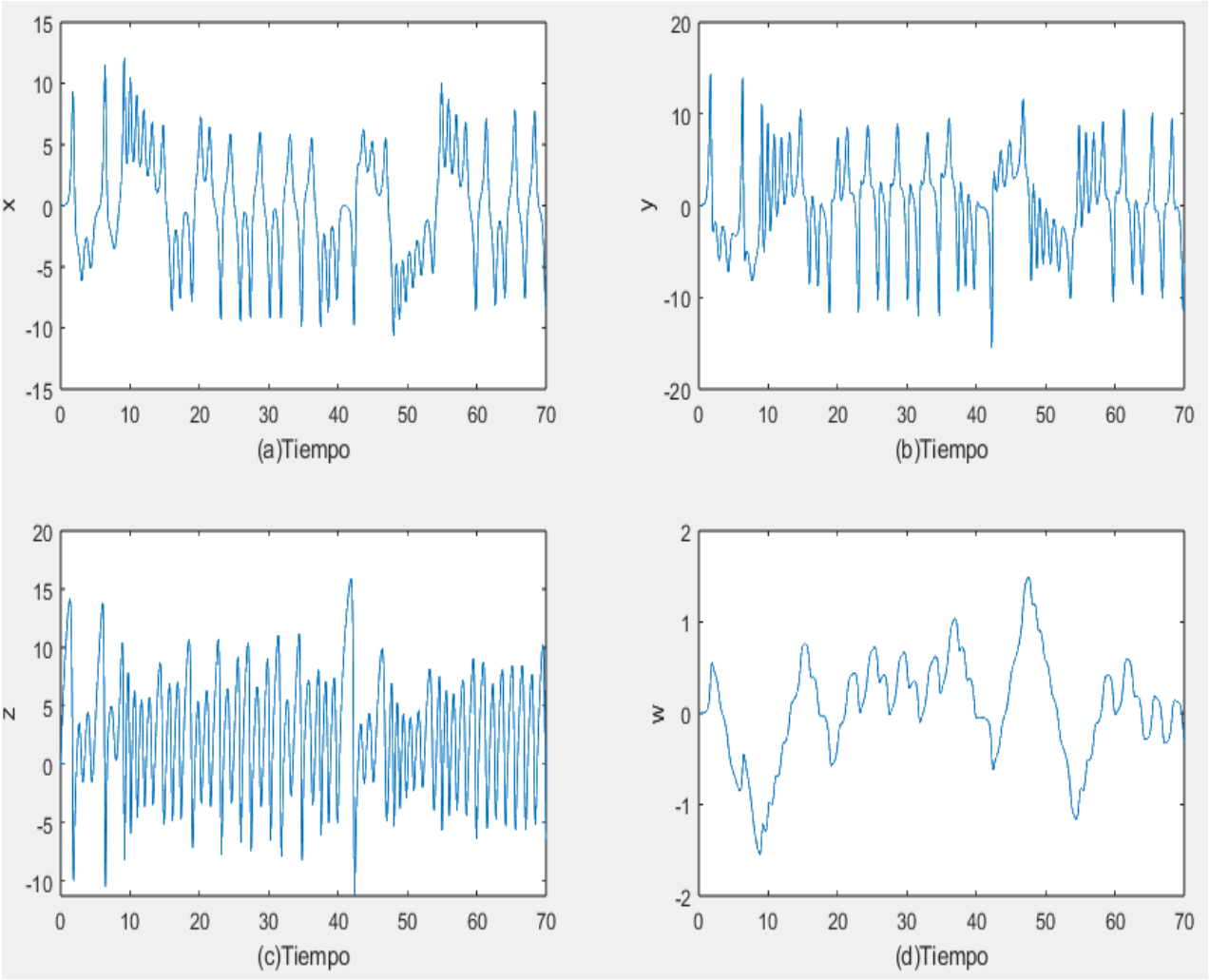


Figura 2.6. Simulación con periodo de 70 segundos del sistema hipercaótico de Lorenz (2.14), respuesta con respecto al tiempo de las variables (a) x, (b) y (c) z (d) w con condiciones iniciales (0.1, 0, 0, 0).

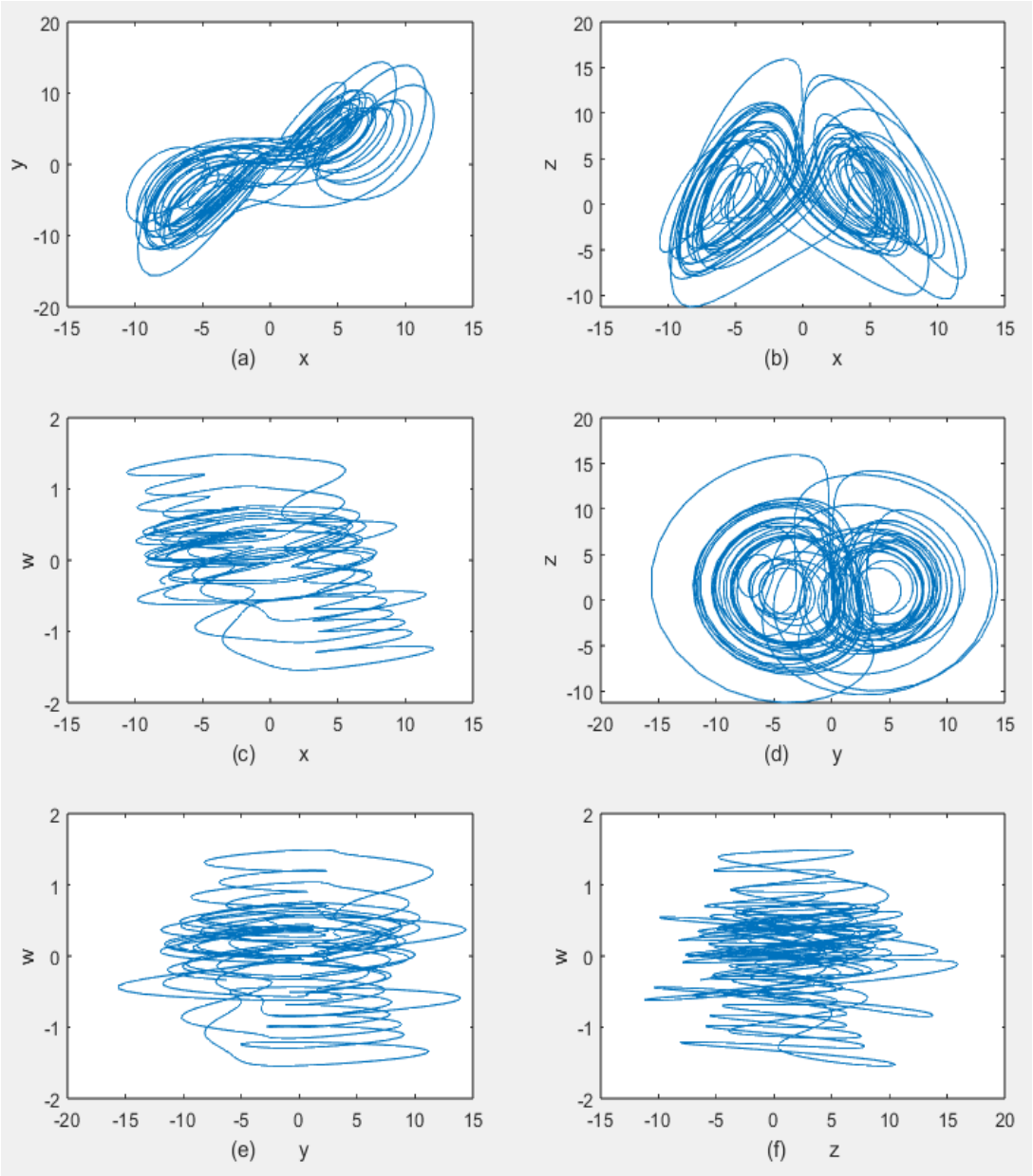


Figura 2.7. Diagramas de fase del sistema hipercaótico de Lorenz (2.14) con condiciones iniciales (0.1,0,0,0) de las variables de estados (a) *y contra x*, (b) *z contra x* (c) *w contra x* (d) *z contra y* (e) *w contra y* (f) *w contra z*.

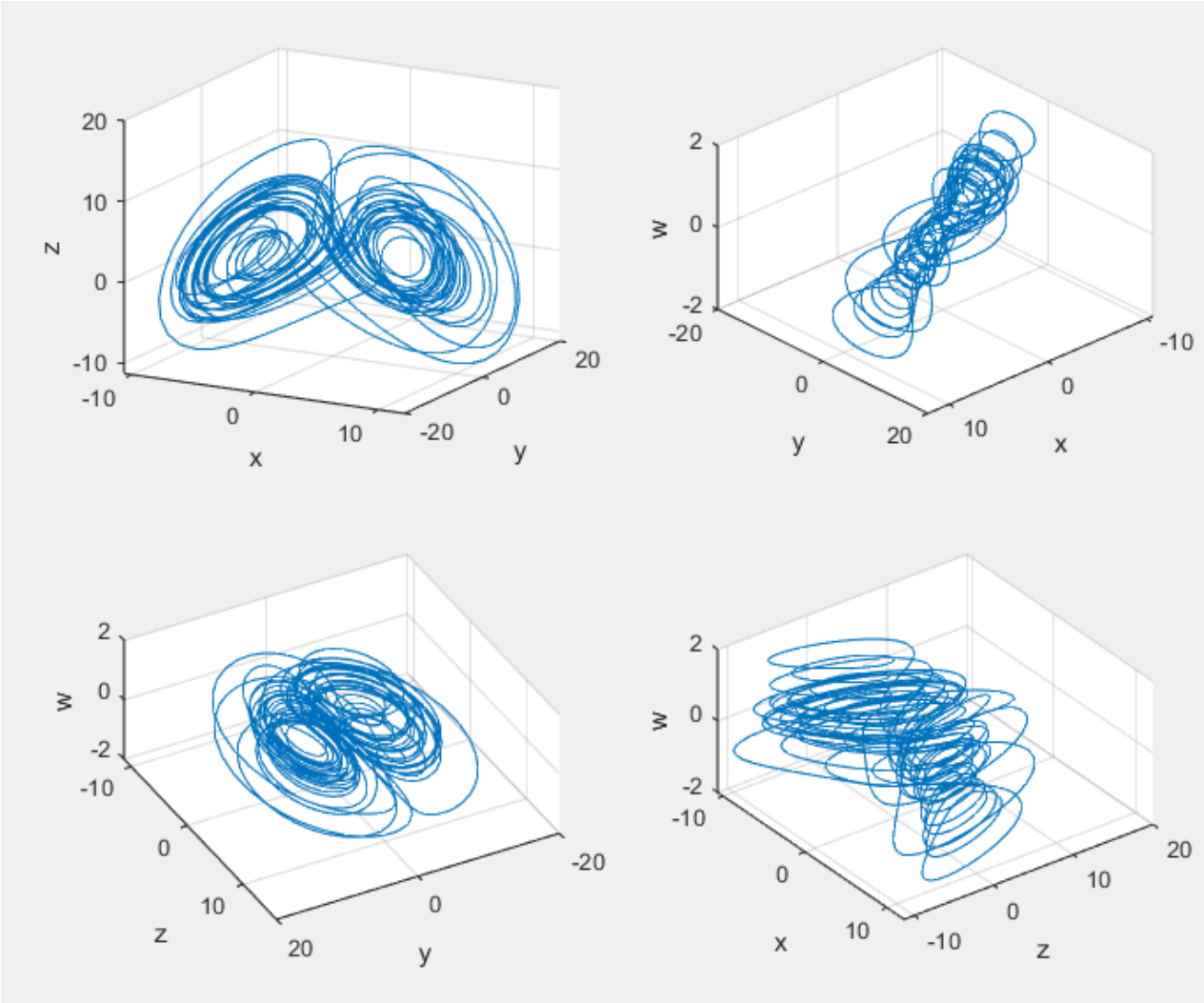


Figura 2.8. Diagramas de fase 3-D del sistema hipercaótico de Lorenz (2.14) con condiciones iniciales (0.1,0,0,0) de las variables de estados.

La simulación de sistema dinámicos nos permite comprender su comportamiento a lo largo del tiempo y con ello, evaluar ajustes o rediseños del sistema en cuestión, mejorando su eficiencia. El comportamiento irregular de los sistemas caóticos se beneficia mucho de la simulación numérica, nos permite afrontar de mejor manera los problemas que la naturaleza compleja inherente en ellos.

Capítulo 3

Metodología para el desarrollo del Generador de Numeros Aleatorios Verdaderos

3. Metodología para el desarrollo del Generador de Numeros Aleatorios Verdaderos.

Los campos de probabilidad y estadística están basados sobre los conceptos abstractos del espacio de probabilidad y variable azarosa [22], dando las bases para una elegante y poderosa teoría matemática, sin embargo, la implementación de estos conceptos exactos de esta teoría en una computadora convencional es más bien imposible, es por esto por lo que se recurre a métodos físicos con comportamientos irregulares.

Con regularidad la obtención de numeros aleatorios se realiza a través de un algoritmo y un valor inicial llamado "semilla", estos son los llamados numeros pseudoaleatorios, los cuales ofrecen una gran velocidad y sencilles en su obtención, sin embargo y debido a su naturaleza, son completamente dependientes de su "semilla". Para aplicaciones mas complejas y especificas en las cuales la entropía y la verdadera aleatoriedad de los numeros importante se utilizan fenómenos físicos que nos aseguren esta aleatoriedad.

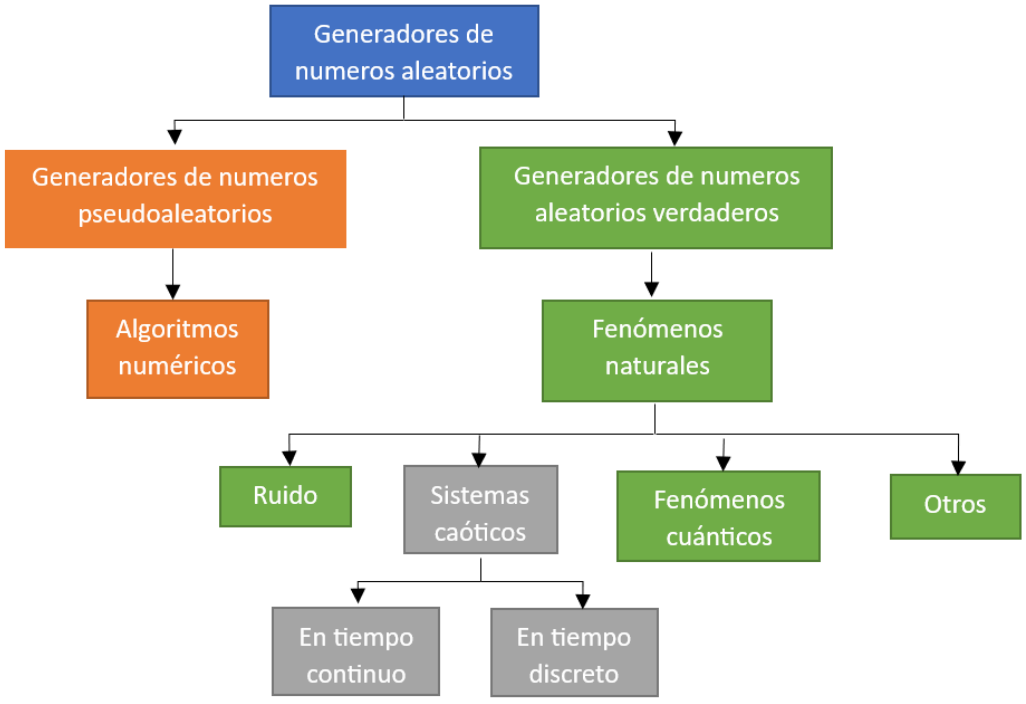


Figura 3.1. Tipos de generadores de numeros aleatorios [30].

Un generador de numeros aleatorios tiene por objetivo de proveer una secuencia de valores numéricos, regularmente una secuencia de 1 y 0, de los que no sea posible observar un patrón. Su esquema básico está representado en la figura 3.2, el cual consiste en tres etapas, un generador de entropía, un acondicionamiento y una conversión analógico digital. En algunas también agrega una etapa de post-procesamiento después de la conversión, esto para aumentar la calidad de los numeros aleatorios.

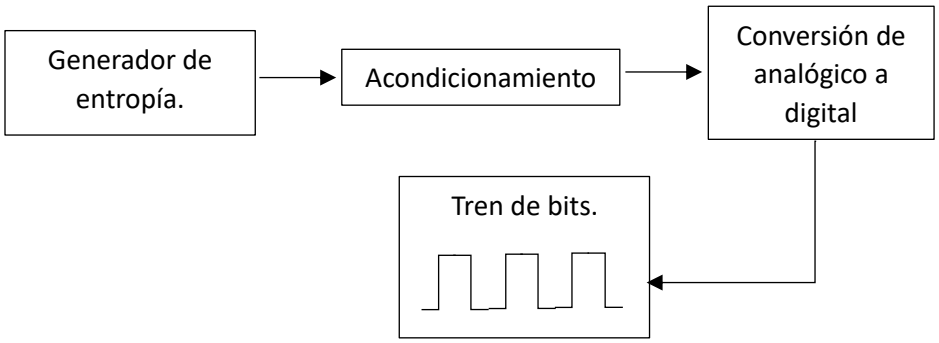


Figura 3.2. Esquema básico de un generador de numeros aleatorios [30].

Cada una de las etapas mostradas en el esquema básico son descritas como sigue:

El generador de entropía es la fuente externa de aleatoriedad que proporciona datos impredecibles y no deterministas. Puede ser un dispositivo físico especializado, como un generador de ruido atmosférico o sistema en base a fenómenos cuánticos, los datos de la fuente de entropía pueden requerir acondicionamiento, para esto se utiliza la segunda etapa de nuestro sistema.

En la etapa de acondicionamiento se pueden utilizar técnicas como el filtrado, escalamiento, o en general cualquier técnica que nos permita adecuar las señales analógicas obtenidas, además, en muchas ocasiones es necesario mejorar la calidad de los numeros aleatorios, así como su uniformidad de ellos, esto se realiza por distintos métodos, alguno de ellos son método Von-Neumann o el Hashing.

Finalmente, la salida analógica obtenida en la etapa dos se convierte a digital por algún método o dispositivo, obteniendo el tren de bits de salida.

La base con la que se diseñó el generador de numeros aleatorios consta de dos sistemas caóticos conectados a un Flip-Flop (FF) y a la salida de este se obtendrá el tren de bits.

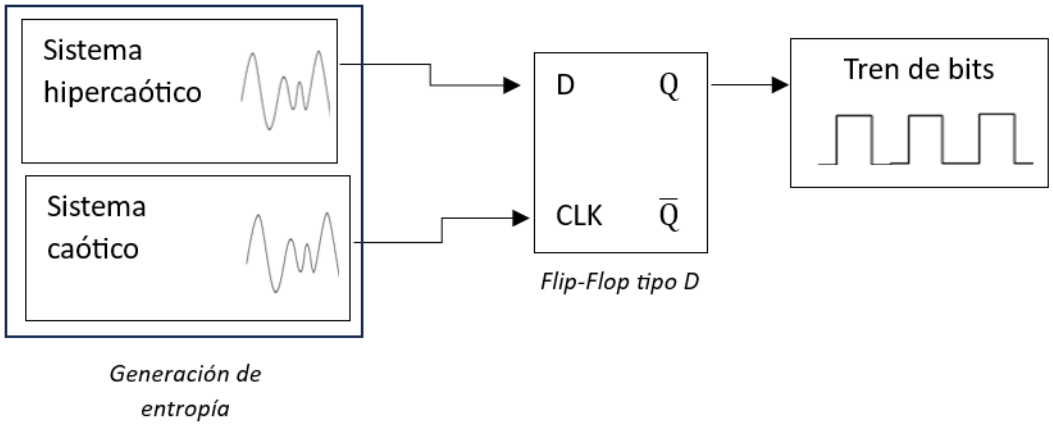


Figura 3.3. Diseño base para el generador de numeros aleatorios.

Un sistema hipercaótico continuo tiene mínimamente 4 variables de estado, esta característica puede ser utilizada para desarrollar un RNG novedoso, permitiéndonos obtener más de un tren de bits, además, un FPAA es un dispositivo el cual no solo no permitirá implementar los sistemas caóticos, sino también, contiene una gran de cantidad de operaciones analógicas, las cuales podemos aprovechar para el acondicionamiento del sistema.

Con los datos anteriores en mente, se desarrolló el modelo mostrado en la figura 3.4, este se secciono en 4 etapas distintas, además se resalta la cantidad de salidas en cada bloque.

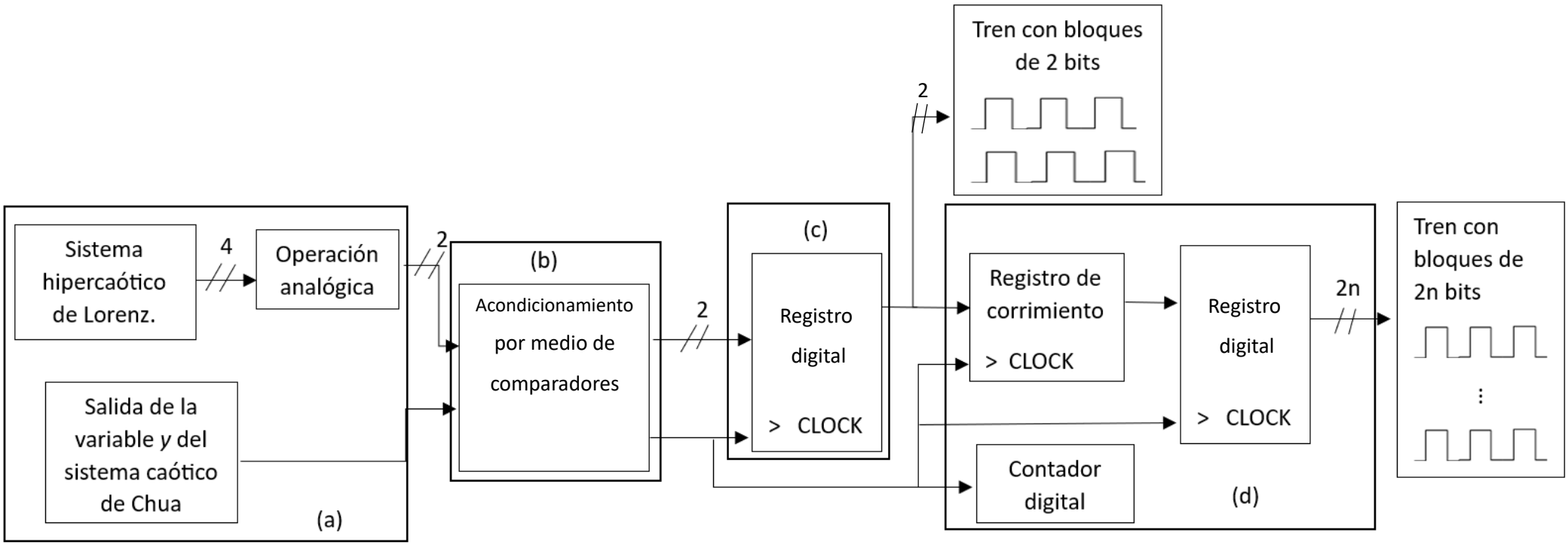


Figura 3.4. Diagrama del modelo propuesto para el generador de numeros aleatorios verdaderos dividido en 4 etapas: (a) Generación de entropía, (b) Acondicionamiento, (c) Registro digital y, (d) Tren con bloques de $2n$ bits.

- Etapa generación de entropía:

Como se tenía pensado en el modelo base de la figura 3.3, se utilizó un sistema caótico, el cual tendrá la función de generar un CLOCK digital no uniforme, y un sistema hipercaótico para generar el tren de bits no uniformes aleatorios; nótese del modelo completo, que se agregó una operación analógica después de las 4 señales hipercaóticos de sistema de Lorenz, la razón de esta es la de desacoplar las señales generadas por el mismo sistema, de otra manera el sistema produce resultados no deseados, la operación utilizada será la suma de las señales analógicas correspondientes a las variables de estado "x" con "y", y "z" con "w" del sistema hipercaótico de Lorenz (2.14), además, el FPAA AN231E04 nos permite realizar distintas operación analógicas, por lo que este modelo aprovecha esta característica. Después de la operación analógica y debido a esta, las señales analógicas pasan a ser 2 en lugar de 4.

Debido a que solo utilizamos una variable de estado del SCC, esta operación analógica no es necesaria en él.

- Etapa de acondicionamiento:

El sistema además agrega un acondicionamiento por medio de comparadores, este nos permitirá tener cierto control sobre el TRNG, las señales analógicas generadas en la primera etapa son convertidas a una secuencia de bits (figura 3.5), su principal uso es el de mejorar la calidad de los numeros aleatorios generados, así como su uniformidad eligiendo el valor correcto de la comparación. La comparación de voltaje también es una de las operaciones disponibles en el FPAA AN231E04. Nótese de las gráficas mostradas en la figura 3.5 que el valor de comparación afecta en gran medida la salida de esta etapa, la gráfica de la variable de estado "*z"* del sistema hipercaótico de Lorenz mostrada en la figura 3.5 (a) es comparada con un valor de 2.5 y la salida es mostrada en la figura 3.5 (b), mientras que las secciones (c) y (d) de la figura se realiza una comparación con un valor de 6.

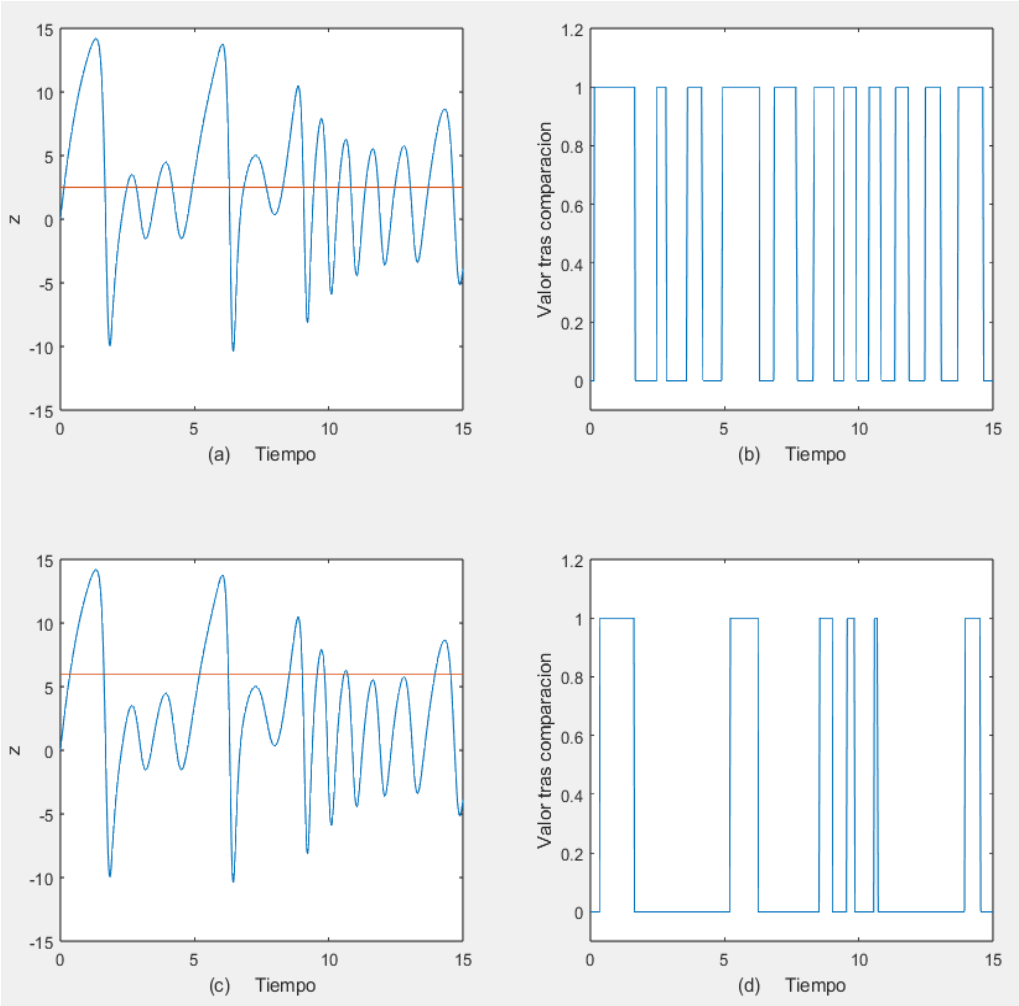


Figura 3.5. Respuesta tras comparación de la variable de estado "z" del hipercaótico de Lorenz. Los valores de comparación para las gráficas (a) y (b) es de 2.5, para las gráficas (c) y (d) de 6.

Para el caso de la variable de estado *y* del SCC, la comparación fue ajustada al punto en donde la órbita es más densa.

- Etapa del registro digital:

En el modelo base del generador de numeros aleatorios (figura 3.2) se utilizaba un FF tipo D para obtener una secuencia de bits, esto fue cambiado por un registro digital debido que se obtienen más de una entrada de los comparadores, tras estos, no se obtiene solo una secuencia de bits, sino, se obtienen secuencias de bloques con 2 bits cada uno,

por lo que, se expanden la cantidad de FF necesarios, un conjunto de FF tipo D en electrónica digital es equivalente a un registro digital, si bien, en este modelo simplemente se requiere de 2 FF, dependiendo de las salidas del sistema hipercaótico o usando un conjunto de sistemas caóticos, la cantidad necesaria de FF aumentara por lo que el tamaño del registro analógico deberá también, ser ampliado.

Para lo que el modelo propuesto se refiere la cantidad de bits obtenida está relacionada por

$$n_\varepsilon = 2^k \qquad 3.1$$

Donde n_ε es la cantidad de bits generados y k el número de bloques generados o, equivalentemente, el número de flancos positivos generados por salida del comparador con la señal la variable de estado *y* del SCC.

- Tren de bloques de 2n bits:

A la salida del registro digital ya se obtienen bloques de 2 bits aleatorios, sin embargo, se diseñó una etapa extra que nos permitirá concatenar los bloques de dos bits en bloques de 2n bits, dependiendo del diseño usado, por ejemplo, se puede una salida con bloques de 8 bits o lo que es equivalente en bytes el cual es una secuencia de 8 bits.

Para poder obtener bloques de bits más grandes, usaremos un dispositivo llamado registro de corrimiento y un registro digital, el primero concatena los bloques de dos bits generados y el segundo nos permite almacenar los bits concatenados. Por medio de estos dos dispositivos podremos obtener bloques de $n * 2^k$ bits, donde n es el valor máximo de la suma realizada por el contador digital.

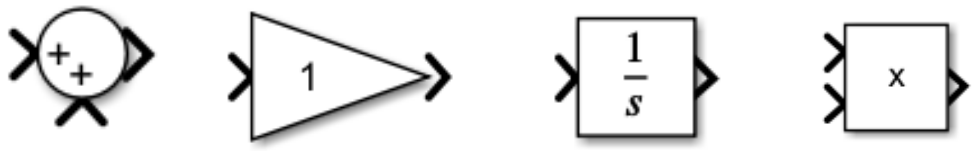
3.1 Simulación del modelo en Matlab-Simulink

Para comprobar el comportamiento esperado del modelo para el TRNG, optimizar su diseño y rendimiento, así como de ajustar los parámetros del modelo, se requiere de la simulación de este, si bien, la simulación de los sistemas caóticos anteriormente

realizada fue desarrollada en el programa base de Matlab, cuando se trata de simular sistemas más complejos de esta manera suele ser más compleja, tediosa y en muchas ocasiones esto nos lleva a errores.

Tomando las desventajas de la programación en texto antes descritas, Matlab contiene una poderosa herramienta de simulación y modelado grafico llamada Simulink, la cual nos permite representar sistemas mediante bloques gráficos los cuales podemos conectar entre sí. Cada uno de estos bloques pueden representar operaciones matemáticas, componentes físicos, sistemas de control, sensores, actuadores, o incluso programación en texto. Simulink también nos permite visualizar los resultados del sistema en un periodo de tiempo establecido o tiempo real.

En el caso de los sistemas dinámicos, es posible transformar una representación del sistema de ecuación diferenciales o SED por sus siglas, a una representación en bloques por medio de sumadores, ganancias, constantes, integradores y cualquier operación matemática con la que esté representada el SED (figura 3.6) interconectadas entre ellas.



(a) (b) (c) (d)

Figura 3.6. Representación en bloques por Simulink de (a) Operación suma (b) Ganancia o producto por un escalar (c) Integración (d) Multiplicación.

La representación en bloque de un integrador esta dado en términos de la variable compleja *s*, este bloque integra la señal de entrada con respecto al tiempo y entrega el resultado en su salida. Podemos expresar un SED introduciendo la variable compleja s de la siguiente forma

$$x_1 = \frac{1}{s}(f_{11}(x_1, x_2, \dots, x_n) + f_{12}(x_1, x_2, \dots, x_n) + \cdots)$$
$$x_2 = \frac{1}{s}(f_{21}(x_1, x_2, \dots, x_n) + f_{22}(x_1, x_2, \dots, x_n) + \cdots) \qquad 3.2$$
$$\vdots$$
$$x_n = \frac{1}{s}(f_{n1}(x_1, x_2, \dots, x_n) + f_{n2}(x_1, x_2, \dots, x_n) + \cdots)$$

Donde $x_1, x_2, \dots, x_n$ son las variables de estado de un sistema de n ecuaciones diferenciales y $f_{ik}(x_1, x_2, \dots, x_n)$ es la k-esima expresión de la i-esima ecuación del SED. Por medio de la ecuación (3.2) podemos expresar el SED por medio bloques como sigue.

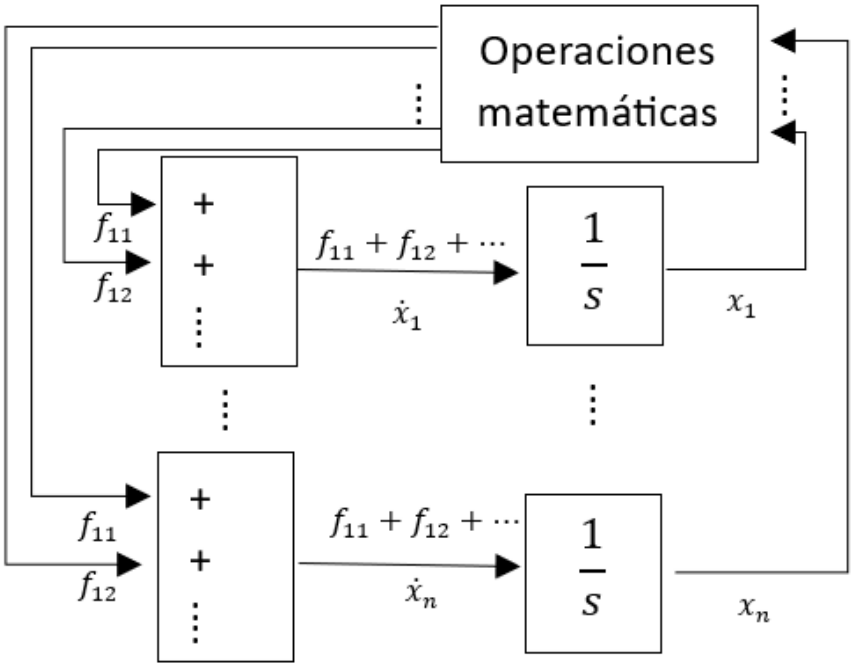


Figura 3.7. Representación general de un SED en bloques.

Tomando como base la representación de la figura 3.7 se desarrolló la programación en bloques del sistema hipercaótico de Lorenz dado por la ecuación (2.14) y la programación del SCC dado por (2.6).

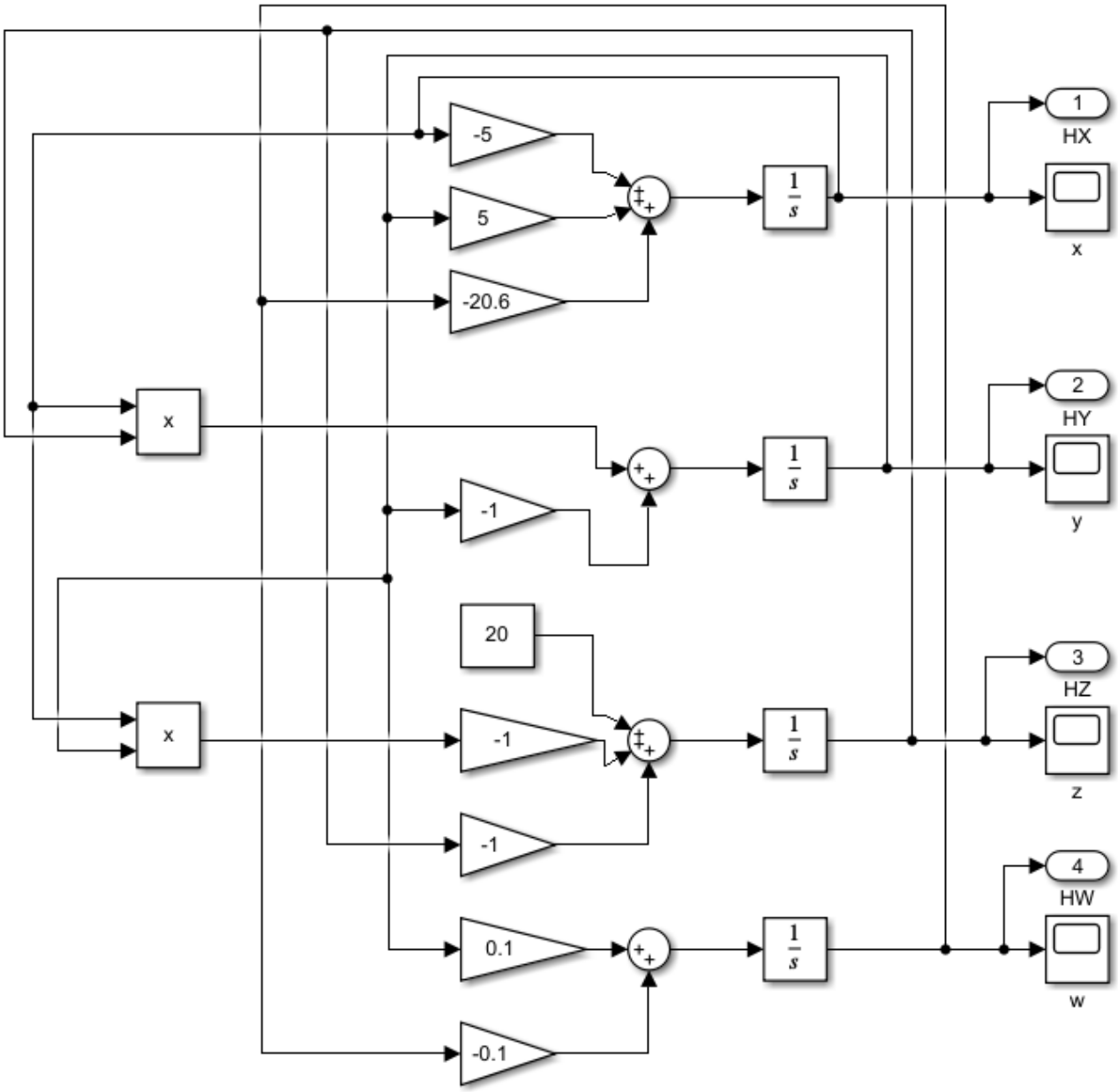


Figura 3.8. Representación del sistema hipercaótico de Lorenz en bloques de Simulink.

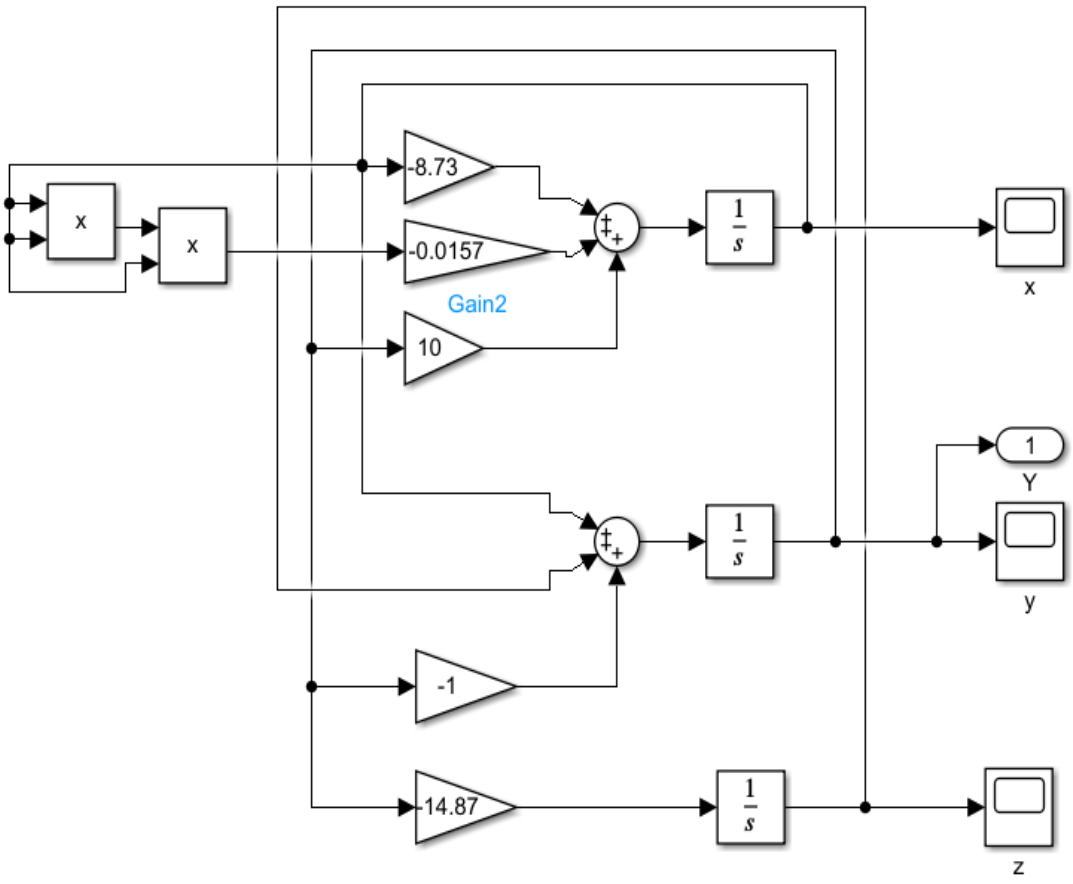


Figura 3.9. Representación del sistema caótico de Chua en bloques de Simulink.

Los FPAA, dispositivos en los cuales se implementará el TRNG, trabajan en un rango de voltaje especificado en su hoja de datos, para el caso del AN231E04, se trabaja en un rango de voltaje entre $\pm 3\,V$, es decir, pasado de este voltaje, el FPAA se satura y el sistema no funciona con normalidad. Esto quiere decir que, el sistema caótico debe trabajar dentro de este rango, para realizar esto el sistema caótico debe ser **escalado**, es decir, el valor máximo y mínimo esperable debe ser de máximo 3V y mínimo -3V, *se tomara un rango de* $\pm 1V$ *para los sistemas caóticos a utilizar.*

Tras la simulación del SCC y el SHL se encontró el valor máximo y mínimo de estos sistemas fue de $(\pm 5, \pm 1.2, \pm 8.4)$ y $(\pm 15.26, \pm 17.2713, 18.1937, 2.6)$ respectivamente. El escalamiento consiste en sustituir cada variable por sí misma multiplicada por un valor de escalamiento, para nuestro caso tenemos

$$\begin{aligned} x \to 5x^* \quad &=> \quad \dot{x} = 3.7x + 2.4y - 3.925z \\ y \to 1.2y^* \quad &=> \quad \dot{y} = 4.167x - y + 7z \\ z \to 8.4z^* \quad &=> \quad \dot{z} = -2.1y \end{aligned} \tag{3.3}$$

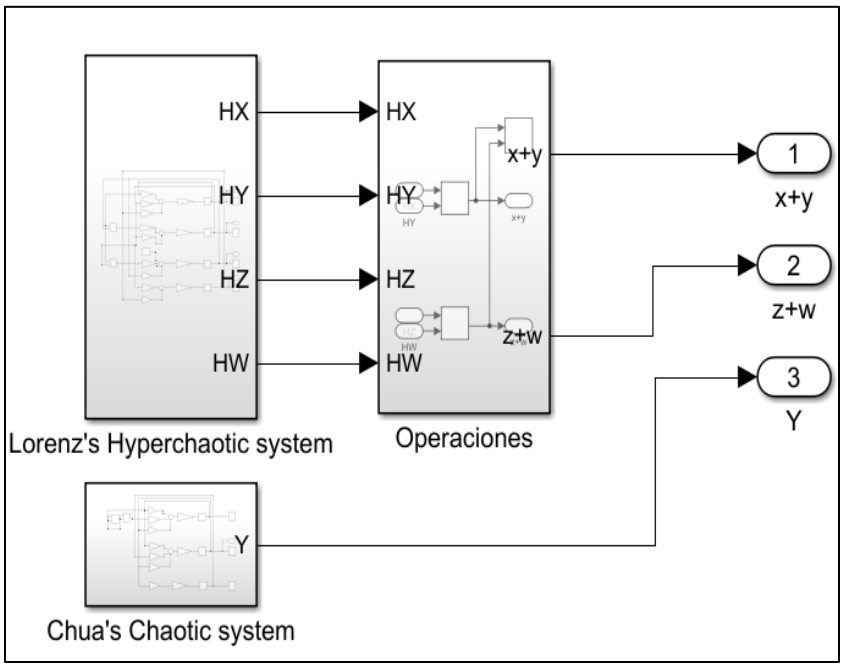
Y el sistema hipercaótico de Lorenz escalonado queda de la siguiente forma

$$\begin{array}{lll} x \to 20x^* & => & \dot{x} = -5x + 5y - w \\ y \to 1.2y^* & => & \dot{y} = 20xz - y \\ z \to 20z^* & => & \dot{z} = 1 - 20xy - z \\ w \to w^* & => & \dot{w} = y - 0.1w \end{array} \qquad (3.4)$$

Los parámetros dentro en los diagramas de los sistemas caóticos fueron actualizados usando los coeficientes en las ecuaciones (3.3) y (3.4), la ecuación (3.3) se referirá como el sistema caótico de Chua escalado (SCCE) y la (3.4) se referirá como el sistema hipercaótico de Lorenz escalado (SHLE).

Con el fin de tener un mejor orden y visualización sobre el generador de numeros aleatorios, se recurrió a dividirlo en subsistemas, cada uno de estos representara una de las etapas del modelo de la figura 3.4, Simulink tiene un bloque especifico con esta función de nombre “Subsystem”, los sistemas caóticos fueron colocados en dos subsistemas, cada uno con su respectivo nombre y conteniendo los diagramas de bloques de las figuras 3.8 y 3.9. Conectado con el subsistema del SHL se encuentra otro subsistema, este contiene las operaciones matemáticas que nos permitirán desacoplar las ecuaciones deferenciales de los sistemas caóticos llamado “Operaciones” y su contenido se muestra en la figura 3.10 (b) junto con el subsistema contenedor de toda la etapa de generación de entropía

(a)



(b)

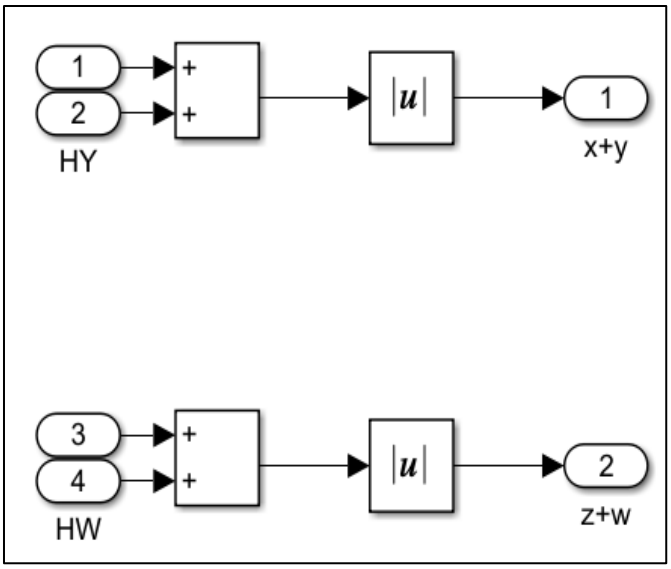


Figura 3.10. (a) Interior del subsistema "Generación de entropía". (b) Interior del subsistema "Operaciones"

Dentro del subsistema "operaciones" también se añadió un valor absoluto sobre las señales de entrada, la razón de esto es evidente en la figura 3.11 en donde se observa la respuesta en el tiempo de la señal analógica "x+y" y "|x+y|", la cual deja en evidencia su efecto sobre la distribución de los bits aleatorios generados tras la comparación y el aumento en frecuencia.

(a)

(b)

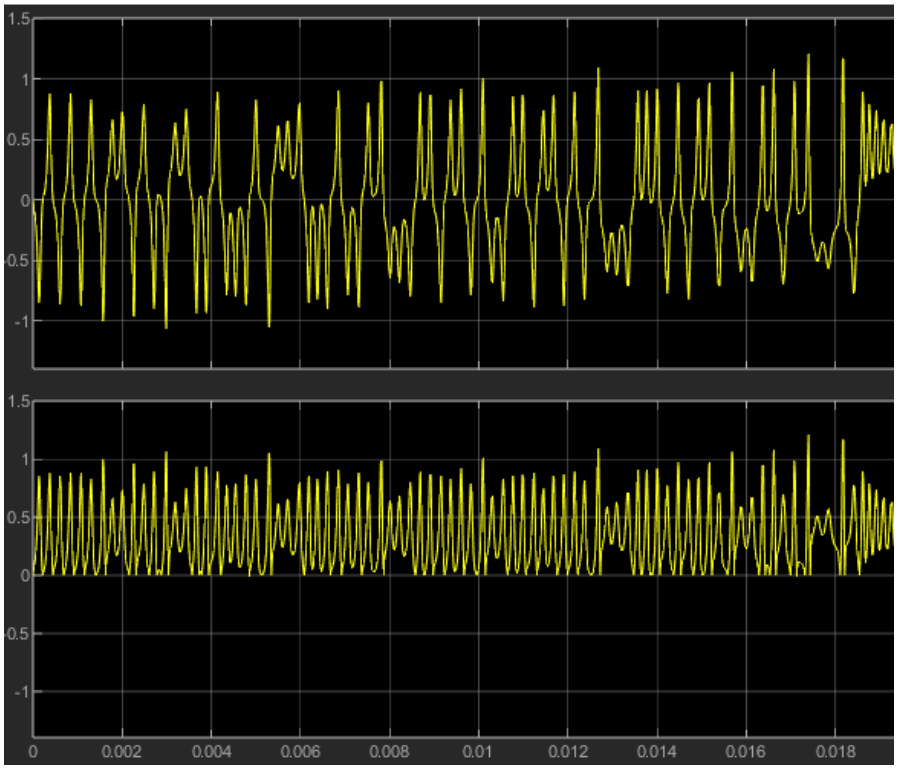


Figura 3.11. (a) Señal x+y, (b) Señal |x+y|.

Cada subsistema tiene el nombre de su etapa correspondiente (figura 3.12), y cumple con las funciones explicadas a través de este capítulo. Además, se observan dos bloques de nombre “out.bloque2bits” y “out.bloque8bits”, estas son las salidas del TRNG con la secuencia de numeros aleatorios, el primero contiene bloques de 2 bits y el segundo bloques de 8 bits o un byte.

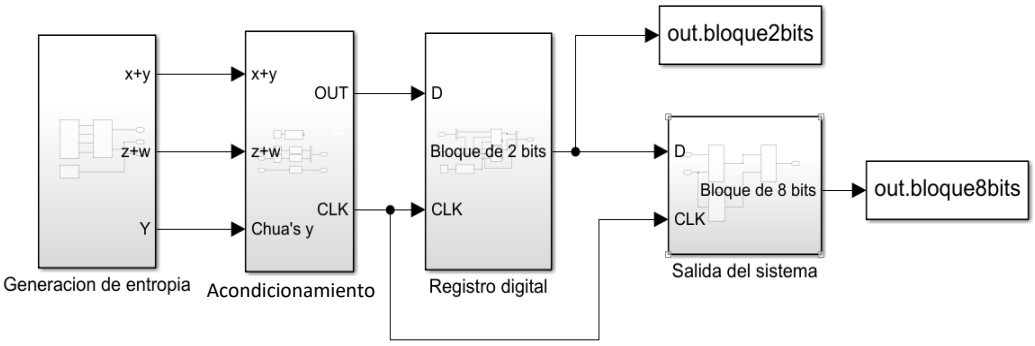


Figura 3.12. Modelo completo del TRNG en Simulink.

El subsistema “Acondicionamiento” (figura 3.13), contiene los comparadores de nivel, y como se explicó antes, nos permiten tener control sobre la distribución del TRNG, estos reciben la señal analógica y la transforman en una señal digital, si el valor analógico es mayor a una referencia la salida es 1, de lo contrario la salida resultara en un 0. Los valores de referencia de los comparadores fueron seleccionados tras varias simulaciones y con el fin de aprobar las pruebas estadísticas NIST-FIPS, las cuales serán detalladas más adelante en este capítulo.

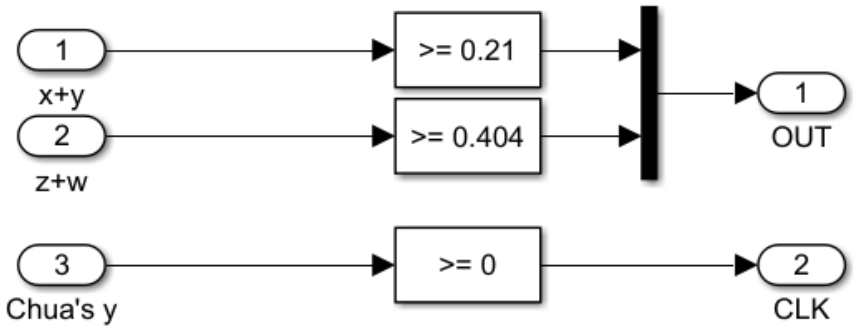


Figura 3.13. Subsistema “Acondicionamiento”.

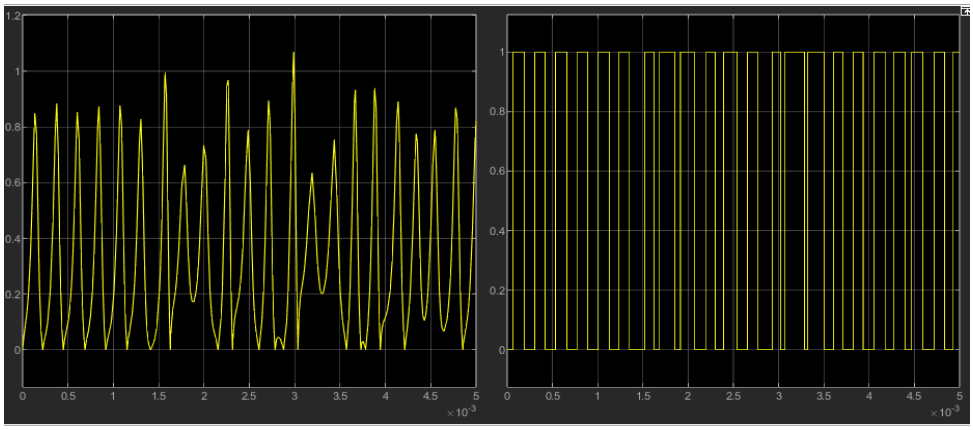


Figura 3.14. Bits de salida tras comparación de la señal |x+y|.

Tras las comparaciones se obtiene una secuencia de bits en bloques de 3, uno correspondiente al SCC y el cual se usará como CLOCK del registro digital en el siguiente subsistema, las dos salidas restantes serán las entradas del siguiente subsistema.

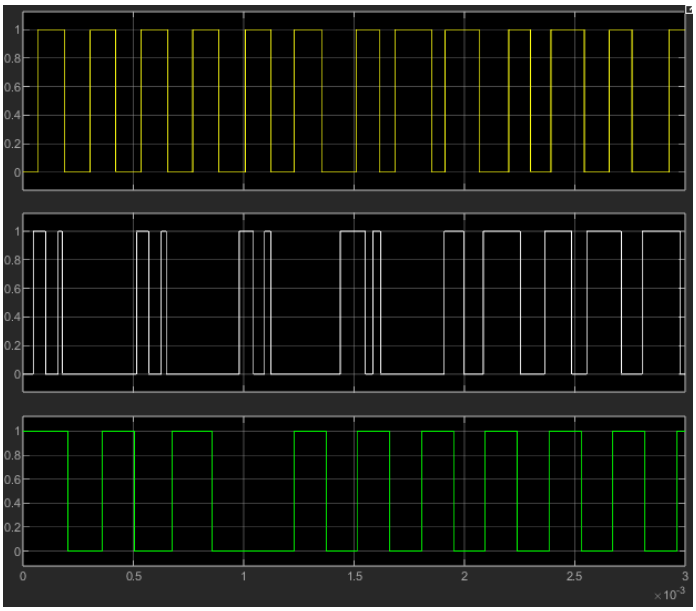


Figura 3.15. Salida del subsistema “Acondicionamiento”.

Los bits de salida obtenidos se almacenan en bloques dos bits por medio de en registro digital, el registro digital es formado por 2 FF tipo D, recordemos que el CLK es obtenido por el tercer tren de bits del subsistema “acondicionamiento”.

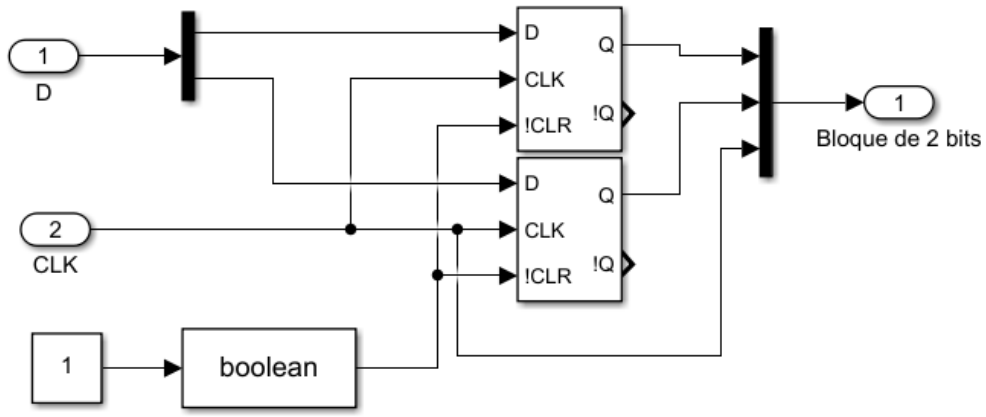


Figura 3.16. Subsistema “Registro Digital”.

Finalmente, el subsistema de nombre “salida del sistema” el cual representa la etapa para la obtención del tren de bloques de 2n bits esta mostrado en la figura 3.17, la razón de esta etapa, para la presente simulación, será la de tomar los bloques de 2 bits y convertirlo

a 1 byte, o bloques de 8 bits, en la figura 3.18 se observa tanto la entrada como la salida del subsistema de salida.

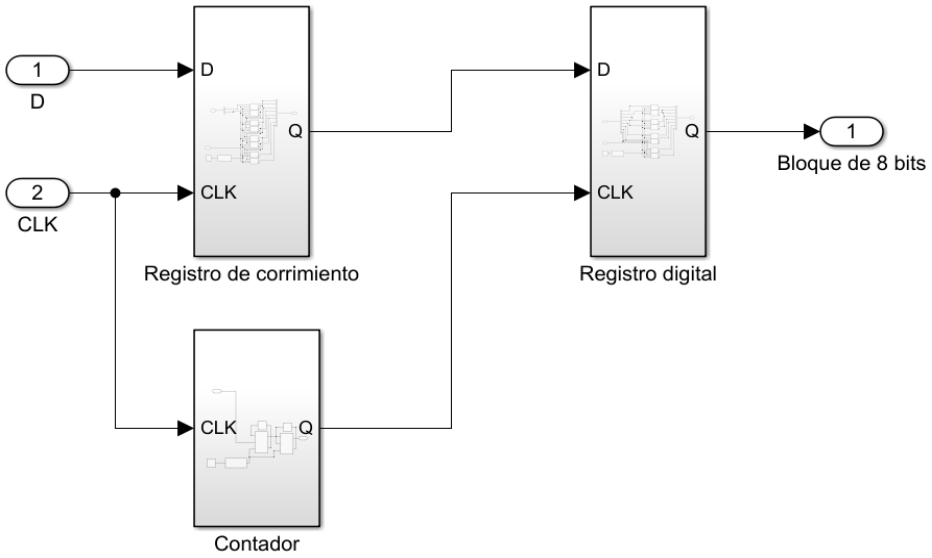
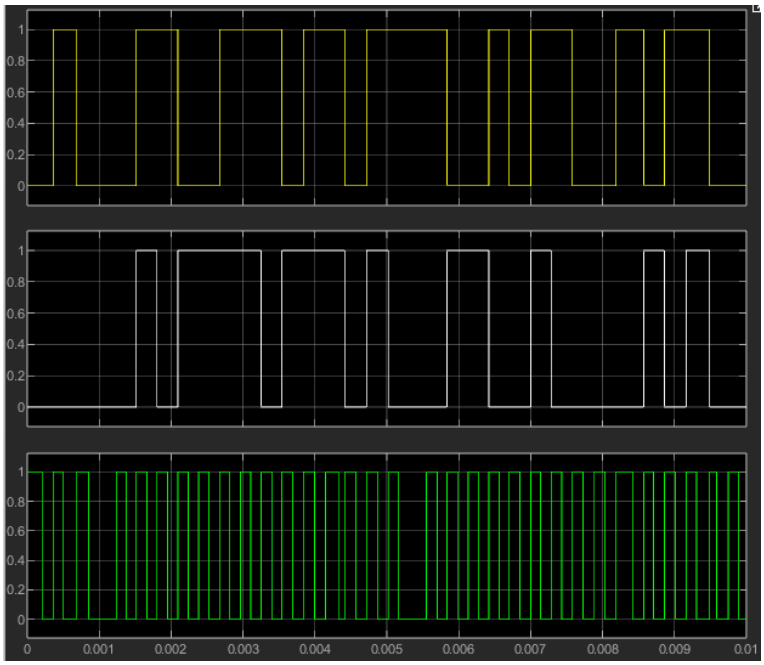


Figura 3.17. Subsistema “Salida del sistema”.

(a)



(b)

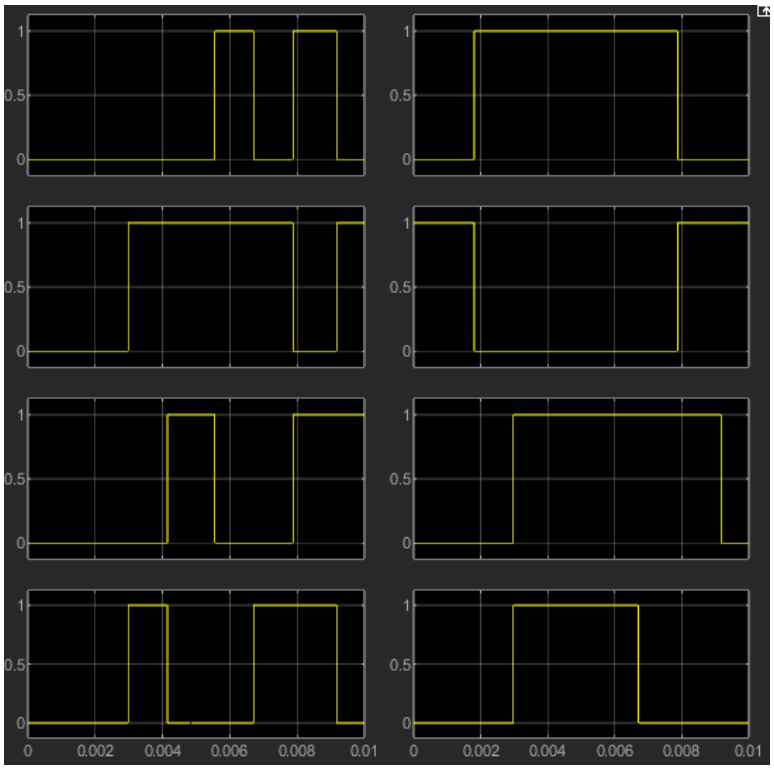
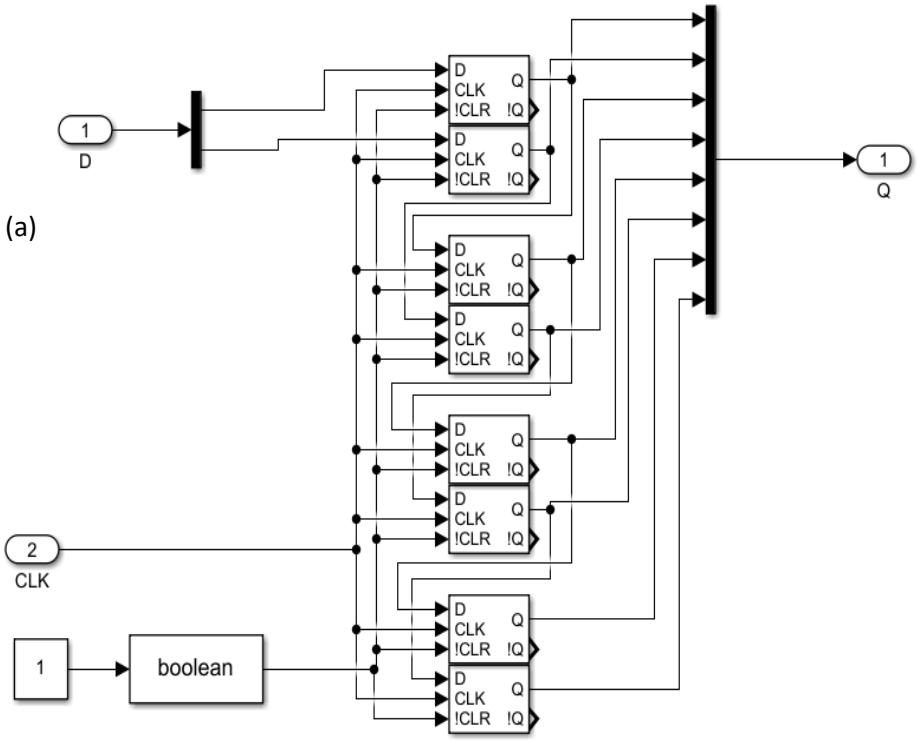


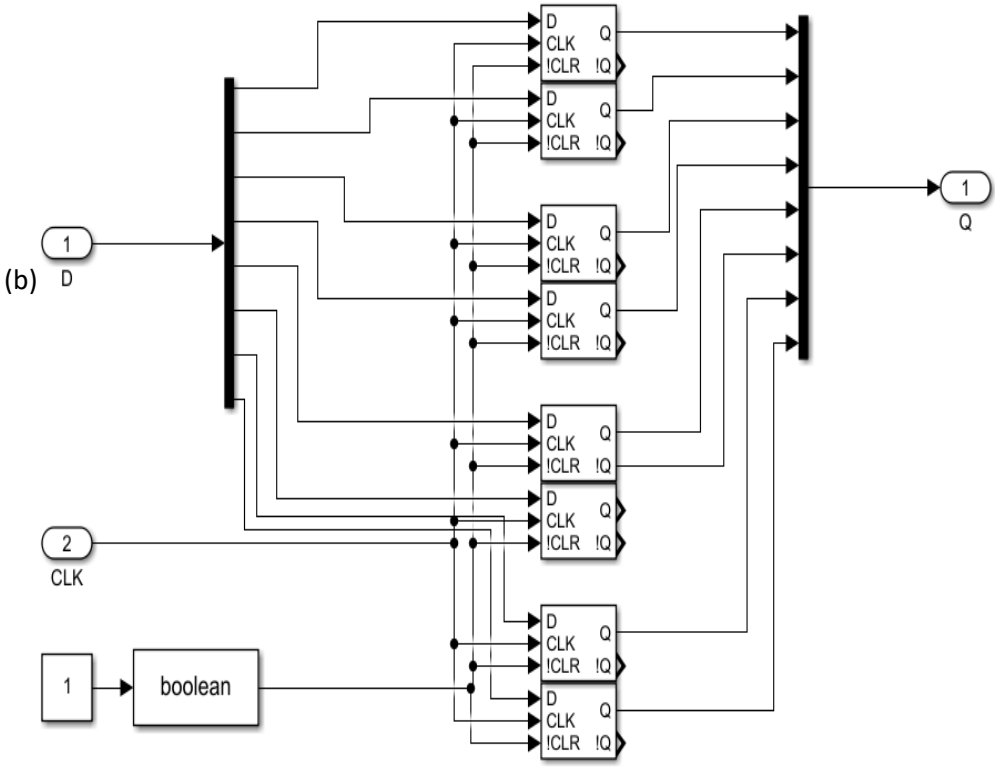
Figura 3.18. (a) Entrada y (b) Salida del subsistema “Salida del sistema”.

Como se observa en la figura 3.17, el subsistema “Salida del sistema” se compone de 3 subsistemas, un registro digital de 8 bits, un registro de corrimiento y contador de 2 bits; el interior de estos subsistemas se muestra en la figura siguiente.

(a)


1
D
D
CLK
!CLR
Q
!Q
1
Q
2
CLK
1
boolean

(b)


1
D
D
CLK
!CLR
Q
!Q
1
Q
2
CLK
1
boolean

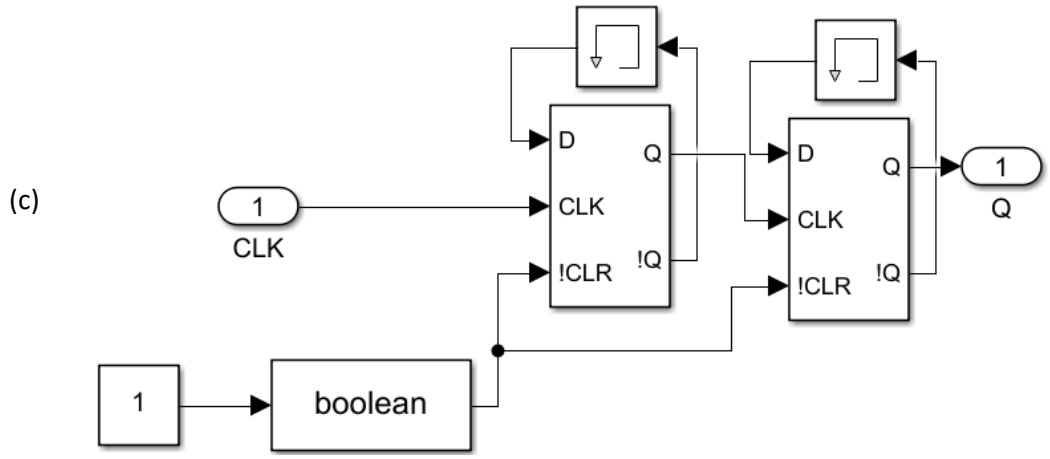


Figura 3.19. (a) Registro de corrimiento de 8 bits. (b) Registro digital de 8 bits. (c) Contador de 2 bits.

La salida de la simulación en Simulink es llevada al espacio de trabajo de Matlab por medio de los bloques “To workspace”; en Matlab se desarrolló un programa en texto (el cual se puede encontrar en el apéndice) que almacena estos bits como texto con el fin de desarrollar las pruebas estadísticas sobre la secuencia de bits encontrados.

3.2 Pruebas estadísticas NIST-FIPS.

La secuencia de bits obtenida debe ser evaluada por pruebas estadísticas que prueben su aleatoriedad, existen una gran cantidad de pruebas que se pueden realizar sobre numeros aleatorios en donde se busca que la secuencia se comporte como se debería esperar de una secuencia de numeros aleatorios.

El Instituto Nacional de Estándares y Tecnología o NIST por sus siglas en inglés, es una agencia federal de los Estados Unidos especializada en seguridad. NIST ha desarrollado distintos avances tecnología de la información, información cuántica, nanotecnología, ciberseguridad, entre otros. Uno de los propósitos de NIST es el de mejorar el sistema de medición, desarrollando nuevas y novedosas tecnologías.

Los Estándares para el Procesamiento de la Información Federal o FIPS por siglas en inglés, son estándares o guías para sistemas computacionales federales que son

desarrollados por el NIST de acuerdo con el Acto de Gestión de Seguridad para la Información Federal (FISMA) y aprobados por la secretaria del Comercio en Estados Unidos.

Dentro de los estándares FIPS existen pruebas evaluadoras para una secuencia de numeros aleatorios, las cuales son descritas en el estándar NIST 800-22 y serán usadas para evaluar la calidad de los numeros aleatorios generados. Las pruebas estadísticas NIST-FIPS son:

- Prueba del Monobit.
- Prueba frecuencia de bloque.
- Prueba de rachas.
- Prueba de sumas acumuladas.

En general, cada prueba tiene el propósito de evaluar una característica especifica que se puede esperar de una secuencia de bits aleatorios, en relación con las pruebas FIPS, para que la prueba en cuestión sea aprobada de forma satisfactoria el valor *P*, obtenido por medio de un algoritmo, debe ser cumplir con una característica especifica de cada prueba, de lo contrario la secuencia será considerada como no aleatoria.

Debe tomarse en cuanta también que es esperable que cierta cantidad de pruebas fallen debido a la naturaleza de las pruebas, según el estándar NIST 800-22, aproximadamente el 1% de las pruebas se espera que fallen. La cantidad de bits generados para las pruebas fue de 100,000, los cuales serán divididos en 100 partes iguales, es decir, tendremos 100 secuencias de 1000 bits, por lo tanto, tendremos 100 pruebas distintas.

Si bien es posibles seguir el estándar para realizar todas las pruebas estadísticas, NIST ofrece un programa en C para realizar estas pruebas, en el cual simplemente tendremos que cargar nuestra secuencia de bits en formato de texto, así como especificar el tamaño de la secuencia de bits (llamadas bitstreams) y la cantidad de secuencias que tenemos.

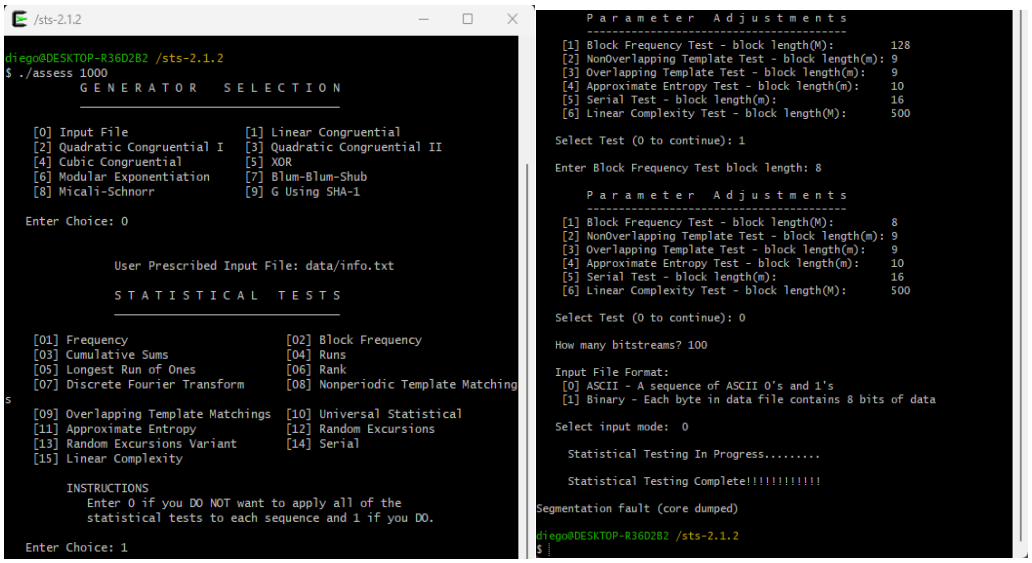


Figura 3.20. Ventana de comandos para realización de las pruebas NIST.

Las pruebas FIPS incluyen solo desde la prueba marcada como [1] hasta la prueba marcada en el programa como [4].

Los resultados de las pruebas son arrojados en dos documentos en texto, uno llamado "results.txt" con los valores *P* que cada secuencia de bits calculado y otro con el nombre de "stats.txt" el cual, además de contener los valores *P* calculados indica si la prueba fue completada con éxito o no.

3.2.1 Prueba del Monobit

Esta prueba medirá la proporción de ceros y unos de la secuencia entera de bits. *Su intención es determinar si la cantidad de ceros y unos* es aproximadamente la misma como se debería esperar de una secuencia de bits verdaderamente aleatoria. Esta es la prueba más básica y fundamental que se debe realizar sobre una secuencia de bits aleatorios, las demás pruebas dependen de la evaluación positiva de esta.

El desarrollo de esta prueba consiste en realizar la sumatoria

$$s_{obs} = \frac{|\sum_{i=1}^{n}(2\varepsilon_i - 1)|}{\sqrt{n}} \qquad 3.5$$

En donde ε_i es el i-ésimo valor binario de la secuencia de bits, *n* el largo de la secuencia de bits.

Para la obtención del valor *P* requiere de la función error definida por

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_{z}^{\infty} e^{-u^2} du \qquad 3.6$$

Y el valor de esta prueba *P* se calcula por

$$P = erfc\left(\frac{S_{obs}}{\sqrt{2}}\right) \qquad 3.7$$

La validez de esta prueba consiste en verificar que el valor de *P* es mayor a 0.01. Es recomendado por NIST que la secuencia consista en por lo menos 100 bits ($n \geq 100$).

Los valores *P* obtenidos usando las 100 secuencias de 1000 bits generadas son presentados en la tabla 3.1, en donde además de colocar el valor *P* de la prueba también se indica el numero secuencia y si este paso o no la prueba.

Valores *P* de la prueba del Monobit					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación
1	0.569214	Aprobado	**51**	0.800282	Aprobado
2	0.164104	Aprobado	**52**	0.800282	Aprobado
3	0.612882	Aprobado	**53**	0.612882	Aprobado
4	0.254945	Aprobado	**54**	0.949571	Aprobado
5	0.205903	Aprobado	**55**	1	Aprobado
6	0.375921	Aprobado	**56**	0.05778	Aprobado
7	0.282297	Aprobado	**57**	0.311572	Aprobado
8	1	Aprobado	**58**	0.75183	Aprobado
9	0.849515	Aprobado	**59**	0.949571	Aprobado
10	0.704336	Aprobado	**60**	0.100097	Aprobado
11	0.145767	Aprobado	**61**	0.205903	Aprobado

12	0.949571	Aprobado	**62**	0.129041	Aprobado
13	0.205903	Aprobado	**63**	0.849515	Aprobado
14	0.800282	Aprobado	**64**	0.022796	Aprobado
15	0.569214	Aprobado	**65**	0.282297	Aprobado
16	0.066636	Aprobado	**66**	0.899343	Aprobado
17	0.75183	Aprobado	**67**	0.410968	Aprobado
18	0.899343	Aprobado	**68**	0.899343	Aprobado
19	0.849515	Aprobado	**69**	0.375921	Aprobado
20	0.282297	Aprobado	**70**	0.410968	Aprobado
21	0.849515	Aprobado	**71**	0.704336	Aprobado
22	0.704336	Aprobado	**72**	0.229493	Aprobado
23	0.899343	Aprobado	**73**	0.569214	Aprobado
24	0.949571	Aprobado	**74**	0.657969	Aprobado
25	0.657969	Aprobado	**75**	0.282297	Aprobado
26	0.75183	Aprobado	**76**	0.949571	Aprobado
27	0.657969	Aprobado	**77**	0.342782	Aprobado
28	0.657969	Aprobado	**78**	0.282297	Aprobado
29	0.375921	Aprobado	**79**	0.410968	Aprobado
30	0.000637	**Reprobado**	**80**	1	Aprobado
31	0.657969	Aprobado	**81**	0.704336	Aprobado
32	0.447884	Aprobado	**82**	0.410968	Aprobado
33	0.899343	Aprobado	**83**	0.527089	Aprobado
34	0.899343	Aprobado	**84**	0.704336	Aprobado
35	0.447884	Aprobado	**85**	0.375921	Aprobado
36	1	Aprobado	**86**	1	Aprobado
37	0.949571	Aprobado	**87**	0.145767	Aprobado
38	0.75183	Aprobado	**88**	0.042985	Aprobado
39	0.229493	Aprobado	**89**	0.569214	Aprobado
40	0.254945	Aprobado	**90**	0.527089	Aprobado
41	0.016246	Aprobado	**91**	0.486616	Aprobado

42	0.342782	Aprobado	**92**	0.486616	Aprobado
43	0.013641	Aprobado	**93**	0.113846	Aprobado
44	0.569214	Aprobado	**94**	0.849515	Aprobado
45	0.100097	Aprobado	**95**	0.229493	Aprobado
46	0.704336	Aprobado	**96**	0.899343	Aprobado
47	0.100097	Aprobado	**97**	0.800282	Aprobado
48	0.113846	Aprobado	**98**	0.657969	Aprobado
49	0.229493	Aprobado	**99**	0.026857	Aprobado
50	0.447884	Aprobado	**100**	0.447884	Aprobado

Tabla 3.1. Valores *P* obtenidos por 100 pruebas Monobit.

En resumen, de las 100 pruebas de Monobit realizadas sobre secuencias de 1000 bits aleatorios, 99 fueron aprobados de forma satisfactoria y uno de ellos no.

3.2.2 Prueba frecuencia de bloque

La prueba frecuencia de bloque consiste en calcular la proporción de unos dentro de bloques de M-bits. *El propósito de esta prueba es la de comprobar que la frecuencia de unos en un bloque de M-bits sea aproximadamente igual a M/2*, como deberíamos esperar de una secuencia de bits aleatorios. Para un bloque de M=1, esta prueba es equivalente a la prueba del Monobit.

El desarrollo matemático de esta prueba consiste en evaluar la sumatoria

$$\pi_i = \frac{\sum_{j=i}^{M} \varepsilon_{(i-1)M+j}}{M}, \quad para\ 1 \leq i \leq \left\lfloor \frac{n}{M} \right\rfloor \qquad 3.8$$

Donde *n* es el largo de la secuencia de bits, M el largo de la secuencia de bits por cada bloque, ε el valor j-esimo binario en el i-esimo bloque.

El valor π obtenido por cada bloque resultara en una sumatoria

$$\chi^2_{obs} = 4M \sum_{i=1}^{\left\lfloor \frac{n}{M} \right\rfloor} \left(\pi_i - \frac{1}{2}\right)^2 \quad 3.9$$

Finalmente, el valor dado por (3.9) es introducido en la función Gamma incompleta dada para obtener el por *P* correspondiente

$$P = igamc\left(\frac{\left\lfloor \frac{n}{M} \right\rfloor}{2}, \frac{\chi^2_{obs}}{2}\right) \quad 3.10$$

La función Gamma incompleta está definida por

$$igamc(a,x) = \frac{1}{\boldsymbol{\Gamma}(a)} \int_x^{\infty} e^{-t} t^{a-1} dt \quad 3.11$$

Donde *igamc (a, 0) = 1* y *igamc (a, ∞) = 0*

Si el valor *P* dado por (3.10) es mayor o igual a 0.01 se concluye que la secuencia de bits es aleatoria, de lo contrario la secuencia se considera no aleatoria.

Es recomendado por NIST [4] que cada secuencia de bits consista en por lo menos 100 bits ($n \geq 100$). El tamaño de cada bloque debe ser elegido tal que M>0.01*n, además $\left\lfloor \frac{n}{M} \right\rfloor < 100$, debido a que el largo de secuencia de bits es de 1000, se tomó **M=128**, para cumplir con las condiciones antes dadas.

De forma similar a la prueba del Monobit, en la figura 3.2 se incluyeron los valores *P* obtenidos, el número de prueba y si esta aprobó o no la prueba.

Valores *P* de la prueba frecuencia de bloque					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación
1	0.633287	Aprobado	**51**	0.445328	Aprobado
2	0.029762	Aprobado	**52**	0.964846	Aprobado
3	0.218722	Aprobado	**53**	0.917394	Aprobado

4	0.228909	Aprobado	**54**	0.2664	Aprobado
5	0.118248	Aprobado	**55**	0.731975	Aprobado
6	0.241632	Aprobado	**56**	0.23733	Aprobado
7	0.51088	Aprobado	**57**	0.50732	Aprobado
8	0.873203	Aprobado	**58**	0.914879	Aprobado
9	0.779777	Aprobado	**59**	0.561806	Aprobado
10	0.848283	Aprobado	**60**	0.042035	Aprobado
11	0.409622	Aprobado	**61**	0.008795	Aprobado
12	0.561806	Aprobado	**62**	0.064851	Aprobado
13	0.731975	Aprobado	**63**	0.110196	Aprobado
14	0.05174	Aprobado	**64**	0.565511	Aprobado
15	0.739441	Aprobado	**65**	0.757946	Aprobado
16	0.003679	Reprobado	**66**	0.319018	Aprobado
17	0.845053	Aprobado	**67**	0.166697	Aprobado
18	0.976402	Aprobado	**68**	0.599184	Aprobado
19	0.44866	Aprobado	**69**	0.768918	Aprobado
20	0.1816	Aprobado	**70**	0.599184	Aprobado
21	0.352278	Aprobado	**71**	0.082647	Aprobado
22	0.58041	Aprobado	**72**	0.332594	Aprobado
23	0.821814	Aprobado	**73**	0.406465	Aprobado
24	0.629484	Aprobado	**74**	0.241632	Aprobado
25	0.092631	Aprobado	**75**	0.390906	Aprobado
26	0.210839	Aprobado	**76**	0.72823	Aprobado
27	0.838524	Aprobado	**77**	0.425633	Aprobado
28	0.479258	Aprobado	**78**	0.048471	Aprobado
29	0.70183	Aprobado	**79**	0.343745	Aprobado
30	0.024274	**Reprobado**	**80**	0.811542	Reprobado
31	0.375734	Aprobado	**81**	0.936212	Aprobado
32	0.532471	Aprobado	**82**	0.346573	Aprobado
33	0.46894	Aprobado	**83**	0.679024	Aprobado

34	0.503771	Aprobado	**84**	0.071351	Aprobado
35	0.558108	Aprobado	**85**	0.637094	Aprobado
36	0.682832	Aprobado	**86**	0.142761	Aprobado
37	0.724477	Aprobado	**87**	0.275878	Aprobado
38	0.259457	Aprobado	**88**	0.099536	Aprobado
39	0.743162	Aprobado	**89**	0.532471	Aprobado
40	0.340933	Aprobado	**90**	0.599184	Aprobado
41	0.3244	Aprobado	**91**	0.360955	Aprobado
42	0.057644	Aprobado	**92**	0.772551	Aprobado
43	0.239473	Aprobado	**93**	0.363879	Aprobado
44	0.134582	Aprobado	**94**	0.931729	Aprobado
45	0.610515	Aprobado	**95**	0.994309	Aprobado
46	0.419184	Aprobado	**96**	0.964846	Aprobado
47	0.059533	Aprobado	**97**	0.055811	Aprobado
48	0.024551	Aprobado	**98**	0.952599	Aprobado
49	0.049002	Aprobado	**99**	0.057027	Aprobado
50	0.007529	**Reprobado**	**100**	0.686638	Aprobado

Tabla 3.2. Valores *P* obtenidos por 100 pruebas de frecuencia de bloque.

En resumen, de las 100 pruebas de frecuencia de bloque realizadas sobre secuencias de 1000 bits aleatorios, 98 fueron aprobadas de forma satisfactoria.

3.2.3 Prueba de racha

La prueba de racha consiste el analizar si el largo de las "rachas" de bits 0 y 1 son como se debe esperar de una secuencia aleatoria, una racha se define como una secuencia ininterrumpida de bits iguales. En particular, *esta prueba determina si la oscilación entre unos y ceros es muy rápido o lento*.

El desarrollo matemático de esta prueba consiste inicialmente en aprobar que la prueba prerrequisito de frecuencia dada por NIST, en la cual es necesario demostrar que

$$\left|\pi - \frac{1}{2}\right| < \frac{2}{\sqrt{n}} \qquad 3.12$$

Donde *n* es el tamaño de la secuencia de bits, π es el valor obtenido tras evaluar $\left(\sum_{j=1}^{n} \varepsilon_j\right)$ y ε_j es el j-ésimo valor binario de la secuencia de bits. De lo contrario esta prueba no puede ser evaluada.

Si la prueba prerrequisito de frecuencia fue aprobada se llevará a cabo la siguiente prueba estadística para hallar el valor *P*.

$$V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1 \qquad 3.13$$

Donde *r(k)* es 0 si $\varepsilon_k = \varepsilon_{k+1}$, de lo contrario *r(k)* es 1.

Posterior se evalúa

$$P = erfc\left(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\pi(1-\pi)\sqrt{2n}}\right) \qquad 3.14$$

Si el valor de *P* dado por (3.14) es menor a 0.01 se concluye que la secuencia es no aleatoria, de lo contrario se concluye que la secuencia es aleatoria. Es recomendado por NIST que la secuencia de bits sea de por lo menos 100 bits ($n \geq 100$).

Los valores *P* encontrados para nuestras secuencias de bits son como siguen.

Valores *P* de la prueba de rachas.					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación
1	0.808149	Aprobado	**51**	0.446646	Aprobado
2	0.847638	Aprobado	**52**	0.570563	Aprobado
3	0.90573	Aprobado	**53**	0.285813	Aprobado
4	0.191013	Aprobado	**54**	0.447807	Aprobado
5	0.713312	Aprobado	**55**	0.849515	Aprobado
6	0.630126	Aprobado	**56**	0.989871	Aprobado
7	0.218781	Aprobado	**57**	0.775141	Aprobado
8	0.410968	Aprobado	**58**	0.058173	Aprobado

9	0.613668	Aprobado	**59**	0.52717	Aprobado
10	0.086817	Aprobado	**60**	0.103839	Aprobado
11	0.229931	Aprobado	**61**	0.007512	Reprobado
12	0.486535	Aprobado	**62**	0.696548	Aprobado
13	0.939406	Aprobado	**63**	0.229918	Aprobado
14	0.947956	Aprobado	**64**	0.212925	Aprobado
15	0.25054	Aprobado	**65**	0.978683	Aprobado
16	0.331018	Aprobado	**66**	0.704708	Aprobado
17	0.002422	**Reprobado**	**67**	0.673297	Aprobado
18	0.999596	Aprobado	**68**	0.999596	Aprobado
19	0.900241	Aprobado	**69**	0.595299	Aprobado
20	0.389931	Aprobado	**70**	0.540874	Aprobado
21	0.164434	Aprobado	**71**	0.483705	Aprobado
22	0.70768	Aprobado	**72**	0.10937	Aprobado
23	0.526752	Aprobado	**73**	0.808149	Aprobado
24	0.042971	Aprobado	**74**	0.284986	Aprobado
25	0.617171	Aprobado	**75**	0.071917	Aprobado
26	0.89683	Aprobado	**76**	0.447807	Aprobado
27	0.795457	Aprobado	**77**	0.121683	Aprobado
28	0.444096	Aprobado	**78**	0.638389	Aprobado
29	0.595299	Aprobado	**79**	0.080001	Aprobado
30	0.175551	Aprobado	**80**	0.704336	**Reprobado**
31	0.894419	Aprobado	**81**	0.748341	Aprobado
32	0.913727	Aprobado	**82**	0.095537	Aprobado
33	0.526752	Aprobado	**83**	0.440171	Aprobado
34	0.999596	Aprobado	**84**	0.313685	Aprobado
35	0.935039	Aprobado	**85**	0.010569	Aprobado
36	0.899343	Aprobado	**86**	0.375921	Aprobado
37	0.612969	Aprobado	**87**	0.609552	Aprobado
38	0.484594	Aprobado	**88**	0.565562	Aprobado

39	0.168882	Aprobado	**89**	0.381315	Aprobado
40	0.330948	Aprobado	**90**	0.560505	Aprobado
41	0.707993	Aprobado	**91**	0.149856	Aprobado
42	0.977274	Aprobado	**92**	0.304076	Aprobado
43	0.374441	Aprobado	**93**	0.623157	Aprobado
44	0.991823	Aprobado	**94**	0.164434	Aprobado
45	0.583396	Aprobado	**95**	0.189362	Aprobado
46	0.086817	Aprobado	**96**	0.612521	Aprobado
47	0.169798	Aprobado	**97**	0.254071	Aprobado
48	0.645746	Aprobado	**98**	0.101316	Aprobado
49	0.351105	Aprobado	**99**	0.591229	Aprobado
50	0.333316	Aprobado	**100**	0.884877	Aprobado

Tabla 3.3. Valores *P* obtenidos por 100 pruebas de racha.

En resumen, de las 100 pruebas de racha realizadas sobre secuencias de 1000 bits aleatorios, 98 fueron aprobadas de forma satisfactoria.

3.2.4 Prueba sumas acumuladas

El objeto de esta prueba es la de analizar el máximo desplazamiento (alejado del cero) generado por las sumas acumuladas ajustando la secuencia de bits de (0, 1) a (-1, +1). El propósito de esta prueba es la de determinar si la suma acumulada es muy grande o muy pequeña relativo a comportamiento esperado de una secuencia de bits aleatoria, para una secuencia aleatoria, el desplazamiento con respecto al cero debería ser largo.

El valor de las sumas acumuladas puede ser representado como

$$S_k = \sum_{i=1}^{k} (2\varepsilon_i - 1) \qquad 3.15$$

Donde ε_i es el valor del i-ésimo bit de la secuencia de bits y S_k es la suma acumulada desde el primer bit hasta el k-ésimo bit de se la secuencia de bits.

Después se debe hallar $z = \max_{1 \le k \le n} |S_k|$, donde z será el valor más grande de los valores absolutos de las sumas parciales S_k.

Finalmente, para hallar el valor aprobador *P* se resuelve

$$P = 1 - \sum_{j=\frac{\left(1-\frac{n}{z}\right)}{4}}^{\frac{\left(\frac{n}{z}-1\right)}{4}} \left[\Phi\left(\frac{z(4k+1)}{\sqrt{n}}\right) - \Phi\left(\frac{z(4k-1)}{\sqrt{n}}\right)\right] + \sum_{j=\left(-3-\frac{n}{z}\right)/4}^{\left(\frac{n}{z}-1\right)/4} \left[\Phi\left(\frac{z(4k+3)}{\sqrt{n}}\right) - \Phi\left(\frac{z(4k+1)}{\sqrt{n}}\right)\right] \qquad 3.16$$

Donde $\boldsymbol{\Phi}$ es la función de distribución de probabilidad estándar normal acumulativa definida por

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{u^2}{2}} du \qquad 3.17$$

Si el valor para *P* hallado por (3.16) es menor a 0.01 la secuencia se considera como no aleatoria, de lo contrario la secuencia se considera como aleatoria, además la secuencia debe consistir en al menos 100 bits. Los valores calculados para cada secuencia están en la tabla 3.4.

Valores *P* de la prueba de sumas acumuladas.					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación
1	0.876384	Aprobado	**51**	0.389382	Aprobado
2	0.73606	Aprobado	**52**	0.21359	Aprobado
3	0.038559	Aprobado	**53**	0.794732	Aprobado
4	0.187473	Aprobado	**54**	0.98972	Aprobado
5	0.411511	Aprobado	**55**	0.534965	Aprobado
6	0.850473	Aprobado	**56**	0.922381	Aprobado

7	0.175409	Aprobado	**57**	0.589898	Aprobado
8	0.411511	Aprobado	**58**	0.676715	Aprobado
9	0.227688	Aprobado	**59**	0.001274	**Reprobado**
10	0.242509	Aprobado	**60**	0.001274	**Reprobado**
11	0.534965	Aprobado	**61**	0.534965	Aprobado
12	0.483072	Aprobado	**62**	0.922381	Aprobado
13	0.434505	Aprobado	**63**	0.850473	Aprobado
14	0.242509	Aprobado	**64**	0.389382	Aprobado
15	0.922381	Aprobado	**65**	0.389382	Aprobado
16	0.922381	Aprobado	**66**	0.483072	Aprobado
17	0.922381	Aprobado	**67**	0.876384	Aprobado
18	0.994708	Aprobado	**68**	0.765607	Aprobado
19	0.958243	Aprobado	**69**	0.434505	Aprobado
20	0.850473	Aprobado	**70**	0.411511	Aprobado
21	0.200192	Aprobado	**71**	0.676715	Aprobado
22	0.115559	Aprobado	**72**	0.676715	Aprobado
23	0.706355	Aprobado	**73**	0.647327	Aprobado
24	0.647327	Aprobado	**74**	0.706355	Aprobado
25	0.291508	Aprobado	**75**	0.794732	Aprobado
26	0.107464	Aprobado	**76**	0.508616	Aprobado
27	0.368117	Aprobado	**77**	0.434505	Aprobado
28	0.562079	Aprobado	**78**	0.058223	Aprobado
29	0.876384	Aprobado	**79**	0.291508	Aprobado
30	0.647327	Aprobado	**80**	0.483072	Reprobado
31	0.049508	Aprobado	**81**	0.032493	Aprobado
32	0.107464	Aprobado	**82**	0.011876	Aprobado
33	0.823133	Aprobado	**83**	0.434505	Aprobado
34	0.994708	Aprobado	**84**	0.291508	Aprobado
35	0.971736	Aprobado	**85**	0.024965	Aprobado
36	0.994708	Aprobado	**86**	0.017346	Aprobado

37	0.765607	Aprobado	**87**	0.589898	Aprobado
38	0.922381	Aprobado	**88**	0.508616	Aprobado
39	0.562079	Aprobado	**89**	0.153163	Aprobado
40	0.227688	Aprobado	**90**	0.16398	Aprobado
41	0.258072	Aprobado	**91**	0.794732	Aprobado
42	0.368117	Aprobado	**92**	0.982178	Aprobado
43	0.765607	Aprobado	**93**	0.092691	Aprobado
44	0.434505	Aprobado	**94**	0.187473	Aprobado
45	0.922381	Aprobado	**95**	0.142934	Aprobado
46	0.971736	Aprobado	**96**	0.227688	Aprobado
47	0.982178	Aprobado	**97**	0.187473	Aprobado
48	0.958243	Aprobado	**98**	0.389382	Aprobado
49	0.434505	Aprobado	**99**	0.200192	Aprobado
50	0.823133	Aprobado	**100**	0.73606	Aprobado

Tabla 3.4. Valores *P* obtenidos por 100 pruebas de sumas acumuladas.

De las 100 pruebas de sumas acumuladas realizadas sobre secuencias de 1000 bits aleatorios, 98 fueron aprobadas de forma satisfactoria.

La media de la cantidad de pruebas aprobadas es igual a 98.25, es decir, las pruebas realizadas tienen un porcentaje de fallo del 1.75%, que es aproximadamente lo uno podría esperar y así fue indicado por el estándar NIST 800-22.

Tomando el hecho anterior, podemos concluir que ***la secuencia de numeros aleatorios, obtenida por simulación, fue validada como tal*** y que el modelo del TRNG simulado en Simulink funciona de forma adecuada.

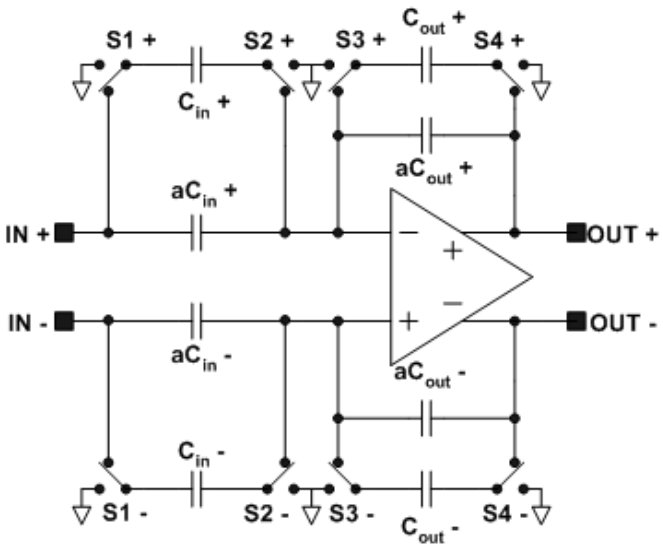
Capítulo 4

Implementación del TRNG en FPAA's

4. Implementación del TRNG en FPAA´s.

La metodología que se usó para el desarrollo del necesito de una gran cantidad de simulaciones y ajustes sobre los parámetros del modelo, la implementación de una metodología así requeriría la implementación de varios circuitos analógicos distintos, en los cuales serían necesario una gran cantidad de ajustes, esto no solo afectaría el tiempo para la realización la implementación, sino la cantidad económica que la implementación requeriría. Todos estos problemas son resueltos por medio de un hardware embebido en donde podremos programar en una tarjeta los circuitos analógicos que describen nuestro modelo del TRNG.

Un FPAA es una matriz de circuitos analógicos programables el cual contiene un conjunto de grupos de bloques analógicos programables interconectados (CAB's), existen distintos FPAA's en el mercado, el usado en este trabajo será el FPAA AN231E04 desarrollada por Anadigm, los cuales utilizan la tecnología de capacitores conmutados, la cual nos permite imitar impedancias por medio de un capacitor e interruptores, además, la tarjeta funciona con una señal diferencial, reduciendo el ruido en el sistema, en la figura 4.1 se observa el diagrama del circuito de una ganancia inversora.



$$\frac{V_{OUT}(s)}{V_{IN}(s)} = -G$$

Figura 4.1. Diagrama del circuito ganancia inversora y su ecuación de diseño.

La tarjeta ASDB también desarrollada por Anadigm, contiene 4 FPAA's AN231E04, un PIC32, así como otros dispositivos activos y pasivos que permiten el desarrollo de circuitos analógicos.

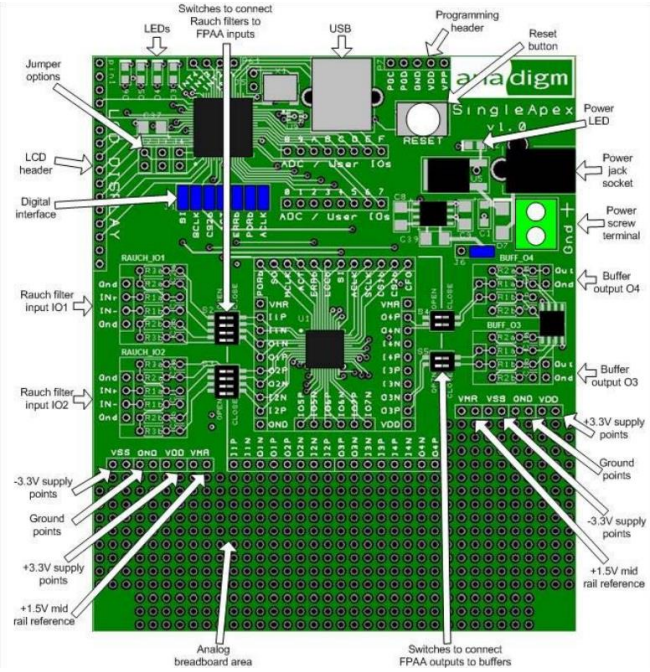
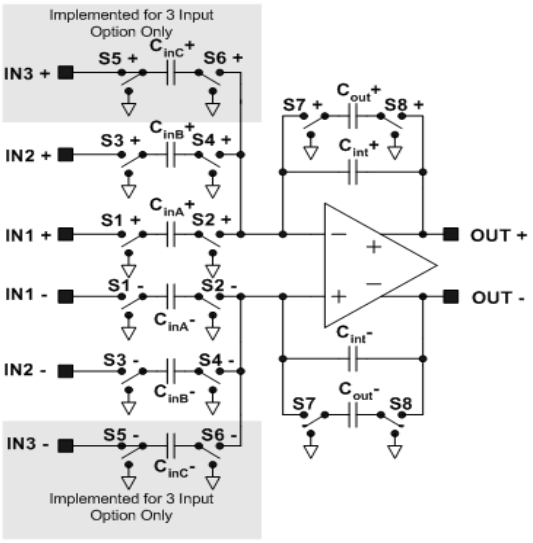


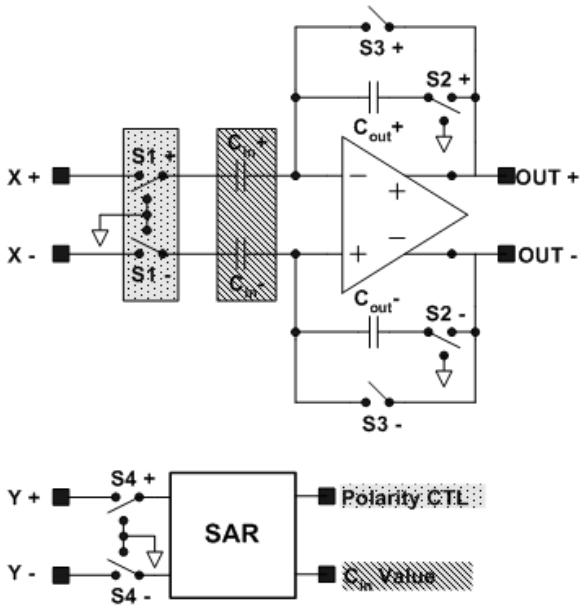
Figura 4.2. Vista superior de la tarjeta Anadigm SingleApex Development Board.

La programación en la tarjeta es llevada a cabo por el software "AnadigmDesigner2" dado por la empresa distribuidora, esta es una programación en bloques llamados CAM's o módulos analógicos configurables, cada CAM tiene su función de transferencia equivalente, para lo relacionado a este trabajo se utilizaron los CAM's: sumador/restador filtro pasa bajas, multiplicador, sumador inversor, comparadores y un rectificador de media onda. En las figuras siguientes se colocaron los diagramas de los circuitos, así como las ecuaciones de diseño de los CAM's a utilizar.



$$V_{out}(s) = \frac{2\pi f_0(\pm G_1 V_{in1} \pm G_2 V_{in2} \pm G_3 V_{in3})}{s + 2\pi f_0} \quad (4.1)$$

Figura 4.3. Diagrama del circuito y ecuación de diseño del sumador/restador filtro pasa bajas.

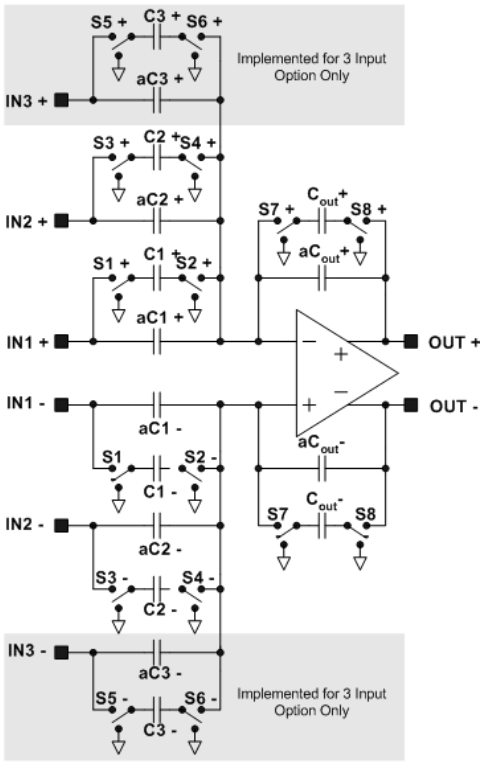


$$V_{out} = M\mathrm{V_x V_y}\ (cuantificados) \quad (4.2)$$

Donde M es el factor de multiplicación, los valores para el capacitor V_{out} satisfacen

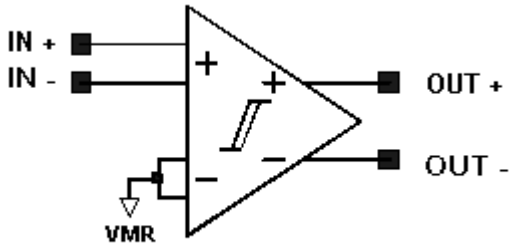
$$C_{out} = \frac{85}{M} \quad (4.3)$$

Figura 4.4. Diagrama del circuito y ecuación de diseño del multiplicador.



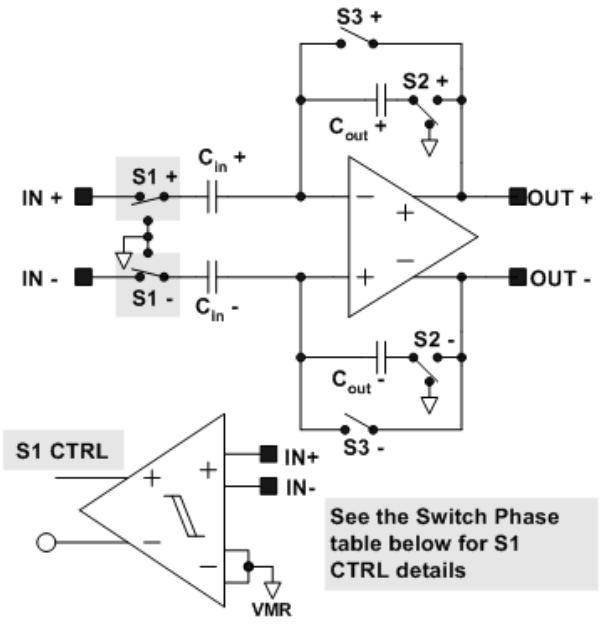
$$V_{out} = -G_1 V_{in1} - G_2 V_{in2} - G_3 V_{in3} \quad (4.4)$$

Figura 4.5. Diagrama del circuito y ecuación de diseño del sumador inversor.



$$V_{out} = 2V \quad si \quad V_{in} > V_{Hyst}$$
$$V_{out} = 0 \quad si \quad V_{in} < -V_{Hyst} \quad (4.5)$$

Figura 4.6. Diagrama del circuito y ecuación de diseño del comparador.



$$V_{out} = \pm G|V_{in}| \tag{4.6}$$

Donde G es una ganancia asignable.

Figura 4.7. Diagrama del circuito y ecuación de diseño del rectificador de media onda.

Por medio de los CAM's antes descritos seremos capaces de desarrollar las primeras dos etapas del modelo descrito en la figura 3.4, es decir, la etapa de generación de entropía y del acondicionamiento.

4.1 Programación y simulación AnadigmDesigner2

El generador de entropía del TRNG se compone de los sistemas caóticos escalados en magnitud expresados por las ecuaciones (3.3) para el SCC y (3.4) para el sistema hipercaótico de Lorenz, para poder expresar estos sistemas en CAM's deben estar en el espacio s usando la transformada de Laplace, cuya definición viene dada por

$$\mathcal{L}_+ \{f(t)\} = F(s) = \int_0^\infty e^{-st} f(t)dt \tag{4.7}$$

La cual cumple las propiedades de la superposición, la transformada de una derivada con condiciones iniciales nulas viene dada por

$$\mathcal{L}_+ \left\{\frac{df(t)}{dt}\right\} = sF(s) \tag{4.8}$$

Aplicando la transformada de Laplace entonces sobre el sistema caótico de Chua escalado con función lineal cubica y parámetros α=10 y β=14.87 tenemos

$$X(s) = \frac{2.4Y(s) + 2.7X(s) - 3.925X(s)^3}{s+1}$$

$$Y(s) = \frac{4.16X(s) + 7Z(s)}{s+1} \qquad 4.10$$

$$Z(s) = \frac{-2.1Y(s) + Z(s)}{s+1}$$

Nótese que el sistema de ecuaciones de expreso de forma que la forma de cada ecuación coincida con la ecuación de diseño del CAM sumador/restador de un filtro pasa bajas mostrada (4.1). Para e sistema hipercaótico de Lorenz escalado tenemos

$$X(s) = \frac{5Y(s) - 4X(s) - W(s)}{s+1}$$

$$Y(s) = \frac{20X(s)Z(s)}{s+1} \qquad 4.11$$

$$Z(s) = \frac{20 - X(s)Y(s)}{s+1}$$

$$W(s) = \frac{0.1Y(s) + 0.9W(s)}{s+1}$$

AnadigmDesigner2 no solo te permite programar las tarjetas Anadigm, sino también realizar simulaciones de los circuitos analógicos, estas serán utilizadas para cerciorarnos que el circuito diseñado funciona como es esperable. Esta simulación no acepta retroalimentación sobre el mismo bloque, por lo que, se añadió un bloque de "Delay" que permita la retroalimentación cuando sea necesario.

Usando los CAM's descritos con anterioridad se forma el siguiente diagrama de bloques en él se describió el sistema caótico de Chua, los parámetros de cada bloque y la configuración de cada FPAA se encuentran en el apéndice B.

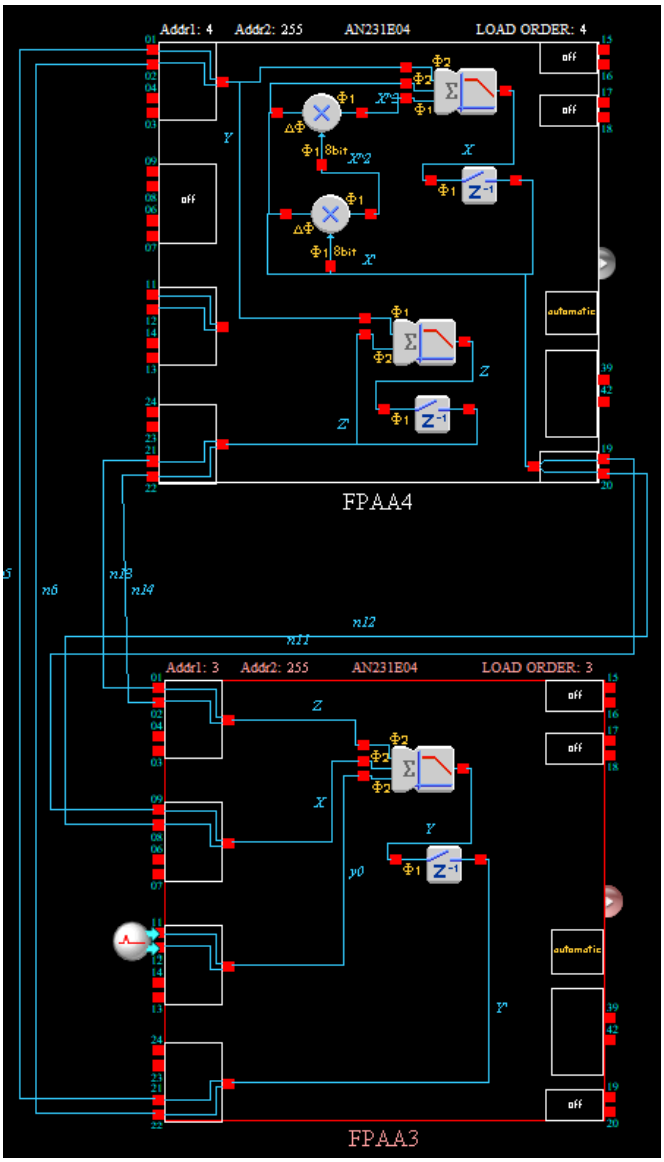


Figura 4.8. Diagrama en bloques del sistema caótico de Chua en AnadigmDesigner2.

Para la simulación se forzó la condición inicial $y_0 = 0.1V$ por medio de una señal impulso por un tiempo de 100ms.

De forma similar también se realizó la simulación del sistema hipercaótico de Lorenz escalado, el diagrama de bloques es presentado en la figura 4.10 y su simulación en la figura siguiente.

Figura 4.9. Simulación del sistema caótico de Chua en AnadigmDesigner2 para las variables (a)x, (b) y (c) z.

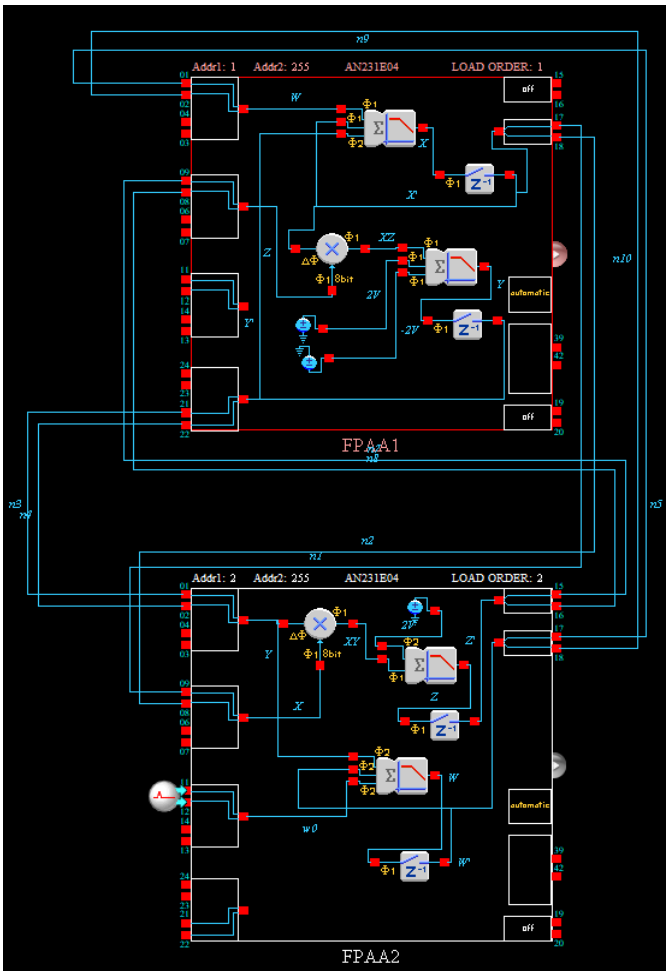


Figura 4.10. Diagrama en bloques del sistema hipercaótico de Lorenz en AnadigmDesigner2

Figura 4.11. Simulación del sistema hipercaótico de Lorenz en AnadigmDesigner2 para las variables (a)x, (b) y (c) z (d) w.

En la etapa de generación de entropía se introdujo un bloque al cual denominamos como "operaciones" en el cual desacoplábamos las variables de estado del sistema hipercaótico de Lorenz y aplicábamos un valor absoluto, de la misma forma se realizarán dichas operaciones en la implementación y simulación en AnadigmDesigner2. La suma de las variables será realizada por el bloque de suma inversora, mostrado en la figura 4.5, y el valor absoluto es realizado por el bloque rectificador de media onda mostrado en la figura 4.7, el nuevo diagrama en AnadigDesigner2 del sistema caótico de Lorenz escalado y su simulación es como se muestra en la figura siguiente.

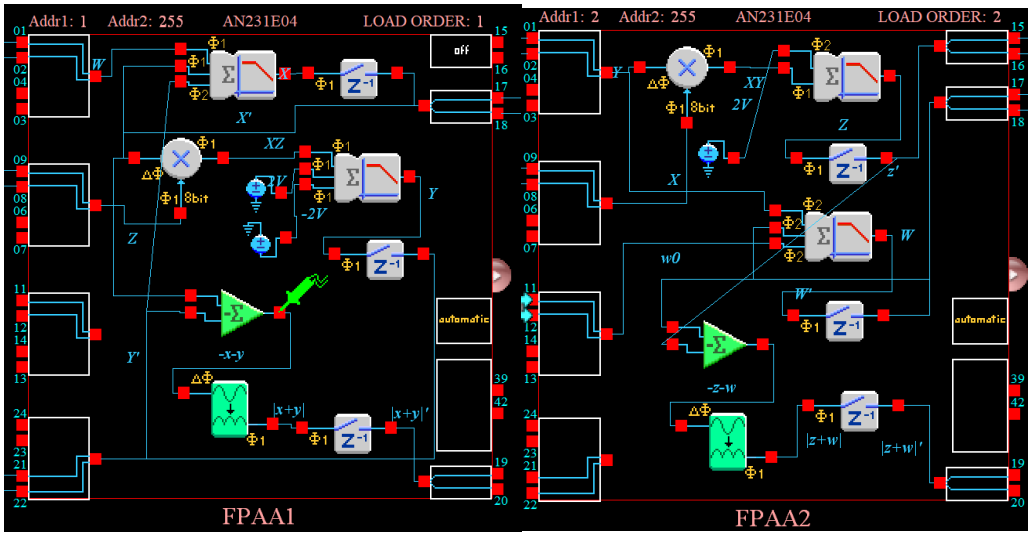


Figura 4.12. Diagrama de bloques del sistema hipercaótico de Lorenz en AnadigmDesigner2 con etapa de operaciones.

$-x-y$

$|x+y|$

$-z-w$

$|z+w|$

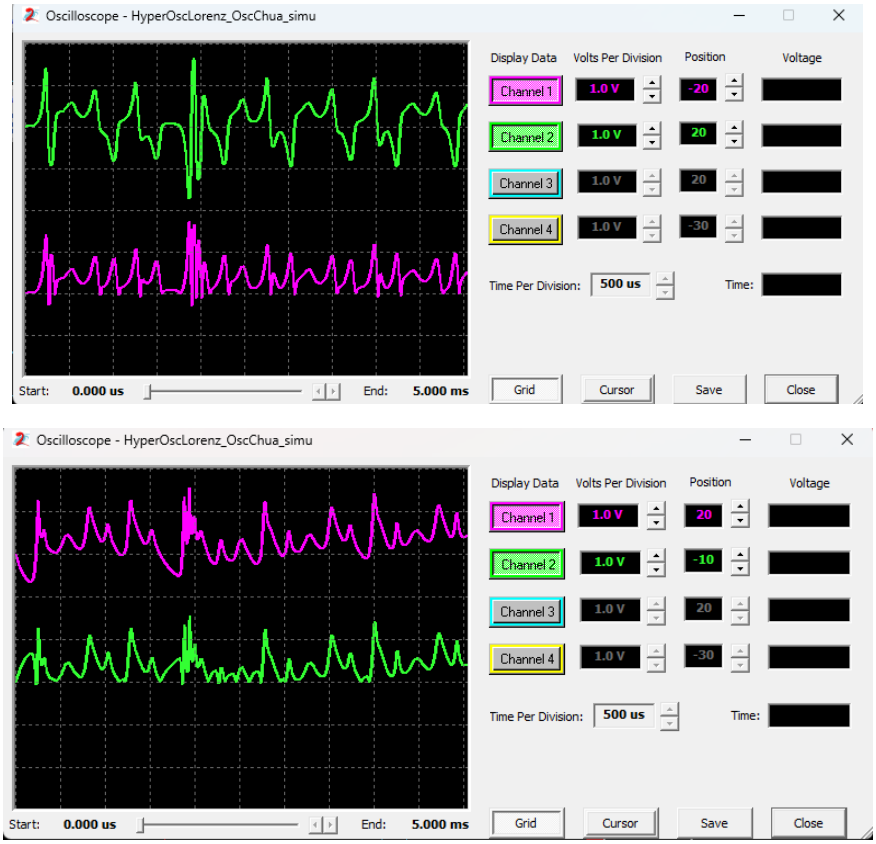


Figura 4.13. Simulación del sistema hipercaótico de Lorenz en AnadigmDesigner2 con etapa de operaciones.

La etapa dos del TRNG usa comparadores, los FPAA contienen comparadores, este puede ser variable o con respecto a tierra, los valores correspondientes a los parámetros ya fueron hallados en el capítulo 3, su simulación es la siguiente.

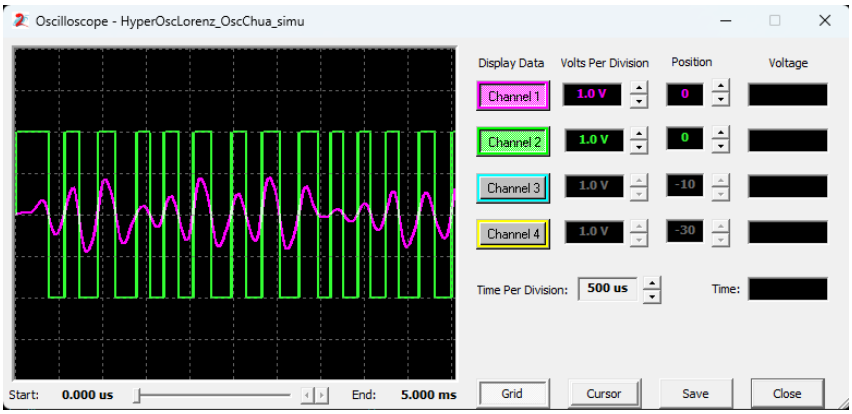


Figura 4.14. Simulación tras comparación de la salida *y* del sistema caótico de Chua.

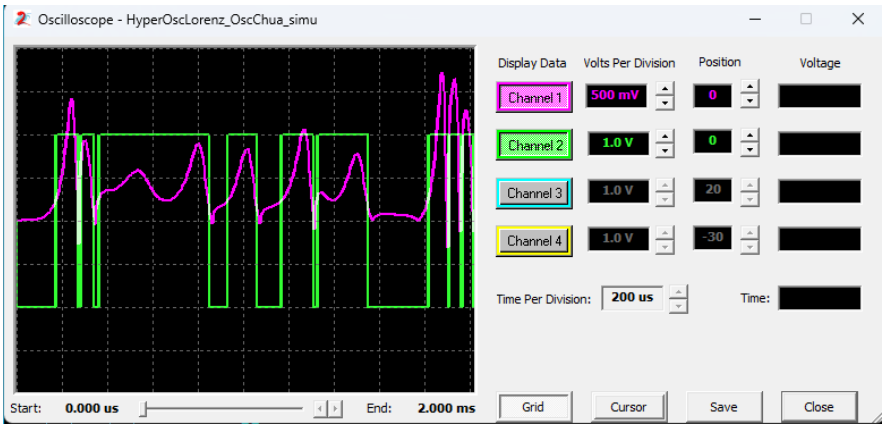


Figura 4.15. Simulación tras comparación de la salida $|x+y|$ del sistema hipercaótico de Lorenz.

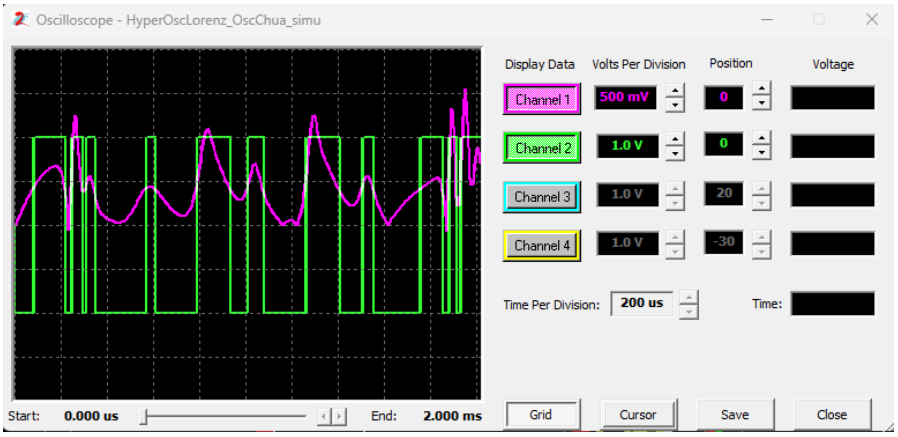
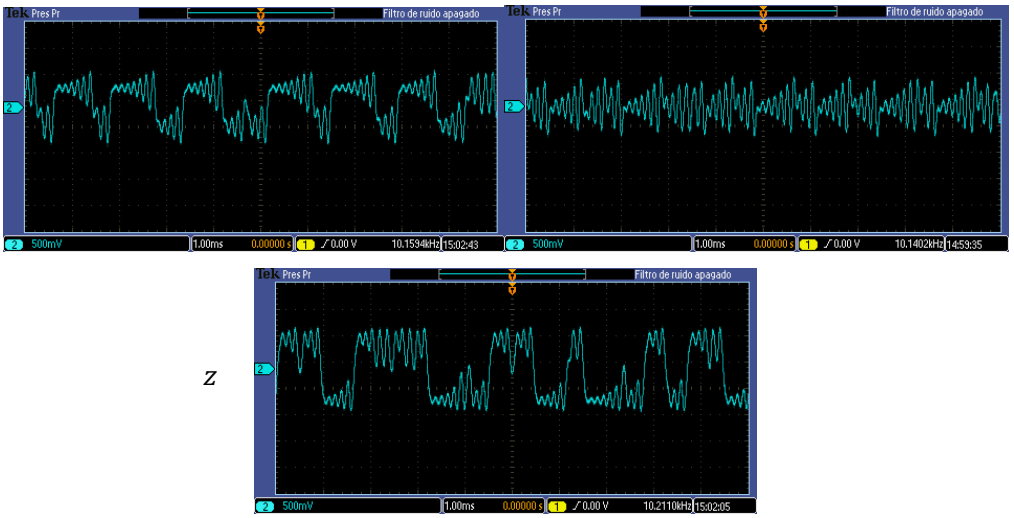


Figura 4.16. Simulación tras comparación de la salida $|z+w|$ del sistema hipercaótico de Lorenz.

4.2 Implementación del TRNG

Al mismo tiempo que AnadigmDesigner2 nos permite simular los circuitos analógicos sobre FPAA's también podemos programar la tarjeta ASDB. Se tomo evidencia en un osciloscopio de cada una de las variables de estados de los sistemas caóticos, así como, incluyendo las operaciones necesarias para el desacoplamiento de las variables de estado. La configuración y parámetros utilizados se encuentran en la sección del apéndice B de esta tesis.

x

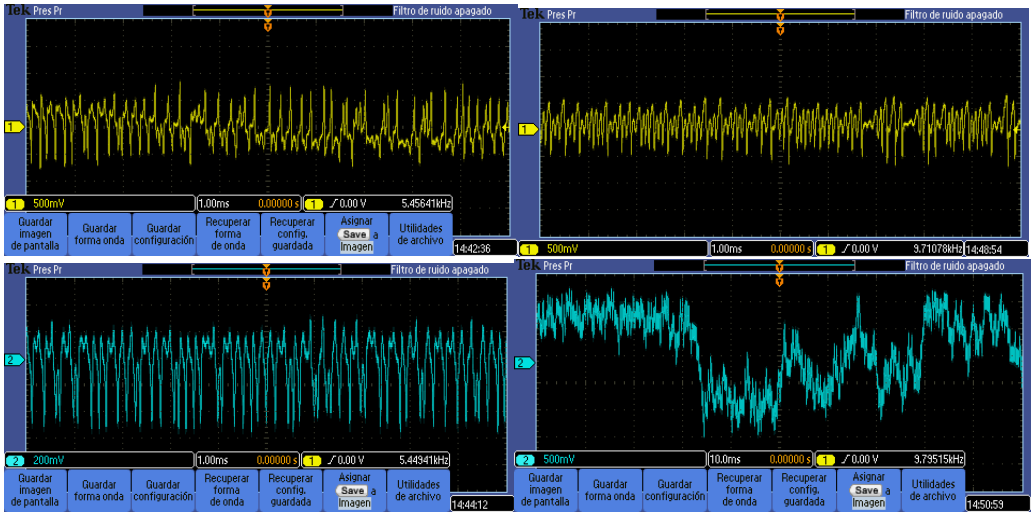


y

z

Figura 4.17. Variables de estado vistas desde un osciloscopio del SCCE implementado en FPAA.

x



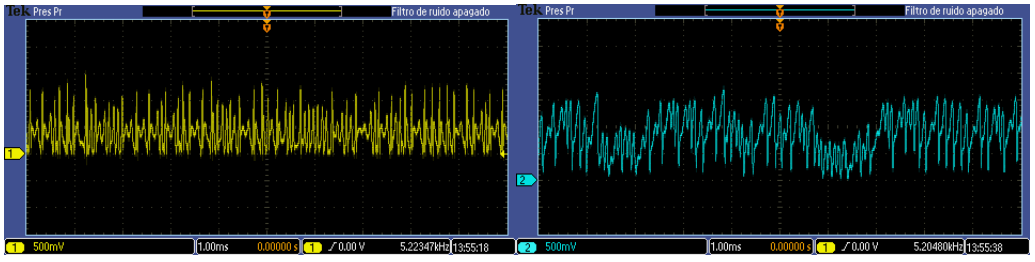
y

z

w

Figura 4.18. Variables de estado vistas desde un osciloscopio del SHLE implementado en FPAA.

$|x + y|$



$|z + w|$

Figura 4.19. Resultado desde un osciloscopio de las operaciones |x+y| y |z+w| del SHLE implementado en FPAA.

Las figuras anteriores concluyen la primera etapa del TRNG, la etapa posterior consiste en una comparación, el osciloscopio nos permitirá observa el comportamiento deseado de forma más detallada. Para la variable de estado "y" del SCC la comparación se realiza con respecto a la tierra, es decir, para valores mayores a 0 tendremos un 1 lógico, como se observa en la figura.

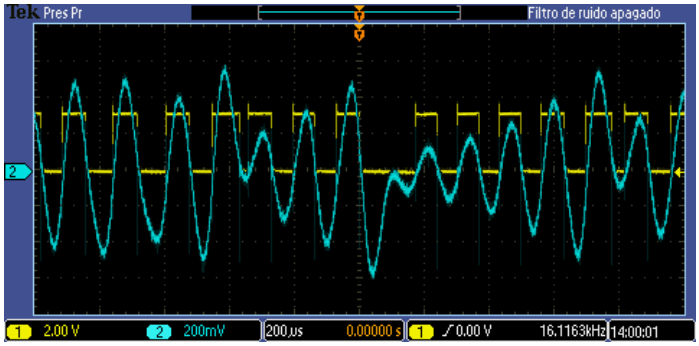


Figura 4.20. Resultado en el osciloscopio tras comparación de la señal de la variable de estado "y" del SCC.

Las salidas obtenidas tras la comparación después de la etapa de operaciones se observan en la figura 4.21, para este caso la comparación es con respecto a un parámetro, 0.21 para el caso de |x+y| y 0.404 para el caso de |z+w|, estos parámetros fueron resaltados por medio de un cursor.

$|x + y|$

$|z + w|$

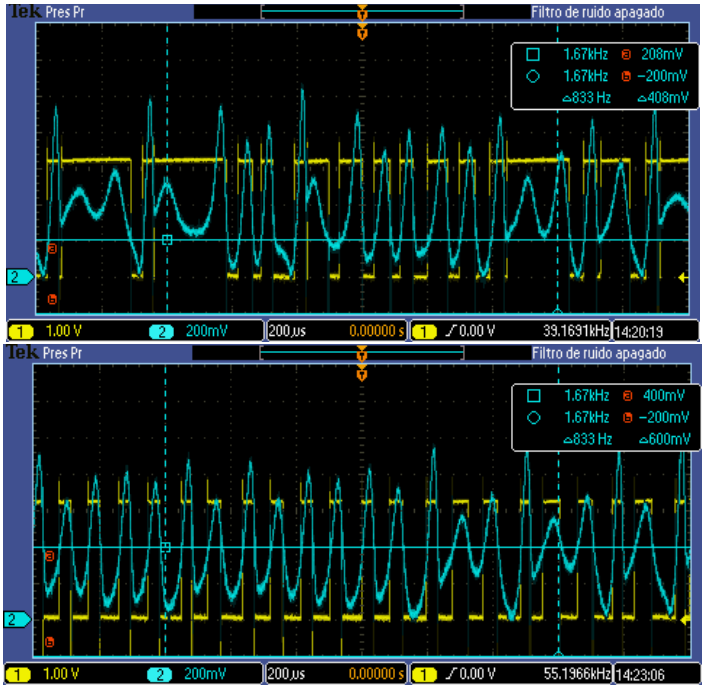


Figura 4.21. Resultado en el osciloscopio tras comparación entre un parámetro establecido y las operaciones |x+y| y |z+w|.

Como se mencionó, los FPAA trabajan con una señal diferencia, por lo que para trabajar con señales de nodo común es necesaria una conversión, las celdas del FPAA AN231E04 contiene salidas digitales, las cuales serán utilizadas para el registro de datos.

La etapa 3 del TRNG conlleva el uso de un registro digital para obtener los bloques de 2 bits, a la salida de este obtendremos una secuencia de bloques de 2 bits aleatorios, con el fin de realizar la etapa 3, la cual conlleva el uso de un registro digital para la obtención del tren de bloques de 2 bits y el registro de datos se utilizará un Arduino Nano con conexión serial a un computador, el código usado para la recopilación de datos está en el apéndice C. La figura 4.22 nos permite ver en bloques como será repartido el trabajo del TRNG entre la tarjeta ASDB y el Arduino Nano.

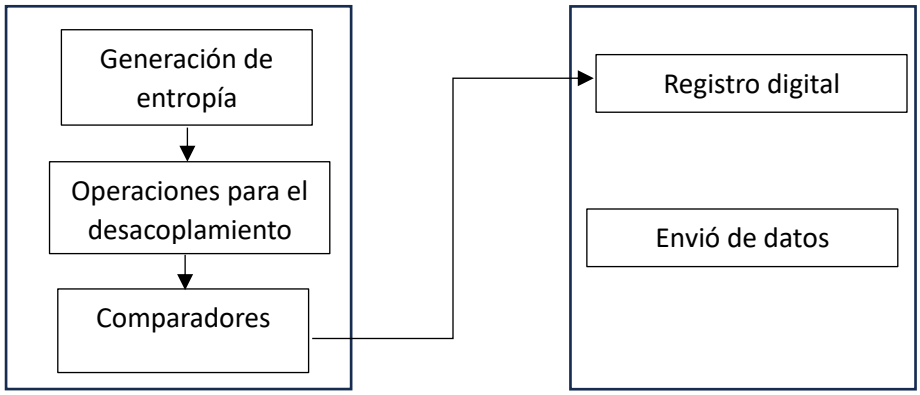


Figura 4.22. Diagrama de bloques trabajo realizado por las tarjetas.

4.3 Pruebas estadísticas NIST-FIPS sobre la implementación.

De la misma manera en cómo se realizaron las pruebas estadísticas en la simulación del TRNG, se obtuvieron un total de 100, 000 bits divididos en secuencias de 1000 bits para los cuales se realizaron 100 pruebas estadísticas, recordando las pruebas NIST-FIPS, se

evaluó la secuencia de bits con las pruebas del Monobit, frecuencia en bloques, prueba de racha y sumas acumuladas, las cuales ya fueron explicadas a detalle en el capítulo 3.

Los valores *P* encontrados para las 100 pruebas estadísticas sobre la secuencia de bits se presentan en las siguientes tablas, así como la aprobación o desaprobación en cada una de las pruebas de cada secuencia de bits.

Valores *P* de la prueba del Monobit.					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación
1	0.016246	Aprobado	**51**	0.076531	Aprobado
2	1	Aprobado	**52**	0.899343	Aprobado
3	0.066636	Aprobado	**53**	0.036879	Aprobado
4	0.800282	Aprobado	**54**	0.164104	Aprobado
5	0.05778	Aprobado	**55**	0.949571	Aprobado
6	0.164104	Aprobado	**56**	0.311572	Aprobado
7	0.016246	Aprobado	**57**	0.949571	Aprobado
8	0.022796	Aprobado	**58**	0.612882	Aprobado
9	0.311572	Aprobado	**59**	0.145767	Aprobado
10	0.05778	Aprobado	**60**	0.036879	Aprobado
11	0.042985	Aprobado	**61**	0.129041	Aprobado
12	0.019279	Aprobado	**62**	0.254945	Aprobado
13	0.042985	Aprobado	**63**	0.184126	Aprobado
14	0.527089	Aprobado	**64**	0.096537	Aprobado
15	0.205903	Aprobado	**65**	0.129041	Aprobado
16	0.076581	Aprobado	**66**	0.184126	Aprobado
17	0.184126	Aprobado	**67**	0.036879	Aprobado
18	0.375921	Aprobado	**68**	0.013641	Aprobado
19	0.899343	Aprobado	**69**	0.066636	Aprobado
20	0.113846	Aprobado	**70**	0.013641	Aprobado
21	0.006537	**Reprobado**	**71**	0.019279	Aprobado
22	0.129041	Aprobado	**72**	0.75183	Aprobado

23	0.410968	Aprobado	**73**	0.076581	Aprobado
24	0.042985	Aprobado	**74**	0.113846	Aprobado
25	0.527089	Aprobado	**75**	1	Aprobado
26	0.205903	Aprobado	**76**	0.009512	**Reprobado**
27	0.016246	Aprobado	**77**	0.311572	Aprobado
28	0.145767	Aprobado	**78**	0.04427	Aprobado
29	0.066636	Aprobado	**79**	0.076581	Aprobado
30	0.254945	Aprobado	**80**	0.164104	Aprobado
31	0.342782	Aprobado	**81**	0.031528	Aprobado
32	0.486616	Aprobado	**82**	0.031528	Aprobado
33	0.100097	Aprobado	**83**	0.949571	Aprobado
34	0.527089	Aprobado	**84**	0.013641	Aprobado
35	0.100097	Aprobado	**85**	0.949571	Aprobado
36	0.704336	Aprobado	**86**	0.145767	Aprobado
37	0.800282	Aprobado	**87**	0.066636	Aprobado
38	0.026857	Aprobado	**88**	0.612882	Aprobado
39	0.164104	Aprobado	**89**	0.704336	Aprobado
40	0.129041	Aprobado	**90**	0.342782	Aprobado
41	0.704336	Aprobado	**91**	0.342782	Aprobado
42	0.042985	Aprobado	**92**	0.012937	Aprobado
43	0.254945	Aprobado	**93**	0.75183	Aprobado
44	0.019279	Aprobado	**94**	0.019279	Aprobado
45	0.129041	Aprobado	**95**	0.375921	Aprobado
46	0.949571	Aprobado	**96**	0.076581	Aprobado
47	0.282297	Aprobado	**97**	0.026857	Aprobado
48	0.486616	Aprobado	**98**	0.087705	Aprobado
49	0.129041	Aprobado	**99**	0.076581	Aprobado
50	0.849515	Aprobado	**100**	0.410968	Aprobado

Tabla 4.1. Valores *P* obtenidos por 100 pruebas del Monobit en la implementación del TRNG.

Valores *P* de la prueba de frecuencia de bloque.					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación
1	0.087059	Aprobado	**51**	0.124312	Aprobado
2	0.349418	Aprobado	**52**	0.885002	Aprobado
3	0.261756	Aprobado	**53**	0.305846	Aprobado
4	0.938396	Aprobado	**54**	0.472366	Aprobado
5	0.599184	Aprobado	**55**	0.514452	Aprobado
6	0.144165	Aprobado	**56**	0.465526	Aprobado
7	0.114735	Aprobado	**57**	0.606733	Aprobado
8	0.257175	Aprobado	**58**	0.599184	Aprobado
9	0.779777	Aprobado	**59**	0.121854	Aprobado
10	0.489691	Aprobado	**60**	0.241632	Aprobado
11	0.675214	Aprobado	**61**	0.671403	Aprobado
12	0.099536	Aprobado	**62**	0.857823	Aprobado
13	0.70183	Aprobado	**63**	0.403322	Aprobado
14	0.801103	Aprobado	**64**	0.261756	Aprobado
15	0.716951	Aprobado	**65**	0.406465	Aprobado
16	0.218722	Aprobado	**66**	0.675214	Aprobado
17	0.102633	Aprobado	**67**	0.088882	Aprobado
18	0.572946	Aprobado	**68**	0.115896	Aprobado
19	0.161964	Aprobado	**69**	0.569225	Aprobado
20	0.001229	**Reprobado**	**70**	0.115896	Aprobado
21	0.054031	Aprobado	**71**	0.110196	Aprobado
22	0.46894	Aprobado	**72**	0.690441	Aprobado
23	0.216729	Aprobado	**73**	0.554419	Aprobado
24	0.139988	Aprobado	**74**	0.854668	Aprobado
25	0.828562	Aprobado	**75**	0.907126	Aprobado
26	0.113585	Aprobado	**76**	0.088882	Aprobado
27	0.08094	Aprobado	**77**	0.1683	Aprobado

28	0.973758	Aprobado	**78**	0.125557	Aprobado
29	0.618091	Aprobado	**79**	0.057644	Aprobado
30	0.366819	Aprobado	**80**	0.390906	Aprobado
31	0.011554	Aprobado	**81**	0.173188	Aprobado
32	0.332594	Aprobado	**82**	0.375734	Aprobado
33	0.335358	Aprobado	**83**	0.70183	Aprobado
34	0.873203	Aprobado	**84**	0.183323	Aprobado
35	0.03722	Aprobado	**85**	0.23733	Aprobado
36	0.705618	Aprobado	**86**	0.119439	Aprobado
37	0.70183	Aprobado	**87**	0.295599	Aprobado
38	0.192143	Aprobado	**88**	0.311066	Aprobado
39	0.013944	Aprobado	**89**	0.142761	Aprobado
40	0.308448	Aprobado	**90**	0.8046	Aprobado
41	0.561806	Aprobado	**91**	0.106896	Aprobado
42	0.047945	Aprobado	**92**	0.023457	Aprobado
43	0.158874	Aprobado	**93**	0.252656	Aprobado
44	0.422402	Aprobado	**94**	0.38784	Aprobado
45	0.064851	Aprobado	**95**	0.409622	Aprobado
46	0.569225	Aprobado	**96**	0.321701	Aprobado
47	0.050628	Aprobado	**97**	0.178195	Aprobado
48	0.19945	Aprobado	**98**	0.606733	Aprobado
49	0.338137	Aprobado	**99**	0.587901	Aprobado
50	0.110196	Aprobado	**100**	0.157347	Aprobado

Tabla 4.2. Valores *P* obtenidos por 100 pruebas de frecuencia de bloque en la implementación del TRNG.

Valores *P* de la prueba de racha.					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación

1	0.854238	Aprobado	**51**	0.174687	Aprobado
2	0.657969	Aprobado	**52**	0.411248	Aprobado
3	0.62591	Aprobado	**53**	0.840015	Aprobado
4	0.376982	Aprobado	**54**	0.25487	Aprobado
5	0.16644	Aprobado	**55**	0.08768	Aprobado
6	0.282271	Aprobado	**56**	0.589957	Aprobado
7	0.06404	Aprobado	**57**	0.849416	Aprobado
8	0.02365	Aprobado	**58**	0.44294	Aprobado
9	0.591006	Aprobado	**59**	0.528616	Aprobado
10	0.858937	Aprobado	**60**	0.100949	Aprobado
11	0.523405	Aprobado	**61**	0.71085	Aprobado
12	0.76318	Aprobado	**62**	0.628152	Aprobado
13	0.523405	Aprobado	**63**	0.442627	Aprobado
14	0.839552	Aprobado	**64**	0.669543	Aprobado
15	0.368373	Aprobado	**65**	0.336799	Aprobado
16	0.92761	Aprobado	**66**	0.207788	Aprobado
17	0.077307	Aprobado	**67**	0.559888	Aprobado
18	0.025082	Aprobado	**68**	0.846512	Aprobado
19	0.849117	Aprobado	**69**	0.132881	Aprobado
20	0.078499	Aprobado	**70**	0.752389	Aprobado
21	0.964526	Aprobado	**71**	0.835664	Aprobado
22	0.142201	Aprobado	**72**	0.344345	Aprobado
23	0.916229	Aprobado	**73**	0.543826	Aprobado
24	0.951791	Aprobado	**74**	0.93683	Aprobado
25	0.167801	Aprobado	**75**	0.254945	Aprobado
26	0.316884	Aprobado	**76**	0.043956	Aprobado
27	0.488549	Aprobado	**77**	0.505719	Aprobado
28	0.748533	Aprobado	**78**	0.179534	Aprobado
29	0.784072	Aprobado	**79**	0.214428	Aprobado
30	0.628152	Aprobado	**80**	0.702285	Aprobado

31	0.545485	Aprobado	**81**	0.833322	Aprobado
32	0.516937	Aprobado	**82**	0.420135	Aprobado
33	0.386756	Aprobado	**83**	0.486535	Aprobado
34	0.518692	Aprobado	**84**	0.443519	Aprobado
35	0.734302	Aprobado	**85**	0.129072	Aprobado
36	0.845927	Aprobado	**86**	0.40789	Aprobado
37	0.114287	Aprobado	**87**	0.420942	Aprobado
38	0.028558	Aprobado	**88**	0.698262	Aprobado
39	0.162676	Aprobado	**89**	0.530009	Aprobado
40	0.841629	Aprobado	**90**	0.121683	Aprobado
41	0.450548	Aprobado	**91**	0.504235	Aprobado
42	0.22876	Aprobado	**92**	0.495465	Aprobado
43	0.720587	Aprobado	**93**	0.313038	Aprobado
44	0.737904	Aprobado	**94**	0.077738	Aprobado
45	0.664212	Aprobado	**95**	0.244373	Aprobado
46	0.447958	Aprobado	**96**	0.154682	Aprobado
47	0.587028	Aprobado	**97**	0.591229	Aprobado
48	0.692865	Aprobado	**98**	0.119539	Aprobado
49	0.007538	Aprobado	**99**	0.265025	Aprobado
50	0.411599	Aprobado	**100**	0.398591	Aprobado

Tabla 4.3. Valores *P* obtenidos por 100 pruebas de racha en la implementación del TRNG.

Valores *P* de la prueba de sumas acumuladas.					
No. de prueba	Valor *P*	Evaluación	**No. de prueba**	Valor *P*	Evaluación
1	0.027282	Aprobado	**51**	0.389382	Aprobado
2	0.011876	Aprobado	**52**	0.045592	Aprobado
3	0.328147	Aprobado	**53**	0.011876	Aprobado
4	0.328147	Aprobado	**54**	0.017346	Aprobado

5	0.124154	Aprobado	**55**	0.291508	Aprobado
6	0.053713	Aprobado	**56**	0.21359	Aprobado
7	0.850473	Aprobado	**57**	0.099849	Aprobado
8	0.98972	Aprobado	**58**	0.115559	Aprobado
9	0.092691	Aprobado	**59**	0.411511	Aprobado
10	0.092691	Aprobado	**60**	0.309419	Aprobado
11	0.079665	Aprobado	**61**	0.175409	Aprobado
12	0.309419	Aprobado	**62**	0.309419	Aprobado
13	0.011876	Aprobado	**63**	0.618347	Aprobado
14	0.019025	Aprobado	**64**	0.175409	Aprobado
15	0.029787	Aprobado	**65**	0.200192	Aprobado
16	0.045592	Aprobado	**66**	0.175409	Aprobado
17	0.142934	Aprobado	**67**	0.589898	Aprobado
18	0.434505	Aprobado	**68**	0.922381	Aprobado
19	0.092691	Aprobado	**69**	0.16398	Aprobado
20	0.058223	Aprobado	**70**	0.079665	Aprobado
21	0.079665	Aprobado	**71**	0.794732	Aprobado
22	0.041947	Aprobado	**72**	0.618347	Aprobado
23	0.017346	Aprobado	**73**	0.706355	Aprobado
24	0.035412	Aprobado	**74**	0.706355	Aprobado
25	0.08597	Aprobado	**75**	0.027282	Aprobado
26	0.073758	Aprobado	**76**	0.045592	Aprobado
27	0.458362	Aprobado	**77**	0.242509	Aprobado
28	0.900481	Aprobado	**78**	0.099849	Aprobado
29	0.291508	Aprobado	**79**	0.175409	Aprobado
30	0.328147	Aprobado	**80**	0.175409	Aprobado
31	0.115559	Aprobado	**81**	0.850473	Aprobado
32	0.010778	Aprobado	**82**	0.508616	Aprobado
33	0.124154	Aprobado	**83**	0.010778	Aprobado
34	0.291508	Aprobado	**84**	0.068227	Aprobado

35	0.534965	Aprobado	**85**	0.175409	Aprobado
36	0.483072	Aprobado	**86**	0.124154	Aprobado
37	0.368117	Aprobado	**87**	0.032493	Aprobado
38	0.291508	Aprobado	**88**	0.035412	Aprobado
39	0.079665	Aprobado	**89**	0.258072	Aprobado
40	0.227688	Aprobado	**90**	0.142934	Aprobado
41	0.008853	Aprobado	**91**	0.765607	Aprobado
42	0.003883	Aprobado	**92**	0.823133	Aprobado
43	0.258072	Aprobado	**93**	0.458362	Aprobado
44	0.153163	Aprobado	**94**	0.073758	Aprobado
45	0.618347	Aprobado	**95**	0.124154	Aprobado
46	0.133272	Aprobado	**96**	0.483072	Aprobado
47	0.073758	Aprobado	**97**	0.142934	Aprobado
48	0.053713	Aprobado	**98**	0.227688	Aprobado
49	0.823133	Aprobado	**99**	0.328147	Aprobado
50	0.389382	Aprobado	**100**	0.227688	Aprobado

Tabla 4.4. Valores *P* obtenidos por 100 pruebas de sumas acumuladas en la implementación del TRNG.

En resumen, de las 400 pruebas estadísticas realizadas 97 pruebas de Monobit, 99 de frecuencia en bloques, 100 pruebas de racha y 100 pruebas de sumas acumuladas fueron aprobadas de forma satisfactoria. Es decir, un 0.75% de las pruebas fallaron, lo cual es muy cercano a lo esperado de una secuencia de bits aleatoria.

En conclusión, con respecto a las pruebas realizadas, se puede decir que *la secuencia de bits es aleatoria* en base a las pruebas NIST-FIPS aplicadas sobre dicha secuencia.

Capítulo 5

Conclusiones y trabajo futuro

5. Conclusiones

El estudio de los sistemas caóticos ha sido más común en los últimos años debido a su implicación en tantas y diversas áreas de estudio, los desafíos que su análisis presenta han despertado un interés importante dentro de la comunidad científica.

Esta tesis ha explorado una de las aplicaciones de los sistemas caóticos, la generación de entropía para un RNG, y a través de su estudio y la experimentación con estos, que su naturaleza impredecible es una fuente de entropía apropiada para la generación de numeros aleatorios, esto con una metodología correcta, demostrado por medio de la aprobación de las distintas pruebas estadísticas NIST-FIPS asegurando la fiabilidad y seguridad del TRNG propuesto.

La selección de los parámetros correctos para optimización de la calidad de la secuencia de numeros aleatorios requiere de muchos ajustes y rediseños, para lo cual, los FPAA ofrecieron una gran flexibilidad y la reconfigurabilidad necesaria, permitiéndonos la implementación de los complejos circuitos hipercaóticos, y que, de otra forma, hubieran presentado un desafío de bastante más complejidad. Aún más, los FPAA facilitaron la integración con otros componentes digitales que también componen el modelo del TRNG.

Los resultados obtenidos demuestran que la aplicación de sistemas caóticos en FPAA's permiten el desarrollo de robustos y seguros TRNG's, del cual se obtuvo una secuencia de bits con propiedades criptográficas deseables. En particular, el TRNG desarrollado es capaz de producir secuencias de bloques de 2 bits, los cuales pueden ser extendidos hasta bloques de $2n$ bits.

En general, la metodología para el TRNG propuesta permite la implementación de distintos parámetros, ajustes o incluso rediseños que proporcionen distintos resultados y permita superar los desafíos que el desarrollo de TRNG.

5.1 Trabajo futuro

- Mejorar la velocidad de generación de bits.
- Analizar sistemas hipercaóticos con mayor número de variables.
- Estudiar otras propuestas de diseño para el TRNG.

Referencias

1. Mendoza, M., Contreras-Cristán, A., & Gutiérrez-Peña, E. (2021). Bayesian Analysis of Finite Populations under Simple Random Sampling. Entropy, 23(3), 318. MDPI AG. Recuperado de http://dx.doi.org/10.3390/e23030318
2. Zhang, S., Wang, J., & Li, S. (2021). A new class of pseudo-random number generators based on chaotic maps with multiple attractors. Entropia, 23(3), 306. https://doi.org/10.3390/e23030306
3. Brugger, Christian & Weithoffer, Stefan & De Schryver, Christian & Wasenmüller, Uwe & Wehn, Norbert. (2014). On parallel random number generation for accelerating simulations of communication systems. Advances in Radio Science. 12. 75-81. 10.5194/ars-12-75-2014.
4. National Institute of Standards and Technology (NIST). (2010). NIST Special Publication 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (p.1-2). https://doi.org/10.6028/NIST.SP.800-22
5. Lee, K., & Lee, M. (2019). True Random Number Generator (TRNG) Utilizing FM Radio Signals for Mobile and Embedded Devices in Multi-Access Edge Computing. Sensors, 19(19), 4130. MDPI AG. Retrieved from http://dx.doi.org/10.3390/s19194130.
6. Kaçar, S. (2016). Analog circuit and microcontroller based RNG application of a new easy realizable 4D chaotic system. Optik, 127(20), 9551–9561. https://doi.org/10.1016/j.ijleo.2016.07.044
7. Ozkaynak, F. (2020). A Novel Random Number Generator Based on Fractional Order Chaotic Chua System. Elektronika Ir Elektrotechnika, 26(1), 52-57. https://doi.org/10.5755/j01.eie.26.1.25310
8. Devaney, R. L. L. (2020). A First Course In Chaotic Dynamical Systems: Theory And Experiment. CRC Press.
9. Alligood, K., Sauer, T., & Yorke, J. (2012). Chaos: An Introduction to Dynamical Systems. Springer.
10. Khalil, H. K. (1992). Nonlinear Systems. MacMillan Publishing Company.
11. Shanker, B., & Emery, R. J. (2021). Introduction to Linear Systems for Scientists and Engineers. Springer.
12. Kurzhanski, A. B., & Varaiya, P. (2020). Linear Systems and Optimal Control. Springer.
13. Steven H. Strogatz, P. (2018). Nonlinear Dinamics and chaos, with applications to physics, Biology, Chemistry, and Engineering. Springer.
14. Casati, G., Prosen, T. (2022). Quantum Chaos. In: Chakraborty, B. (eds) Statistical and Nonlinear Physics. Encyclopedia of Complexity and Systems Science Series. Springer, New York, NY. https://doi.org/10.1007/978-1-0716-1454-9_427
15. Wang, B., Liu, J., Alassafi, M. O., Alsaadi, F. E., Jahanshahi, H., & Bekiros, S. (2021). Intelligent parameter identification and prediction of variable time

fractional derivative and application in a symmetric chaotic financial system. Chaos Solitons & Fractals, 154, 111590. https://doi.org/10.1016/j.chaos.2021.111590
16. Gleick, J. (2008). Chaos: Making a New Science. Penguin Books.
17. Denton, T. A., Diamond, G. A., Helfant, R. H., Khan, S. S., & Karagueuzian, H. S. (1990). Fascinating rhythm: A primer on chaos theory and its application to cardiology. American Heart Journal, 120(6), 1419-1440. https://doi.org/10.1016/0002-8703(90)90258-y
18. Edward Norton Lorenz, "Deterministic non- periodic flow", Journal of Atmospheric Science, no.20, pp.130-141, 1963
19. Fernandez-Canque, H. L. (2019). Analog Electronics Applications: Fundamentals of Design and Analysis. CRC Press.
20. Kapoulea, S., Psychalinos, C., & Elwakil, A. S. (2022). Versatile Field-Programmable Analog Array Realizations of Power-Law Filters. Electronics. https://doi.org/10.3390/electronics11050692
21. Tlelo-Cuautle, E., Pano-Azucena, A. D., Guillén-Fernández, O., & Silva-Juárez, A. (2020). Analog/Digital Implementation of Fractional Order Chaotic Circuits and Applications. En Springer eBooks. https://doi.org/10.1007/978-3-030-31250-3
22. L'Ecuyer, P. (1994). Uniform random number generation. Annals of Operations Research, 53(1), 77-120. https://doi.org/10.1007/bf02136827
23. Knuth, D. E. (1998). The Art of Computer Programming. Volume 2: Seminumerical Algorithms. American Mathematical Monthly. https://doi.org/10.2307/2317055
24. C. S. Petrie and J. A. Connelly, "A noise-based IC random number generator for applications in cryptography," in IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 47, no. 5, pp. 615-621, May 2000, doi: 10.1109/81.847868.
25. Li, Y., Fei, Y., Wang, W. *et al.* Quantum random number generator using a cloud superconducting quantum computer based on source-independent protocol. *Sci Rep* 11, 23873 (2021). https://doi.org/10.1038/s41598-021-03286-9
26. Letellier, C., & Rössler, O. E. (2007). Hyperchaos. Scholarpedia, 2(8), 1936. https://doi.org/10.4249/scholarpedia.1936
27. Kilic, R. (2010). A Practical Guide for Studying Chua's Circuits. World Scientific.
28. Rössler, O. E. (1979b). An equation for hyperchaos. Physics Letters, 71(2-3), 155-157. https://doi.org/10.1016/0375-9601(79)90150-6
29. Zhang, G., Zhang, F., Liao, X., Lin, D., & Zhou, P. (2017). On the dynamics of new 4D Lorenz-type chaos systems. Advances in Difference Equations, 2017(1). https://doi.org/10.1186/s13662-017-1280-5
30. Yu, F., Li, L., Tang, Q., Song, Y. S., & Xu, Q. (2019). A Survey on True Random Number Generators Based on Chaos. Discrete Dynamics in Nature and Society, 2019, 1-10. https://doi.org/10.1155/2019/2545123

APENDICE A

A Códigos en Matlab

A.1 Simulación del sistema caótico de Chua

```
%% Simulación de sistema caótico de Chua, solución por medio del integrador
ODE45
%
%
%% parámetros para la simulación ----------------------------------------

% Tiempo inicial y final de simulación
ti = 0;
tf = 50;
% Valores iniciales
x0 = 1;
y0 = 0;
z0 = 0;

%% Solución por integrador ODE45 ----------------------------------------------

[t,x] = ode45(@ODE,[ti tf],[x0;y0;z0]);


%% Graficas con respecto al tiempo y las demás variables --------------------
% Graficas con respecto a tiempo
figure
    plot(t, x(:,1))
    xlabel('Time t')
    ylabel('x')

figure
    plot(t, x(:,2))
    xlabel('Time t')
    ylabel('y')

figure
    plot(t, x(:,3))
    xlabel('Time t')
    ylabel('z')

% Graficas de fase
% Graficacion de líneas de puntos de equilibrio
L = -1.5:3/(length(t)-1):1.5;

    epx1 = zeros(length(t),1);
    epx2 = zeros(length(t),1);
    epy1 = zeros(length(t),1);
    epz1 = zeros(length(t),1);
```

```
    epz2 = zeros(length(t),1);
    for i = 1:length(t)
       epx1(i) = -0.8294;
       epx2(i) = 0.8294;
       epz1(i) = -0.4879;
       epz2(i) = 0.4879;
    end
figure
    plot(x(:,1), x(:,2),epx1, L, 'r--', epx2, L, 'r--', L, epy1, 'r--')
    xlabel('x')
    ylabel('y')

    figure
    plot(x(:,1), x(:,3),epx1, L, 'r--', epx2, L, 'r--', L, epz1, 'r--', L,
epz2, 'r--')
    xlabel('y')
    ylabel('z')

    figure
    plot(x(:,2), x(:,3),epy1, L, 'r--', L, epz1, 'r--', L, epz2, 'r--')
    xlabel('y')
    ylabel('z')

% Graficacion 3D del sistema caótico de Chua

figure
plot3(x(:,1), x(:,2), x(:,3))
grid on
xlabel("x")
ylabel("y")
zlabel("z")
```

A.2 Simulación del sistema hipercaótico de Lorenz

```
%% Simulacion de sistema hipercaotico de Lorenz con retroalimentacion en la
% variable de estado x, solucion por medio del integrador ODE45.
%
%
%% Parametros para la simulacion  ----------------------------------------
% Tiempo inicial y final de simulacion
ti = 0;
tf = 70;

% Valores iniciales
x0 = 0.1;
y0 = 0;
z0 = 0;
w0 = 0;

% Solucion de la ODE por medio del integrador
[t,x] = ode45(@ODE,[0 70],[x0;y0;z0;w0]);

%% Graficacion ----------------------------------------------------------
% Graficas con respecto al tiempo
figure
grid on
    subplot(2,2,1)
```

```
    plot(t, x(:,1))
    xlabel("(a)Tiempo")
    ylabel("x")
    subplot(2,2,2)
    plot(t, x(:,2))
    xlabel("(b)Tiempo")
    ylabel("y")
    subplot(2,2,3)
    plot(t, x(:,3))
    xlabel("(c)Tiempo")
    ylabel("z")
    subplot(2,2,4)
    plot(t, x(:,4))
    xlabel("(d)Tiempo")
    ylabel("w")

% Graficas de fase

figure
grid on
    subplot(3,2,1)
    plot(x(:,1), x(:,2))
    xlabel("(a)        x")
    ylabel("y")

    subplot(3,2,2)
    plot(x(:,1), x(:,3))
    xlabel("(b)        x")
    ylabel("z")

    subplot(3,2,3)
    plot(x(:,1), x(:,4))
    xlabel("(c)        x")
    ylabel("w")

    subplot(3,2,4)
    plot(x(:,2), x(:,3))
    xlabel("(d)        y")
    ylabel("z")

    subplot(3,2,5)
    plot(x(:,2), x(:,4))
    xlabel("(e)        y")
    ylabel("w")

    subplot(3,2,6)
    plot(x(:,3), x(:,4))
    xlabel("(f)        z")
    ylabel("w")

% Graficas 3D

figure
grid on
    subplot(2,2,1)
    plot3(x(:,1),x(:,2),x(:,3))
    xlabel("x")
    ylabel("y")
    zlabel("z")
```

```
grid on
    subplot(2,2,2)
    plot3(x(:,1),x(:,2),x(:,4))
    xlabel("x")
    ylabel("y")
    zlabel("w")
grid on
    subplot(2,2,3)
    plot3(x(:,2),x(:,3),x(:,4))
    xlabel("y")
    ylabel("z")
    zlabel("w")
grid on
    subplot(2,2,4)
    plot3(x(:,1),x(:,3),x(:,4))
    xlabel("x")
    ylabel("z")
    zlabel("w")
grid on

%% Funciones
% Funcion para la ODE del sistema hipercaotico de Lorenz
function dydt = ODE(t, xn)
a = 5;
b = 20;
c = 1;
d = 0.1;
k = 0.1;
e = 20.6;
h = 1;

x = xn(1);
y = xn(2);
z = xn(3);
w = xn(4);

dydt = [a*(y-x)-e*w;
        x*z-h*y;
        b-x*y-c*z;
        k*y-d*w];
end
```

A.3 Almacenamiento de datos

```
%% Programa almacenamiento de bits generados por Simulink
%
%
%% Variables
clear numeros
data = out.salida.data;
time = out.salida.time;
clk = data(:,3);
reg = [data(:,1),data(:,2)];
numeros = zeros(1,2);
tiempo = zeros(1,1);
pos = 1;
```

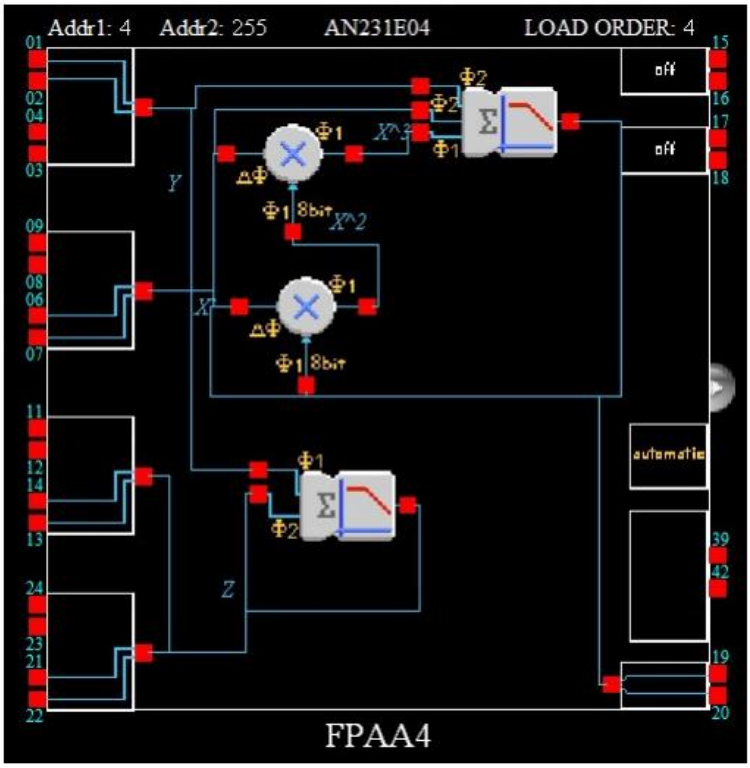
```
%% Flancos positivos
for i = 1:length(data)
    if i == 1
        continue
    end
    if clk(i)==1 && clk(i-1)==0
        numeros(pos, 1) = reg(i,1);
        numeros(pos, 2) = reg(i,2);
        tiempo(pos) = time(i);
        pos = pos + 1;
    else
        continue
    end
end

%% Almacenamiento en archivo "info.txt"
file = fopen('C:\cygwin64\sts-2.1.2\data\info.txt', 'w');

for i= 1:50000
    fprintf(file,'%s',string(numeros(i,1)), string(numeros(i,2)));
end
fclose(file);
```

APENDICE B

B Configuración de los FPAA y la tarjeta Anadigm SingleApex Development Board

B.1 Configuración FPAA para la simulación del sistema caótico de Chua, su etapa de operación y acondicionamiento.

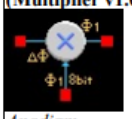
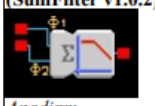
Chip: FPAA4



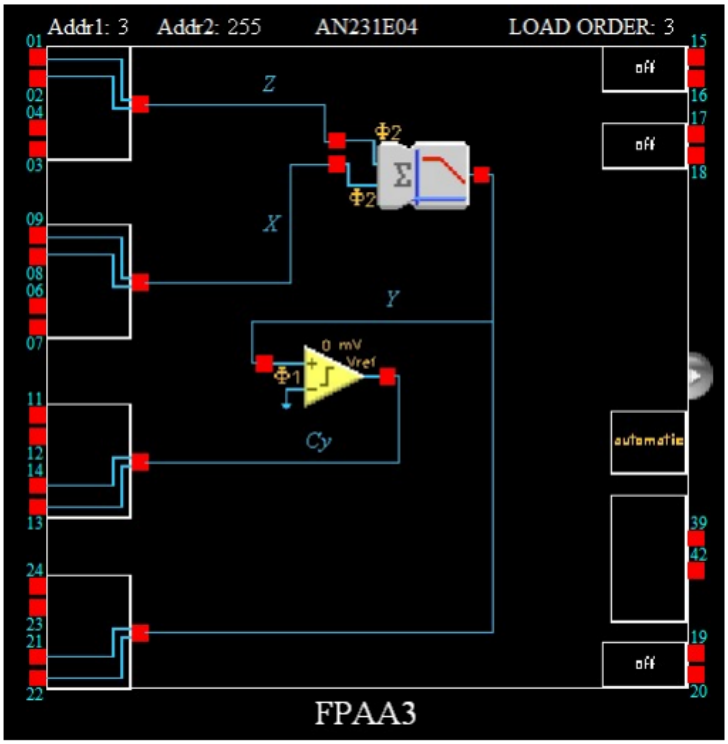
Clocks: FPAA4

Master Clock - ACLK (fc)	*16 MHz*		
System Clock 1 (sys1 = fc / 1)	*16 MHz*	**System Clock 2 (sys2 = fc / 1)**	*16 MHz*
Clock 0 (sys1 / 64)	*250 kHz*	**Clock 1 (sys1 / 4)**	*4 MHz*
Clock 2 (sys1 / 8)	*2 MHz*	**Clock 3 (sys1 / 64)**	*250 kHz*
Clock 4 (sys1 / 1)	*16 MHz*	**Clock 5 (sys1 / 1)**	*16 MHz*

Configurable Analog Modules: FPAA4

Name	Options	Parameters	Clocks
SumFilter1 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Inverting*	**Corner Frequency [kHz]** *0.999* **Gain 1 (UpperInput)** *2.33* **Gain 2 (MiddleInput)** *2.67* **Gain 3 (LowerInput)** *4.00*	**ClockA** *250 kHz (Chip Clock 0)*
Multiplier2 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 0)* **ClockB** *4 MHz (Chip Clock 1)*
Multiplier1 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 0)* **ClockB** *4 MHz (Chip Clock 1)*
SumFilter2 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Off*	**Corner Frequency [kHz]** *0.999* **Gain 1 (UpperInput)** *2.17* **Gain 2 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*

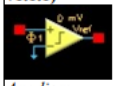
Chip: FPAA3



Clocks: FPAA3

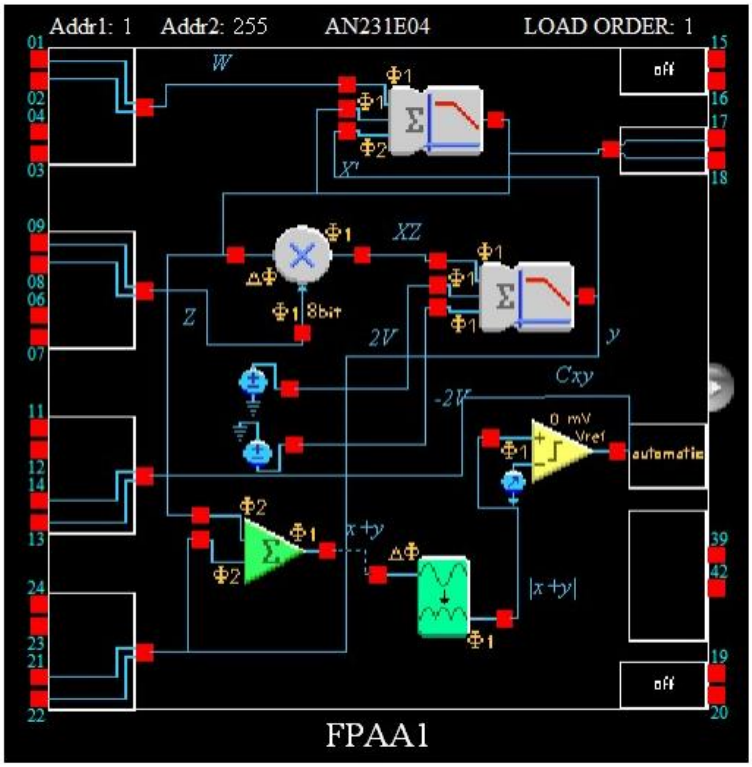
Master Clock - ACLK (fc)	*16 MHz*		
System Clock 1 (sys1 = fc / 1)	*16 MHz*	**System Clock 2 (sys2 = fc / 1)**	*16 MHz*
Clock 0 (sys1 / 64)	*250 kHz*	**Clock 1 (sys1 / 4)**	*4 MHz*
Clock 2 (sys1 / 8)	*2 MHz*	**Clock 3 (sys1 / 64)**	*250 kHz*
Clock 4 (sys1 / 1)	*16 MHz*	**Clock 5 (sys1 / 1)**	*16 MHz*

Configurable Analog Modules: FPAA3

Name	Options	Parameters	Clocks
SumFilter3 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Off*	**Corner Frequency [kHz]** *0.999* **Gain 1 (UpperInput)** *7.00* **Gain 2 (LowerInput)** *4.17*	**ClockA** *250 kHz (Chip Clock 0)*
Comparator1 (Comparator v1.1.1)  *Anadigm (Approved)*	**Compare To** *Signal Ground* **Input Sampling** *Phase 1* **Output Polarity** *Non-inverted* **Hysteresis** *0 mV* **Output Synch** *None*		**ClockA** *250 kHz (Chip Clock 0)*

B.2 Configuración FPAA para la simulación del sistema hipercaótico de Lorenz, su etapa de operación y acondicionamiento.

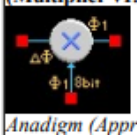
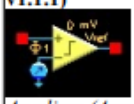
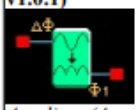
Chip: FPAA1



Clocks: FPAA1

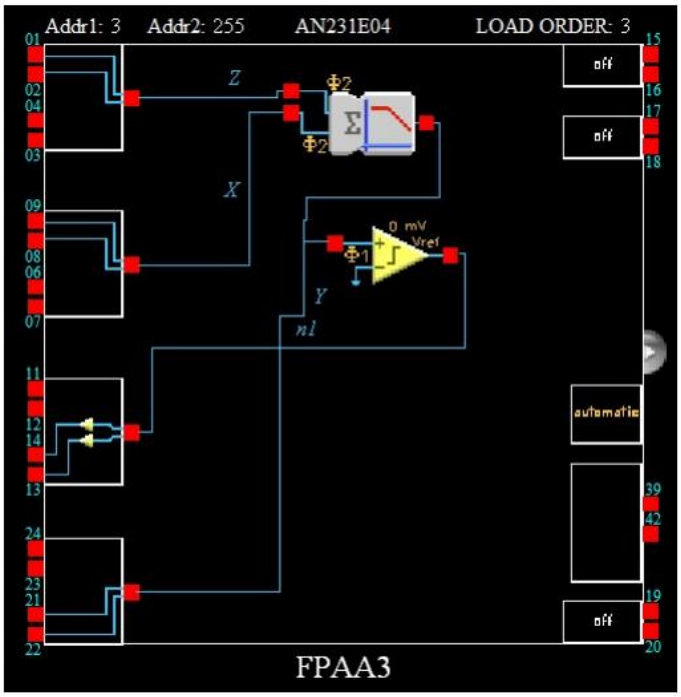
Master Clock - ACLK (fc)	*16 MHz*		
System Clock 1 (sys1 = fc / 1)	*16 MHz*	**System Clock 2 (sys2 = fc / 1)**	*16 MHz*
Clock 0 (sys1 / 64)	*250 kHz*	**Clock 1 (sys1 / 4)**	*4 MHz*
Clock 2 (sys1 / 8)	*2 MHz*	**Clock 3 (sys1 / 1)**	*16 MHz*
Clock 4 (sys1 / 16)	*1 MHz*	**Clock 5 (sys1 / 1)**	*16 MHz*

Configurable Analog Modules: FPAA1

Name	Options	Parameters	Clocks
SumFilter1 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Inverting* **Input 2** *Inverting* **Input 3** *Non-inverting*	**Corner Frequency [kHz]** *1.00* **Gain 1 (UpperInput)** *1.00* **Gain 2 (MiddleInput)** *4.00* **Gain 3 (LowerInput)** *5.00*	**ClockA** *250 kHz (Chip Clock 0)*
SumFilter2 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 2* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Non-inverting*	**Corner Frequency [kHz]** *1.00* **Gain 1 (UpperInput)** *20.0* **Gain 2 (MiddleInput)** *1.00* **Gain 3 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*
Multiplier1 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 0)* **ClockB** *4 MHz (Chip Clock 1)*
Voltage1 (Voltage v1.0.1)  *Anadigm (Approved)*	**Polarity** *Positive (+2V)*		
Voltage2 (Voltage v1.0.1)  *Anadigm (Approved)*	**Polarity** *Negative (-2V)*		
Comparator1 (Comparator v1.1.1)  *Anadigm (Approved)*	**Compare To** *Variable Reference* **Input Sampling** *Phase 1* **Output Polarity** *Non-inverted* **Output Synch** *None*	**Reference Voltage** *0.210*	**ClockA** *250 kHz (Chip Clock 0)*
RectifierHalf1 (RectifierHalf v1.0.1)  *Anadigm (Approved)*	**Rectification** *Full Wave* **Polarity** *Non-inverting* **Output Phase** *Phase 1*	**Gain** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*
SumDiff1 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *1.00* **Gain 2 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*

B.3 Configuración FPAA para la implementación del sistema caótico de Chua, su etapa de operación y acondicionamiento.

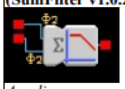
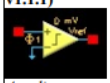
Chip: FPAA3



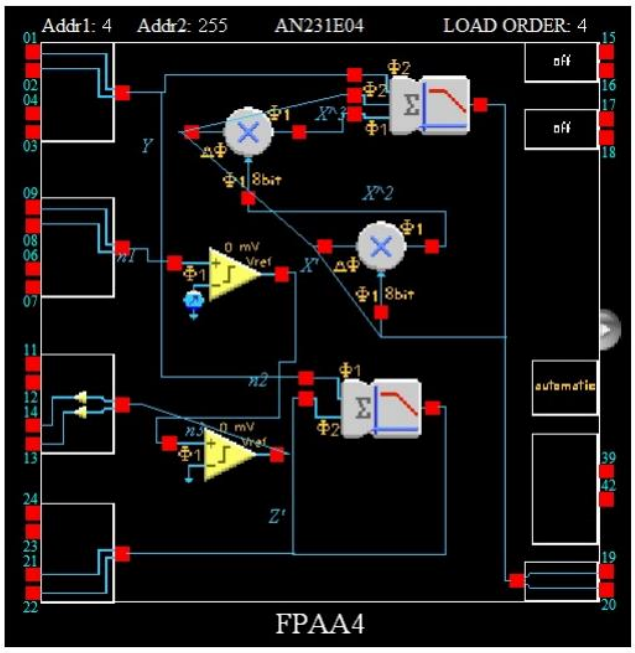
Clocks: FPAA3

Master Clock - ACLK (fc)	*16 MHz*		
System Clock 1 (sys1 = fc / 1)	*16 MHz*	**System Clock 2 (sys2 = fc / 1)**	*16 MHz*
Clock 0 (sys1 / 64)	*250 kHz*	**Clock 1 (sys1 / 4)**	*4 MHz*
Clock 2 (sys1 / 8)	*2 MHz*	**Clock 3 (sys1 / 64)**	*250 kHz*
Clock 4 (sys1 / 1)	*16 MHz*	**Clock 5 (sys1 / 1)**	*16 MHz*

Configurable Analog Modules: FPAA3

Name	Options	Parameters	Clocks
SumFilter3 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Off*	**Corner Frequency [kHz]** *0.898* **Gain 1 (UpperInput)** *7.00* **Gain 2 (LowerInput)** *4.20*	**ClockA** *250 kHz (Chip Clock 0)*
Comparator1 (Comparator v1.1.1)  *Anadigm (Approved)*	**Compare To** *Signal Ground* **Input Sampling** *Phase 1* **Output Polarity** *Non-inverted* **Hysteresis** *0 mV* **Output Synch** *None*		**ClockA** *250 kHz (Chip Clock 0)*

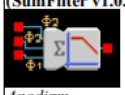
Chip: FPAA4



Clocks: FPAA4

Master Clock - ACLK (fc)	*16 MHz*		
System Clock 1 (sys1 = fc / 1)	*16 MHz*	**System Clock 2 (sys2 = fc / 1)**	*16 MHz*
Clock 0 (sys1 / 64)	*250 kHz*	**Clock 1 (sys1 / 4)**	*4 MHz*
Clock 2 (sys1 / 8)	*2 MHz*	**Clock 3 (sys1 / 64)**	*250 kHz*
Clock 4 (sys1 / 1)	*16 MHz*	**Clock 5 (sys1 / 1)**	*16 MHz*

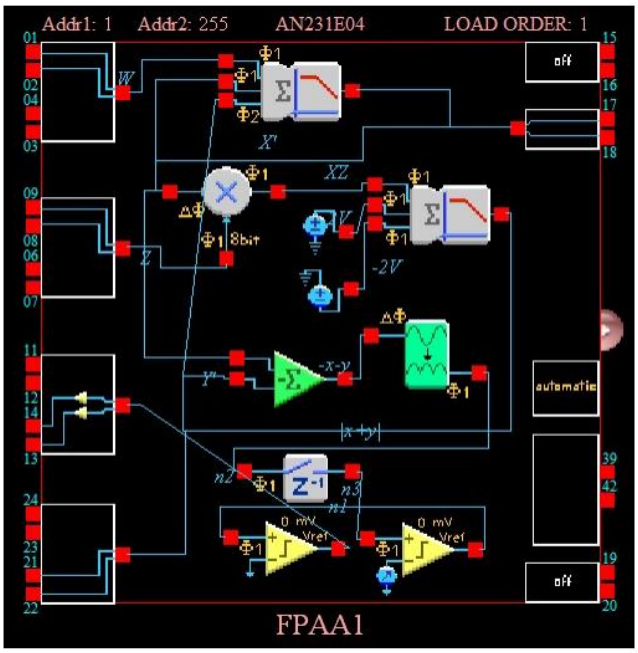
Configurable Analog Modules: FPAA4

Name	Options	Parameters	Clocks
SumFilter1 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Inverting*	**Corner Frequency [kHz]** *0.901* **Gain 1 (UpperInput)** *2.33* **Gain 2 (MiddleInput)** *2.67* **Gain 3 (LowerInput)** *4.00*	**ClockA** *250 kHz (Chip Clock 0)*
Multiplier2 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 0)* **ClockB** *4 MHz (Chip Clock 1)*

Multiplier1 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 0)* **ClockB** *4 MHz (Chip Clock 1)*
SumFilter2 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Off*	**Corner Frequency [kHz]** *0.925* **Gain 1 (UpperInput)** *2.17* **Gain 2 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*
Comparator1 (Comparator v1.1.1)  *Anadigm (Approved)*	**Compare To** *Variable Reference* **Input Sampling** *Phase 1* **Output Polarity** *Non-inverted* **Output Synch** *None*	**Reference Voltage** *0.404*	**ClockA** *250 kHz (Chip Clock 0)*
Comparator2 (Comparator v1.1.1)  *Anadigm (Approved)*	**Compare To** *Signal Ground* **Input Sampling** *Phase 1* **Output Polarity** *Non-inverted* **Hysteresis** *0 mV* **Output Synch** *None*		**ClockA** *250 kHz (Chip Clock 0)*

B.4 Configuración FPAA para la implementación del sistema hipercaótico de Lorenz, su etapa de operación y acondicionamiento.

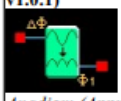
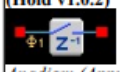
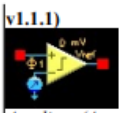
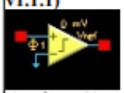
Chip: FPAA1



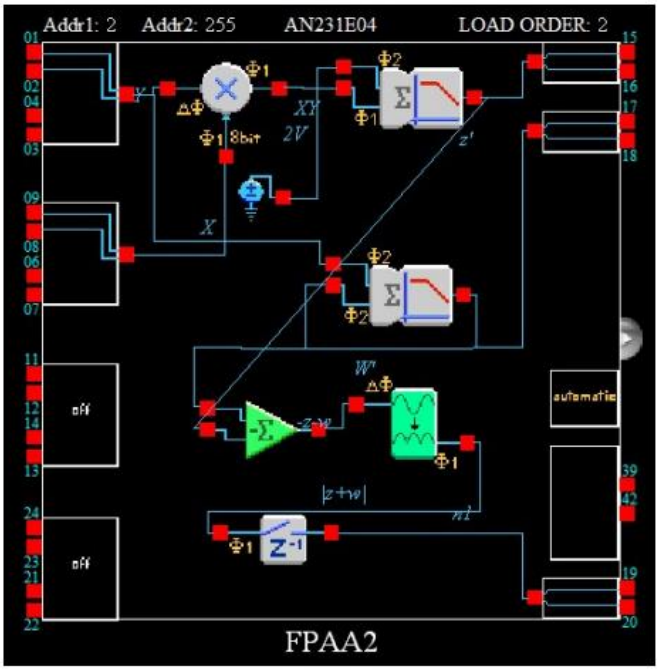
Clocks: FPAA1

Master Clock - ACLK (fc)	*16 MHz*		
System Clock 1 (sys1 = fc / 1)	*16 MHz*	**System Clock 2 (sys2 = fc / 1)**	*16 MHz*
Clock 0 (sys1 / 64)	*250 kHz*	**Clock 1 (sys1 / 4)**	*4 MHz*
Clock 2 (sys1 / 8)	*2 MHz*	**Clock 3 (sys1 / 1)**	*16 MHz*
Clock 4 (sys1 / 16)	*1 MHz*	**Clock 5 (sys1 / 1)**	*16 MHz*

Configurable Analog Modules: FPAA1

Name	Options	Parameters	Clocks
SumFilter1 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Inverting* **Input 2** *Inverting* **Input 3** *Non-inverting*	**Corner Frequency [kHz]** *0.901* **Gain 1 (UpperInput)** *1.00* **Gain 2 (MiddleInput)** *4.00* **Gain 3 (LowerInput)** *5.00*	**ClockA** *250 kHz (Chip Clock 0)*
SumFilter2 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 2* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Non-inverting*	**Corner Frequency [kHz]** *0.901* **Gain 1 (UpperInput)** *20.0* **Gain 2 (MiddleInput)** *1.00* **Gain 3 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*
Multiplier1 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 0)* **ClockB** *4 MHz (Chip Clock 1)*
Voltage1 (Voltage v1.0.1)  *Anadigm (Approved)*	**Polarity** *Positive (+2V)*		
Voltage2 (Voltage v1.0.1)  *Anadigm (Approved)*	**Polarity** *Negative (-2V)*		
SumInv1 (SumInv v1.0.1)  *Anadigm (Approved)*	**Input 3** *Off*	**Gain 1 (UpperInput)** *1.00* **Gain 2 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*
RectifierHalf1 (RectifierHalf v1.0.1)  *Anadigm (Approved)*	**Rectification** *Full Wave* **Polarity** *Non-inverting* **Output Phase** *Phase 1*	**Gain** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*
Hold3 (Hold v1.0.2)  *Anadigm (Approved)*	**Input Sampling Phase** *Phase 1*		**ClockA** *250 kHz (Chip Clock 0)*
Comparator1 (Comparator v1.1.1)  *Anadigm (Approved)*	**Compare To** *Variable Reference* **Input Sampling** *Phase 1* **Output Polarity** *Non-inverted* **Output Synch** *None*	**Reference Voltage** *0.210*	**ClockA** *250 kHz (Chip Clock 0)*
Comparator2 (Comparator v1.1.1)  *Anadigm (Approved)*	**Compare To** *Signal Ground* **Input Sampling** *Phase 1* **Output Polarity** *Non-inverted* **Hysteresis** *0 mV* **Output Synch** *None*		**ClockA** *250 kHz (Chip Clock 0)*

Chip: FPAA2

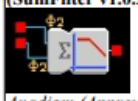
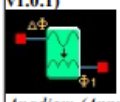
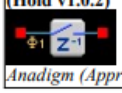


Clocks: FPAA2

Master Clock - ACLK (fc)	*16 MHz*		
System Clock 1 (sys1 = fc / 1)	*16 MHz*	**System Clock 2 (sys2 = fc / 1)**	*16 MHz*
Clock 0 (sys1 / 64)	*250 kHz*	**Clock 1 (sys1 / 4)**	*4 MHz*
Clock 2 (sys1 / 8)	*2 MHz*	**Clock 3 (sys1 / 64)**	*250 kHz*
Clock 4 (sys1 / 1)	*16 MHz*	**Clock 5 (sys1 / 1)**	*16 MHz*

Configurable Analog Modules: FPAA2

Name	Options	Parameters	Clocks
Multiplier1 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 0)* **ClockB** *4 MHz (Chip Clock 1)*
SumFilter2 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Off*	**Corner Frequency [kHz]** *0.899* **Gain 1 (UpperInput)** *0.500* **Gain 2 (LowerInput)** *20.0*	**ClockA** *250 kHz (Chip Clock 0)*
Voltage1 (Voltage v1.0.1)  *Anadigm (Approved)*	**Polarity** *Positive (+2V)*		

SumFilter1 (SumFilter v1.0.2)  *Anadigm (Approved)*	**Output Changes On** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Non-inverting* **Input 3** *Off*	**Corner Frequency [kHz]** *0.925* **Gain 1 (UpperInput)** *1.00* **Gain 2 (LowerInput)** *0.833*	**ClockA** *250 kHz (Chip Clock 0)*	
SumInv1 (SumInv v1.0.1)  *Anadigm (Approved)*	**Input 3** *Off*	**Gain 1 (UpperInput)** *1.00* **Gain 2 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*	
RectifierHalf1 (RectifierHalf v1.0.1)  *Anadigm (Approved)*	**Rectification** *Full Wave* **Polarity** *Non-inverting* **Output Phase** *Phase 1*	**Gain** *1.00*	**ClockA** *250 kHz (Chip Clock 0)*	
Hold3 (Hold v1.0.2)  *Anadigm (Approved)*	**Input Sampling Phase** *Phase 1*		**ClockA** *250 kHz (Chip Clock 0)*	

APENDICE C

C Códigos en Arduino

C.1 Etapa tres y registro de datos de la implementación del TRNG.

```
/*
  Programa simulacion de registro digital y recolección de datos. Envio de
  forma serial
*/
void setup() {
  // Iniciar la comunicacion serial
  Serial.begin(9600);
}
int y = 0;//Variable y del SCC
int yp = 0;//Variable y pasada del SCC

void loop() {
  //Activar por señal digital 2
if  (digitalRead(2)==HIGH){
  y = analogRead(A7);
  //Identificar flanco de subida
  if (y>=600 && yp<600){
    //Funcionamiento como registro digital de 2 bits
    if (analogRead(A5)>600){
      Serial.print(1);
    }else{
```

```
    Serial.print(0);
   }
   if (analogRead(A6)>600){
    Serial.println(1);
   }else{
    Serial.println(0);
   }
  }
  yp = y;
}
}
```