Implementación en FPGA con Salida VGA de un Generador de Osciladores Caóticos en 3D y 4D

M.C. Miguel Angel-Estudillo Valdez[a], Dr. Yuma-Sandoval Ibarra[b], M.C. Andrés-Calvillo Tellez[a], Dr. José Cruz-Núñez Pérez[a].

[a] Instituto Politécnico Nacional, IPN-CITEDI, mestudillo@citedi.mx, calvillo@citedi.mx, nunez@citedi.mx, Tijuana, Baja California, México.
[b] Universidad Politécnica de Lázaro Cárdenas, UPLC, yumasandoval@uplc.edu.mx, Lázaro Cádenas, Michoacán, México.

Resumen

En este artículo se implementa un módulo digital en una tarjeta de FPGA usando una conexión VGA y código VHDL, con el propósito de generar y visualizar el comportamiento de osciladores caóticos en tres y cuatro dimensiones. Para ello se llevó a cabo el diseño de la arquitectura mediante el software Quartus y posteriormente se implementó el circuito en la tarjeta de FPGA Cyclone IV de Intel. Este diseño solo utiliza los estándares IEEE, por lo que se puede implementar en tarjetas de diversos fabricantes. Además, la arquitectura puede ser reconfigurada para adaptarse a diversas resoluciones y en diversos monitores. La interfaz gráfica de usuario permite realizar simulaciones de sistemas caóticos completos, y muestra los canales de comunicación. Con el fin de demostrar el correcto funcionamiento se presentan los resultados del oscilador de Qi y de cuatro alas. La principal contribución de este trabajo de investigación es el diseño de una arquitectura con una interfaz gráfica de usuario que permite generar formas de onda caóticas en 3D y 4D.

Palabras clave— Caos, FPGA, Generador, Interfaz gráfica, VGA.

Abstract

In this article, a digital module is implemented in an FPGA board using a VGA connection and VHDL code, with the purpose of generating and visualizing the behavior of chaotic oscillators in three and four dimensions. Firstly, the architecture design was carried out using the Quartus software and secondly, the circuit was implemented on an FPGA Cyclone IV board from Intel. This design uses only IEEE standards libraries, so it can be implemented on boards from a variety of manufacturers. The architecture can also be reconfigured to accommodate various resolutions and monitors. The graphical user interface allows simulating complete chaotic systems, and shows the communication channels, axis, or dimensions. To demonstrate the correct functioning, the results of the Qi and four wings oscillators are presented. This research work contributes to designing an architecture with a graphical user interface that allows generating chaotic waveforms in 3D and 4D.

Keywords— *Chaos, FPGA, Generator, Graphic interface, VGA.*

1. INTRODUCCIÓN

Actualmente existen múltiples herramientas digitales que permiten a profesores e investigadores analizar y resolver problemas específicos. Ejemplos son los programas de software instalados en una computadora y que usualmente requieren de dispositivos periféricos para complementar los cálculos o las mediciones. Sin embargo, también existen nuevos esquemas en los que las herramientas existentes resultan incompletas o ineficientes, y para resolver esto se recurre a dispositivos embebidos reconfigurables.

Algunos de los dispositivos más populares son las tarjetas de arreglos de compuertas programables en campo (FPGA), debido a su gran versatilidad. La ventaja de estos dispositivos es que permiten diseñar la arquitectura de acuerdo con la tarea a realizar. Adicionalmente, los dispositivos de FPGA tienen versatilidad operativa y baja potencia de consumo, aunado a la velocidad de procesamiento para el procesamiento de video en tiempo real. No obstante, es común que ciertas tareas como el procesamiento de imágenes o video se realicen en co-procesamiento entre una computadora y el dispositivo de FPGA, utilizando un protocolo de comunicación y con esto se limita la velocidad de procesamiento. Una alternativa para resolver este inconveniente es diseñar una interfaz gráfica de usuario (GUI) directamente en la tarjeta de FPGA, y así se eliminan los tiempos de comunicación entre una computadora y la tarjeta de FPGA. Además, un sistema con una interfaz gráfica proporciona mayor portabilidad para un prototipo final.

En la literatura, ya se han reportado proyectos de desarrollo de prototipos que usan controladores de video mediante conexión de arreglos de gráficos de video (VGA). En [1] se explican a detalle los fundamentos del control de video y las señales de sincronización y de color. Sin embargo, la aplicación de este controlador solo se limita a imprimir en pantalla diferentes colores y una imagen a color. Por otro lado, en [2] se lleva a cabo la implementación en tarjetas de FPGA una conexión de VGA para dibujar en pantalla circuitos para filtros. A partir del análisis de la función de transferencia, el programa calcula los elementos del circuito y se dibujan en pantalla. La ventaja de diseñar circuitos en FPGA es que la arquitectura se puede adaptar a la complejidad de la tarea que se desea realizar. Por ejemplo, en [3] se implementa un programa en FPGA con conexión mediante el puerto Transmisor-Receptor Asíncrono Universal (UART) a una computadora personal (PC) para realizar una petición de una imagen. Primero la PC1 solicita a la PC2, mediante una conexión remota de red de área local (LAN), que tome una fotografía con su cámara y la devuelva a PC1. Finalmente la imagen es enviada a la tarjeta de FPGA, mediante un cable VGA, para mostrarla en un monitor. En [4] se implementa una arquitectura para unir los pixeles de una imagen de dos sensores de video diferentes, creando una sola secuencia de pixeles, en donde se pueden procesar fácilmente 30 a 60 fps. En [5] se presenta un sistema implementado en FPGA con conexión VGA para encriptar imágenes en formato rojo-verde-azul (RGB) utilizando los sistemas

caóticos de Rossler y Genesio-Tesi. En [6] se presenta una arquitectura para visión y rastreo de un objetivo a 60 fps y una resolución de 1280 × 1024 pixeles utilizando una tarjeta de FPGA y una cámara para el procesamiento y una interfase multimedia de alta definición (HDMI), para mostrar en una pantalla el video en tiempo real y el objetivo rastreado. En el área médica también se han reportado aplicaciones. En [7] donde se describe el diseño de una interfaz gráfica implementada en System-on-Chip (SoC), basada en pantallas de cristal líquido *touch-screen*, para un sistema de monitoreo cardíaco, se realizan pruebas de electrocardiograma (ECG) y se muestra un diagnóstico al final de la prueba.

En este trabajo de investigación se plantea el diseño de un firmware con una interfaz gráfica orientada a conexión VGA, con aplicación en el campo de las telecomunicaciones y la encriptación de información. La GUI muestra en pantalla los canales de un oscilador de tres y cuatro dimensiones (3D y 4D). El prototipo desarrollado en esta investigación se implementó en lenguaje de descripción de hardware para circuitos integrados de muy alta velocidad (VHDL) para sistemas embebidos basados en FPGA, mediante conexión VGA con el propósito de generar y visualizar diversos osciladores caóticos, en tres y cuatro planos. Para ello se diseñó la arquitectura mediante el software Quartus y posteriormente se implementó el circuito en una tarjeta Intel FPGA Cyclone IV. Actualmente existen herramientas de software que permiten desarrollar interfaces de usuario basadas en FPGA ya sea utilizando lenguaje VHDL o Verilog. Sin embargo, estas requieren del pago de una licencia y además solo se implementan en el hardware del fabricante. El prototipo propuesto en este trabajo se diseñó utilizando únicamente los estándares IEEE por lo que se puede implementar en hardware de diversos fabricantes, tales como Intel y Xilinx. Por otro lado, la arquitectura puede ser modificada para adaptarse a diversas resoluciones y en diferentes monitores. El propósito es la generación de portadoras caóticas y realizar simulaciones de sistemas de comunicación completos, modificando desde el tipo de modulación y demodulación hasta el canal de comunicación. La principal contribución es el diseño de una GUI que permite generar formas de onda caóticas en 3D y 4D.

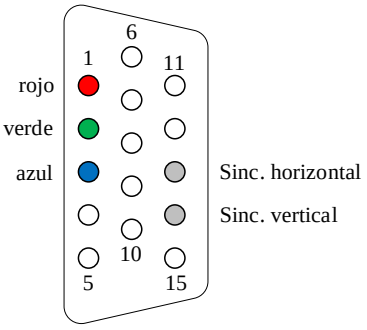
2. MARCO TEÓRICO

En esta sección se explicarán los fundamentos teóricos del diseño de un módulo digital para generar caos en 3D y 4D e implementado en una tarjeta de FPGA, usando un conector VGA.

2.1 Controlador gráfico VGA

El estándar VGA fue creado por IBM en la década de los años 80 y requiere esencialmente de cinco señales para su funcionamiento: sincronización horizontal, sincronización vertical, y las entradas de color rojo, verde y azul. La variación en los tonos de color se consigue cambiando los niveles analógicos de las entradas de RGB. La Figura 1 muestra el diagrama de los pines del estándar VGA, con los cinco pines necesarios para la implementación de las señales.

Fig. 1. Diagrama de pines de un conector VGA.



Fuente: elaboración propia a partir de “referencia”.

Para el ojo humano una actualización de 30 Hz da la sensación de continuidad en una imagen. Por lo tanto, la pantalla debe tener como mínimo esa frecuencia de actualización. La frecuencia de reloj mínima requerida por el dispositivo de FPGA se puede calcular con la ecuación (1),

$$f = p \times l \times s, \tag{1}$$

en donde p corresponde al ancho de la pantalla en pixeles, l representa la altura de la pantalla en líneas y s la frecuencia de actualización de la pantalla. La interfaz de esta propuesta tiene una resolución de 1024 x 768 pixeles a una frecuencia de actualización de 60 Hz, por lo tanto, el reloj de píxel necesario debe ser de 50 MHz. En la Tabla 1 se muestra la distribución de tiempo y pixeles para el ancho y alto de la pantalla.

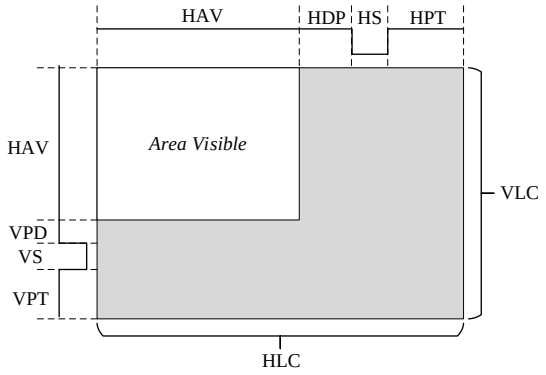
Tabla 1. Tiempos de barrido horizontal y vertical para la interfaz VGA.

Barrido	Nombre	Porción	Longitud	Tiempo
Horizontal	HAV	Área visible	1024 pixeles	20.48 µs
	HPD	Porción delantera	10 pixeles	0.2 µs
	HS	Pulso de sincronización	26 pixeles	0.52 µs
	HPT	Porción trasera	40 pixeles	0.8 µs
	HLC	Línea completa	1100 pixeles	22 µs
Vertical	VAV	Área visible	700 líneas	15.4 ms
	VPD	Porción delantera	0 líneas	0 Ms
	VS	Pulso de sincronización	8 líneas	0.176 ms
	VPT	Porción trasera	50 líneas	1.1 ms
	VLC	Línea completa	758 líneas	16.676 ms

Fuente: elaboración propia.

Por otro lado, en la Figura 2 se indican las regiones visibles y no visibles de una pantalla de cristal líquido (LCD), así como las nomenclaturas presentadas en la Tabla 2. En las regiones no visibles ocurre la sincronización de la nueva línea de pixeles y en la región visible se muestran los pixeles en diferentes tonos de color.

Fig. 2. Diagrama de tiempos para el control de video mediante VGA.

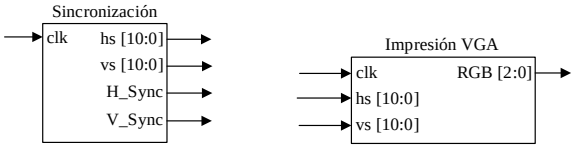


Fuente: elaboración propia.

2.2 Arquitectura del circuito

El módulo de sincronización se encarga de indicar la posición actual del barrido en la pantalla mediante las señales *hs* y *vs*. Adicionalmente este indica cuando el barrido se encuentra dentro de la zona visible de la pantalla mediante las señales *H_Sync* y *V_Sync*, las señales involucradas se muestran en la Figura 3(a). En la Figura 3(b) se muestra el módulo de impresión usando conexión VGA, este indica el color de cada píxel dependiendo de la posición del barrido y de la imagen que se desea imprimir en pantalla, el color se determina mediante la salida en formato RGB de 3 bits.

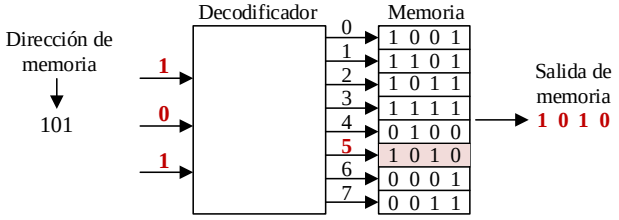
Fig. 3. Módulos para el control de mediante conexión VGA a) sincronización; b) impresión de 3 bits.



Fuente: elaboración propia.

Para almacenar la información de los elementos a mostrar en pantalla se crean memorias, las cuales se pueden crear en forma de vectores o matrices de datos, por ejemplo una imagen se puede almacenar en forma de matriz. Posteriormente esta imagen puede ser referenciada a una zona específica de un monitor con estándar VGA e imprimirla en pantalla. La estructura de una memoria depende de las dimensiones de ancho y profundidad, donde la multiplicación de ambos parámetros determina el tamaño de la memoria. Para recuperar los datos de la memoria se debe realizar una petición de lectura indicando la dirección del dato solicitado. Y para extraer la información de las memorias se utilizan módulos decodificadores, estos convierten una dirección binaria en un índice de posición del renglón solicitado. En la Figura 4 se muestra un esquema del decodificador y la selección del renglón de información dentro de una memoria de 4×8.

Fig. 4. Esquema de un decodificador de dirección de memoria.



Fuente: elaboración propia.

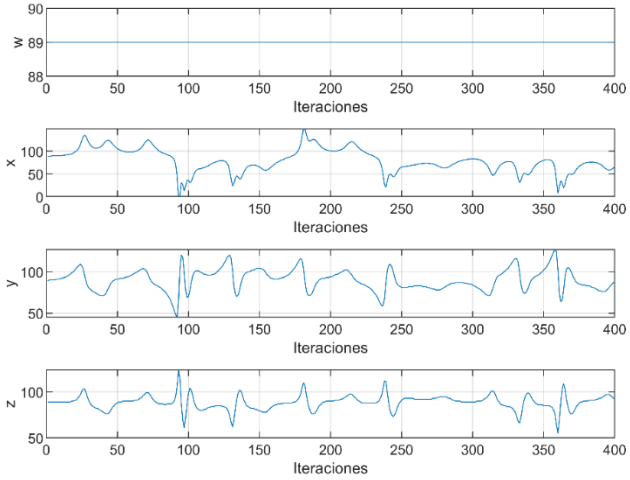
2.3 Osciladores caóticos

A continuación, se presentan los resultados de dos osciladores caóticos, en 3D y 4D, respectivamente. El oscilador caótico de cuatro alas presentado en [8] se genera utilizando el sistema de ecuaciones (2), con los valores de parámetros $a = 14, b = 43, c = -1, d = 16, e = 4$ y las condiciones iniciales $x(1) = 1, y(1) = 1, z(1) = 1$.

$$\begin{cases} \dot{x} = a(y - x) + eyz \\ \dot{y} = cx + dy - xz \\ \dot{z} = xy - bz \end{cases} \quad (2)$$

En la Figura 5 se muestran los canales del oscilador caótico de cuatro alas, en este caso el eje ó dimensión *w* se mantiene inactivo, debido a que solo se requieren 3 de los cuatro ejes disponibles.

Fig. 5. Canales de un oscilador caótico de cuatro alas simulado en Matlab.



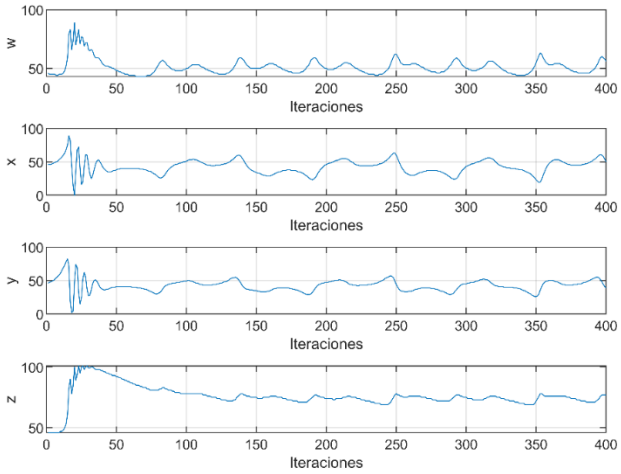
Fuente: elaboración propia a partir del sistema (2).

Por otro lado, para generar las formas de onda del oscilador Qi de 4D, presentado en [9], se utiliza el sistema de ecuaciones (3), con los valores de parámetros $a = -10, b = 30, c = 10$, y las condiciones iniciales $w(1) = 1,\ x(1) = 1,\ y(1) = 1,\ z(1) = 1$.

$$\begin{cases} \dot{w} = aw + xyz \\ \dot{x} = b(y - x) + yzw \\ \dot{y} = c(x + y) - xzw \\ \dot{z} = -z + ywx \end{cases} \quad (3)$$

La Figura 6 muestra las señales generadas en Matlab de las cuatro dimensiones del oscilador caótico Qi en 4D.

Fig. 6. Ejemplo de figura en un artículo.

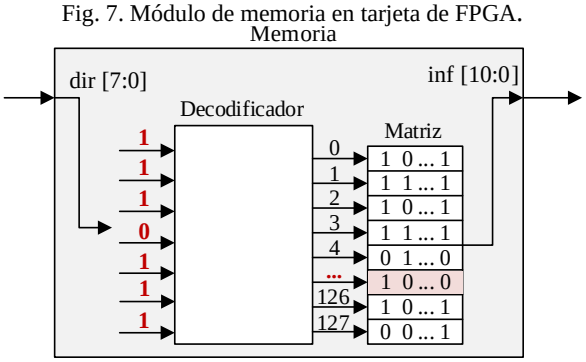


Fuente: elaboración propia a partir del sistema (3).

3. IMPLEMENTACION EN FPGA

El diseño inicia con la creación de módulos de memoria para guardar los diferentes elementos de la interfaz gráfica. La figura 7 muestra el módulo de memoria, la solicitud se hace mediante una dirección de 8 bits, los cuales entran a un decodificador y convierten la dirección de binario a entero. Este índice indica cual renglón de la matriz de datos será enviado a la salida, donde cada renglón de memoria tiene un ancho de 11 bits

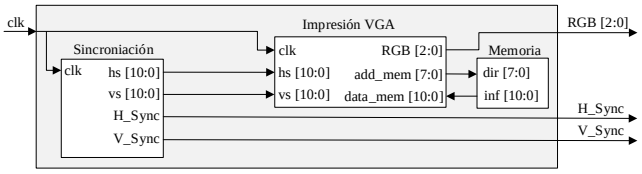
Fig. 7. Módulo de memoria en tarjeta de FPGA.



Fuente: elaboración propia.

Los bloques de memoria complementan a los de sincronización e impresión VGA para indicar los elementos que han de imprimirse y las coordenadas dentro de la región visible de la pantalla. De este modo se pueden imprimir en pantalla desde líneas, hasta imágenes completas. El circuito completo incluye el bloque de memoria y se muestra en la Figura 8. El módulo de "Impresión VGA" indica la posición actual en pantalla, y en el módulo de memoria previamente se cargaron los datos de imagen o señales. Si la posición actual del barrido coincide con alguna de las posiciones indicadas en la memoria se imprime un píxel de color en la pantalla.

Fig. 8. Diagrama a bloques para el control de video.

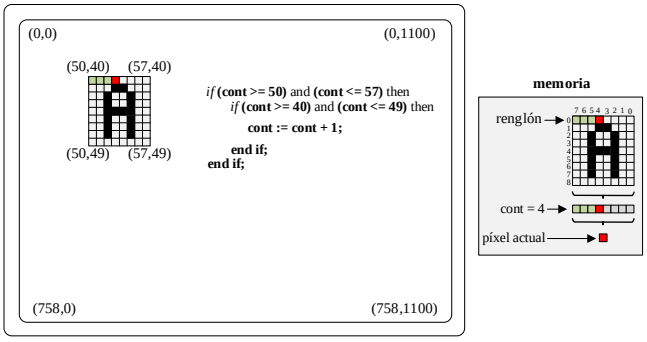


Fuente: elaboración propia.

Por otra parte, el barrido de la pantalla se realiza de izquierda a derecha y de arriba hacia abajo. Cabe destacar que, aunque el barrido se hace en toda la región de la pantalla, la imagen solo se mostrará en la región visible de la pantalla. El barrido se lleva a cabo mediante un contador que reinicia cada que se alcanzan los límites superior e inferior.

Para imprimir el contenido de una memoria se deben referenciar a la pantalla las coordenadas donde se imprimirá la porción de memoria. En la Figura 9 se muestra una matriz para la letra "A" guardada en una memoria de 9 × 8. Esta memoria ha sido asignada para imprimirse en el área indicada por la cuadrícula, posteriormente se asigna un contador que solo se active dentro de esa zona para llevar las posiciones de la memoria.

Fig. 9. Impresión en pantalla de un elemento de memoria.



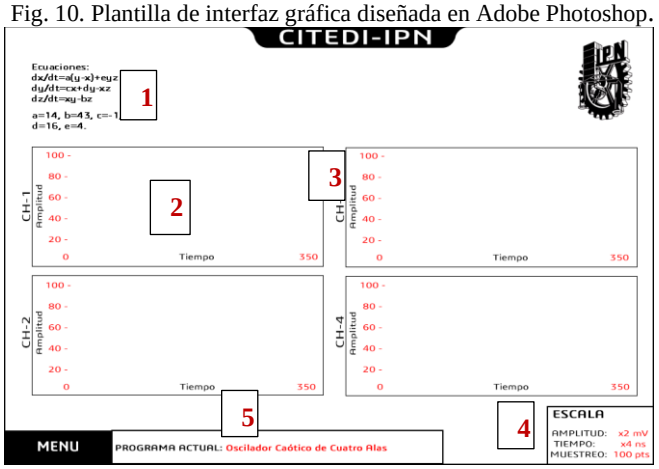
Fuente: elaboración propia.

4. RESULTADOS

La metodología que se siguió fue primeramente crear la plantilla de interfaz gráfica en Adobe Photoshop debido a las herramientas prácticas de diseño. Posteriormente se tomó el diseño creado como guía y se diseñaron cada uno de los componentes en código VHDL. Después se cargaron los osciladores caóticos en el módulo de memoria: un oscilador de cuatro alas en tres dimensiones y un oscilador de Qi en cuatro dimensiones. Al final se imprimieron en las regiones de cada señal de la interfaz gráfica.

La Figura 10 muestra los componentes de la interfaz gráfica, en donde: 1) representa las ecuaciones diferenciales del oscilador caótico que se va a graficar; 2) es la región en donde se va a imprimir; 3) indica el canal, eje o dimensión que se está graficando, ya sea un oscilador de 3 o 4 canales; 4) representa las escalas de los ejes de impresión tanto vertical como horizontal; y 5) indica el nombre del programa en ejecución.

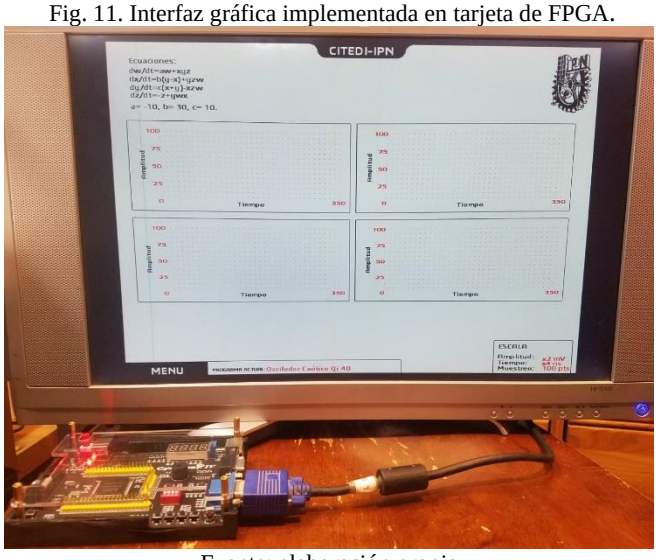
Fig. 10. Plantilla de interfaz gráfica diseñada en Adobe Photoshop.



Fuente: elaboración propia.

La implementación se realizó en la tarjeta de FPGA Cyclone IV EP4CE6E22C8. La Figura 11 muestra la implementación en FPGA, usando una pantalla LCD con resolución 1024 × 700 pixeles y frecuencia de 60 Hz.

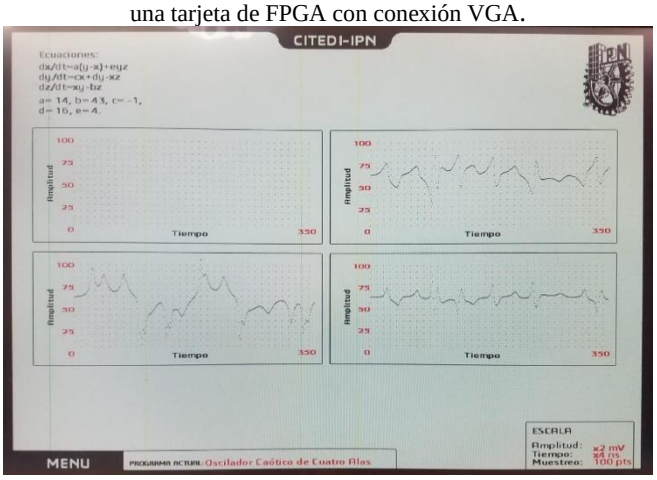
Fig. 11. Interfaz gráfica implementada en tarjeta de FPGA.



Fuente: elaboración propia.

Adicionalmente, en la Figura 12 se muestra el oscilador caótico de cuatro alas, con los canales *x* (arriba izquierda), *y* (abajo izquierda), *z* (arriba derecha).

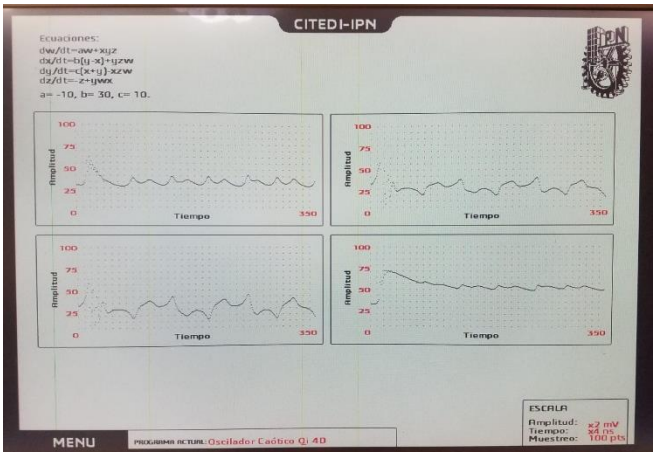
Fig. 12. Canales de un oscilador caótico de cuatro alas de 3D en una tarjeta de FPGA con conexión VGA.



Fuente: elaboración propia a partir del sistema (2).

Mientras que en la Figura 13 se muestran los cuatro canales del oscilador de Qi generadas en una tarjeta de FPGA. Las señales mostradas en el monitor con conexión VGA son los canales *w* (arriba izquierda), *x* (abajo izquierda), *y* (arriba derecha), *z* (abajo derecha).

Fig. 13. Canales de un oscilador Qi de 4D en una tarjeta de FPGA con conexión VGA.



Fuente: elaboración propia a partir del sistema (3).

Los recursos lógicos requeridos por cada uno de los módulos se muestran en la Tabla 2.

Tabla 2. Recursos lógicos del módulo implementado en FPGA.

Recurso	Disponible	Modulo Sincronización	Módulo VGA	Total utilizado
Registros	460	46	314	360(78.2%)
LUT	6,272	54	4706	4,841(76%)
I/O	92	0	0	10 (11%)
DSP	30	0	0	0
Memoria	276,480	0	0	0
PLLs	2	0	0	0

Fuente: elaboración propia.

En este diseño las escalas de los ejes son estáticos y no se adaptan a la amplitud real de la forma de onda, y no es posible reconfigurar el oscilador desde la interfaz gráfica.

5. CONCLUSIONES Y RECOMENDACIONES

En este proyecto se desarrolló un firmware en tarjeta de FPGA que tiene como objetivo generar y mostrar en pantalla formas de onda caóticas con fines experimentales. Para ello se diseñó una interfaz gráfica para los canales de osciladores caóticos en 3D y 4D. La interfaz gráfica se implementó en lenguaje VHDL, en una resolución de 1024 × 768 pixeles y una frecuencia de pantalla de 60 Hz, con una frecuencia de reloj de FPGA de 50 MHz. La principal contribución de esta interfaz gráfica es su portabilidad, debido a que utiliza únicamente estándares IEEE y es compatible para Intel y Xilinx. Además, el diseño presentado en esta investigación no requiere de un osciloscopio digital ni de DAC's para visualizar las formas de onda del oscilador, en su lugar maneja una interfaz gráfica por VGA para observar los canales del oscilador directamente del FPGA en la pantalla. Las secuencias caóticas pueden ser enviadas a la salida del dispositivo FPGA para utilizarse en diversos procesos como esquemas de modulación, demodulación, generación de secuencias pseudoaleatorias, enmascaramiento de señales, entre otras múltiples aplicaciones.

6. REFERENCIAS

[1] C.A. Ramos-Arreguín, et. al., FPGA Open Architecture Design for a VGA Driver, The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science, Vol. 2012, No. 3, Pp. 324-333, 2012.

[2] M. Rejab Khalil, S. Mohammed Ali, Designing FPGA based VGA system to display the component of synthesized electrical circuits using embedded design techniqus, International Conference on Electrical Communication, Computer, Power, and Control Engineering, Mosul Iraq, del 17 al 18 de diciembre del 2013.

[3] Z. Syed, M. Shaik, FPGA Implementation of VGA Controller, International Conference on Electronics and Communication Engineering, Pune, India, 16 de septiembre del 2012.

[4] A. Jose, et. al., FPGA Based Novel Architecture for Real-time Video Stitching, Innovations in Power and Advanced Computing Technologies, Kuala Lumpur, Malaysia, del 27 al 29 de noviembre del 2021.

[5] R. Kaibou, M. Salah Azzaz, FPGA Implementation of Mixed Robust Chaos-based Digital Color Image Watermarking, Annaba, Algeria, del 27 al 28 de octubre del 2021.

[6] R. Morris, S. Mirzaei, Efficient FPGA Implementation of Parameterized Real Time Color Based Object Tracking, Vancouver, BC, Canada, del 27 al 30 de octubre del 2021.

[7] W. Yong Chia, et. al., On-board touch screen graphical interface design for SoC-based arrhythmia detector, Penang, Malaysia, del 19 al 21 de agosto del 2014.

[8] G. Qi, G. Chen, "A spherical chaotic system", Nonlinear Dynamics, Vol. 2015, No. 81, Pp. 1381-1392, 2015.

[9] Q. Lai, T. Nestor, J. Kengne, X. W. Zhao, "Coexisting attractors and circuit implementation of a new 4D chaotic system with two equilibria", Chaos, Solitons and Fractals, Vol. 107, Pp. 92-102, 2018.