Implementación de red neuronal tipo Hebbiana en FPGA

Fernando Evier López-Pérez[a], Teodoro Alvarez-Sánchez[b*], David J Saucedo[c], Roberto Herrera-Charles[d].

[a] Instituto Politécnico Nacional, flopezp@ipn.mx, México.
[*b] Instituto Politécnico Nacional, talvarezs@ipn.mx, México.
[c] Instituto Politécnico Nacional, dsaucedo@ipn.mx, México.
[d] Instituto Politécnico Nacional, rcharles@ipn.mx, México.
*Corresponding author

Resumen

El desarrollo de redes neuronales artificiales en entornos de descripción de hardware es un área de creciente interés en la inteligencia artificial, especialmente en aplicaciones de procesamiento digital hacia la inteligencia artificial y otras áreas afines. Este trabajo explora la implementación de una red neuronal artificial en VHDL para la simulación de neuronas artificiales, demostrando su viabilidad en sistemas digitales programables.

En la metodología, se diseñó e implementó una neurona tipo hebbiana en una red neuronal del mismo tipo. La topología consta de una capa de entrada con cuatro neuronas, dos capas ocultas para el procesamiento intermedio y una capa de salida encargada de generar la operación lógica.

Los resultados obtenidos muestran que la red neuronal implementada en VHDL es capaz de simular correctamente las compuertas lógicas propuestas, validando su funcionamiento a través de pruebas en un entorno de simulación. Además, la implementación demuestra estabilidad, eficiencia en términos de recursos de hardware utilizados y consumo de energía por la red neuronal artificial.

Se concluye que el uso de redes neuronales en entornos de descripción de hardware es una estrategia viable para el desarrollo de sistemas inteligentes. Este enfoque permite explorar nuevas aplicaciones en lógica programable y en el diseño de circuitos con capacidad de aprendizaje adaptativo.

Palabras clave— Red *Neuronal, VHDL, FPGA, Hebbiana*

Abstract

The development of artificial neural networks in hardware description environments is an area of growing interest in artificial intelligence, particularly in applications of digital processing toward artificial intelligence and other related fields. This work explores the implementation of an artificial neural network in VHDL for simulating artificial neurons, demonstrating its feasibility in programmable digital systems.

In the methodology, a Hebbian-type neuron was designed and implemented in a neural network of the same type. The topology consists of an input layer with four neurons, two hidden layers for intermediate processing, and an output layer responsible for generating the logical operation.

The results obtained show that the neural network implemented in VHDL is capable of correctly simulating the proposed logic gates, validating its operation through tests in a simulation environment. Furthermore, the implementation demonstrates stability and efficiency in terms of hardware resources used and energy consumption by the artificial neural network.

It is concluded that the use of neural networks in hardware description languages is a viable strategy for developing intelligent systems. This approach enables the exploration of new applications in programmable logic and the design of circuits with adaptive learning capabilities.

Keywords— *Neural Networks, VHDL, FPGA, Hebbiana*

1. INTRODUCCIÓN

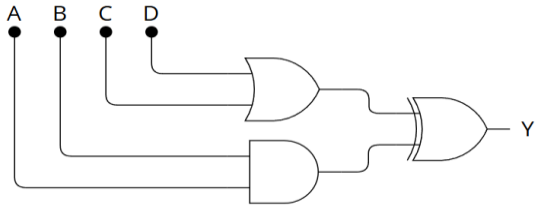
Las redes neuronales artificiales implementadas en hardware tienen un gran potencial como herramientas para simular diversos procesos computacionales, incluyendo las operaciones lógicas fundamentales. La implementación de compuertas lógicas a través de redes neuronales artificiales (RNA) en VHDL no solo resalta la versatilidad de estos sistemas, sino que también establece una base sólida para comprender cómo las RNA pueden ser eficientemente integradas en hardware para replicar operaciones lógicas complejas.

En términos prácticos, esta capacidad de replicar operaciones lógicas convierte a las RNA en recursos valiosos para el desarrollo de sistemas de hardware más inteligentes y eficientes. Estas redes aprenden a imitar el comportamiento de las compuertas lógicas a través del aprendizaje de patrones de entrada y salida, combinando principios matemáticos y de programación que les permiten realizar operaciones lógicas fundamentales.

Este trabajo se enfoca en la implementación en VHDL de una red neuronal artificial de cuatro capas que replica el funcionamiento de un arreglo específico de compuertas lógicas. La arquitectura propuesta utiliza neuronas hebbianas, que se basan en el principio de que las conexiones entre neuronas se fortalecen cuando se activan simultáneamente, facilitando así el aprendizaje de patrones [1].

En este caso, el sistema que se busca replicar mediante la red neuronal consiste en un arreglo de compuertas lógicas con cuatro entradas (A, B, C y D). Las entradas A y B se conectan a una compuerta AND, mientras que las entradas C y D se conectan a una compuerta OR. La salida de ambas compuertas se conecta a las entradas de una compuerta XOR, cuya salida corresponde a la salida de la red neuronal. La figura 1 ilustra el diagrama del arreglo de compuertas a replicar.

Fig. 1. Diagrama de arreglo de compuertas a replicar.



Fuente: elaboración propia.

Un beneficio clave de replicar un sistema como este, se puede prever con precisión cuál será la salida en función de las entradas, al consultar la tabla de verdad correspondiente para el arreglo de compuertas. Esta tabla de verdad se presenta en la tabla 1.

Tabla 1. Tabla de verdad de sistema a replicar.

A	B	C	D	Salida
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

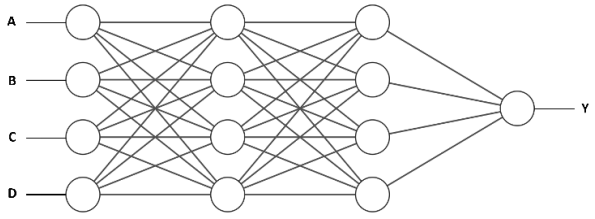
Fuente: elaboración propia.

2. METODOLOGÍA

El desarrollo y la implementación de la red neuronal se llevarán a cabo en la tarjeta de desarrollo FPGA AC701 de Xilinx, que cuenta con un chip Artix 7 (xc7a200tfbg676-2).

La estructura de la red se ha diseñado de manera que incluya una capa de entrada con 4 neuronas, dos capas ocultas, cada una con 4 neuronas, y una capa de salida con una única neurona. Esta configuración se elige para reflejar las 4 entradas del sistema y su correspondiente salida. La figura 2 ilustra la red neuronal propuesta [2] [4].

Fig. 2. Red neuronal propuesta.



Fuente: elaboración propia.

3. IMPLEMENTACIÓN

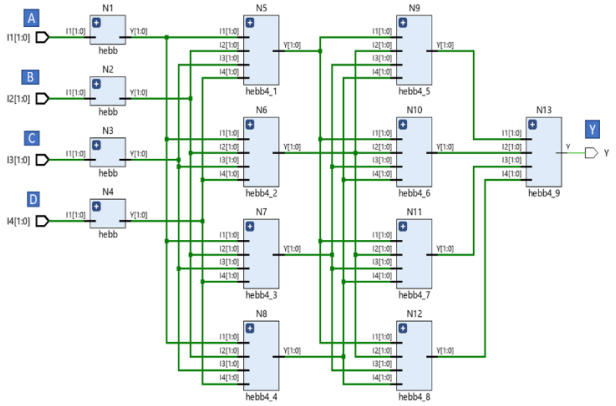
Primero, se llevó a cabo la construcción de dos tipos de neuronas hebbianas en VHDL. La primera neurona tiene una única entrada y una salida, mientras que la segunda cuenta con 4 entradas y una salida. La primera neurona se utilizará en la capa de entrada de la red neuronal, mientras que la segunda será empleada en las capas ocultas y de salida.

A continuación, se calcularon los pesos necesarios para que la red neuronal artificial replicara el comportamiento del diagrama presentado en la figura 1. Inicialmente, se realizó un análisis detallado, seguido del diseño del programa en VHDL para el entrenamiento de la red. Dado que el comportamiento de esta implementación era relativamente predecible, gracias a la experiencia en el manejo de compuertas lógicas, se definieron los pesos de forma manual. Este enfoque resultó en un rendimiento óptimo, sin necesidad de utilizar recursos computacionales adicionales para su obtención [3].

Es importante mencionar que la validación de los pesos se llevó a cabo mediante simulaciones en VHDL. Además, se desarrolló un programa en Matlab 2024 que replicaba el funcionamiento matemático de la red, permitiendo realizar pruebas para evaluar los pesos seleccionados.

Una vez definida la estrategia y los pesos de la red neuronal artificial, se procedió a programar en VHDL las conexiones correspondientes. El resultado de este programa se puede observar en la figura 3 [5][6].

Fig. 3. Diagrama de bloques resultante para red neuronal en VHDL.



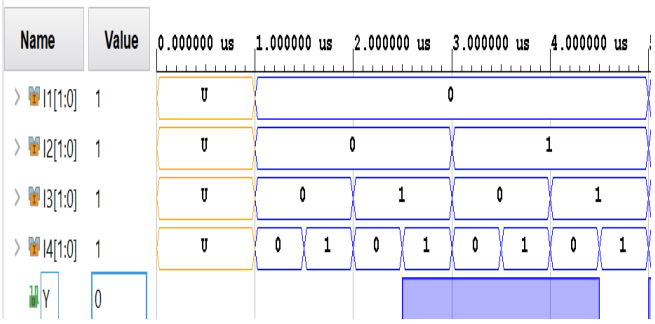
Fuente: elaboración propia.

4. RESULTADOS

A continuación, se presentan los resultados detallados de la simulación llevada a cabo para verificar la funcionalidad de la red neuronal. En este proceso, se introdujeron valores específicos conforme a la tabla de verdad que se muestra en la Tabla 1. Este enfoque permite evaluar de manera exhaustiva el comportamiento y la precisión de la red neuronal bajo diferentes condiciones de entrada.

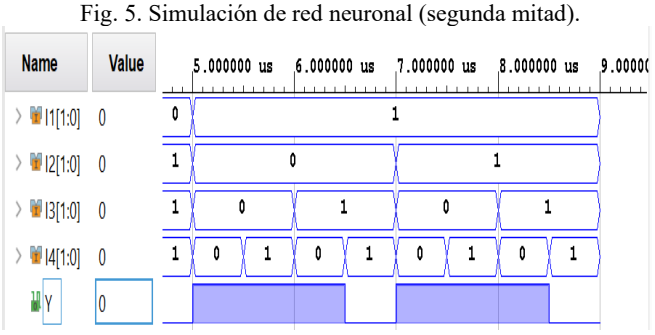
La Figura 4 ilustra los resultados obtenidos durante la primera mitad de la simulación, donde se pueden observar las respuestas iniciales del sistema ante los valores introducidos. Por otro lado, la Figura 5 representa los resultados correspondientes a la segunda mitad de la simulación, lo que permite comparar y analizar cómo evoluciona el rendimiento de la red a medida que se procesan más datos.

Fig. 4. Simulación de red neuronal (primera mitad).



Fuente: elaboración propia.

Fig. 5. Simulación de red neuronal (segunda mitad).



Fuente: elaboración propia.

Además, en la Tabla 2 se detallan los recursos utilizados para implementar el diseño en un FPGA Artix 7. Esta información es crucial para entender las capacidades del hardware empleado y su adecuación para ejecutar las tareas requeridas por la red neuronal.

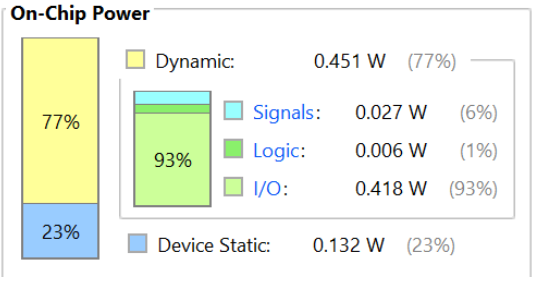
Tabla 2. Recursos utilizados por Artix 7 (xc7a200tfbg676-2).

Recursos	Utilización	Disponibles	Utilización %
LUT	2	133800	0.01
IO	9	400	2.25

Fuente: elaboración propia.

Por último, se incluye un informe sobre el consumo energético del FPGA, representado en la Figura 6. En este informe se observa que el dispositivo tiene un consumo total aproximado de 0.583 Watts. De este total, un notable 73% corresponde al consumo dinámico del dispositivo, lo que indica una alta actividad durante el procesamiento de datos. Es relevante destacar que, dentro de este porcentaje, un impresionante 93% del consumo dinámico se atribuye a las entradas y salidas del sistema, lo que resalta la importancia de optimizar estas operaciones para mejorar la eficiencia energética general del sistema. Todos estos datos fueron recopilados a una temperatura ambiente controlada de 26.1°C [2], lo cual es fundamental para garantizar condiciones estables durante las pruebas y asegurar resultados reproducibles.

Fig. 6. Consumo de energía del FPGA (xc7a200tfbg676-2).



Fuente: elaboración propia.

3. CONCLUSIONES Y RECOMENDACIONES

Para el sistema construido, se logró una utilización de recursos notablemente baja, con un 0.01% para las tablas de revisión disponibles y un 2.25% para las entradas y salidas del FPGA. Esta eficiencia en el uso de recursos sugiere que hay margen para optimizar aún más el diseño. Específicamente, al modificar la configuración de las entradas en las neuronas, sería posible reducir aún más el uso de entradas y salidas del sistema, lo que podría resultar en una implementación más eficiente.

En términos de rendimiento, la red neuronal demostró resultados satisfactorios al replicar completamente el arreglo de compuertas propuesto. La simulación tuvo una duración de 9 µs, lo cual se debe a los tiempos seleccionados durante la prueba; se estableció un intervalo de 500 ns entre cada cambio en las entradas. Sin embargo, es importante destacar que este no es el límite del sistema. Se realizaron pruebas adicionales con intervalos tan cortos como 5 ns entre cambios y los resultados continuaron siendo correctos, lo que indica una robustez considerable en el funcionamiento del sistema.

El informe de temporización no presentó problemas relacionados con las restricciones impuestas por la tarjeta FPGA utilizada, lo que respalda la viabilidad de su implementación en aplicaciones prácticas. En cuanto a la posibilidad de implementar este sistema en otra tarjeta FPGA, se considera completamente factible debido a su bajo consumo de recursos. Dado que la red no es excesivamente compleja, el procesamiento de datos dentro de esta no debería afectar significativamente el funcionamiento general del sistema.

Sin embargo, es importante tener en cuenta que el tiempo de simulación podría verse afectado por las diferencias en los relojes operativos de los distintos FPGA que pudieran utilizarse. Este impacto sería relevante únicamente si el reloj operativo del nuevo FPGA fuera inferior a 2 MHz; en tal caso, manejar un intervalo de 500 ns entre cambios en las entradas podría resultar inviable. Por lo tanto, al considerar la migración a otro dispositivo FPGA, es crucial evaluar sus

especificaciones técnicas para asegurar un rendimiento óptimo y evitar limitaciones en la operación del sistema.

Agradecimientos

Agradecemos las facilidades otorgadas para la realización de este trabajo al Instituto Politécnico Nacional, a través de la Secretaría de Investigación y Posgrado con los proyectos SIP 20241430 y SIP 20241219. A la Unidad Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas y al Centro de Investigación y Desarrollo de Tecnología Digital. Asimismo, al Programa de Estímulo al Desempeño de los Investigadores (EDI) y al Programa de Estímulo al Desempeño Docente (EDD).

4. REFERENCIAS

[1] Haykin, S. (2009). "Neural Networks and Learning Machines", 3rd Edition. Pearson Education, New Jersey.
[2] Lee, M., hwang, K., Park, j., Choi, S., Shin, S. y Sung, W., 2016. FPGA-Based Low-Power Speech Recognition with Recurrent Neural Networks. 2016 IEEE International Workshop on Signal Processing Systems (SiPS). S.l.: s.n., pp. 230-235. DOI 10.1109/SiPS.2016.48.
[3] XILINX, 2020. Vivado Design Suite User Guide: High-Level Synthesis (UG902) [en línea]. 2020. [Consulta: 11 enero 2025]. Disponible en: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug902-vivado-high-levelsynthesis.pdf.
[4] McCulloch, W. S., & Pitts, W. (1990). A logical calculus of the ideas immanent in nervous activity. 1943. Bulletin of mathematical biology, 52(1-2), 99–97.
[5] W. Stallings, Computer architecture and organization: Designing for performance, 4 ed., Englewood Cliffs, New Jersey: Prentince hall, 1996, p. 684.
[6] S. Haykin, Neural networks: A comprehensive foundation, 1 ed., Englewood Cliffs, New Jersey: Mcmillan Publishing Company, 1994, p. 696.