VISIÓN DE CASO

Clasificación del artículo:
INVESTIGACIÓN

IMPLEMENTACIÓN DE REDES NEURONALES UTILIZANDO DISPOSITIVOS LÓGICOS PROGRAMABLES

Jaime Alberto Parra Plaza*
Danilo Ramos Zapata**
Alejandra Tigreros Tascón***

Resumen

Desde el comienzo de la revolución computacional hubo interés por las máquinas inteligentes. Después del fracaso de los métodos de búsqueda de inteligencia artificial aplicados a problemas reales, un nuevo enfoque: *inteligencia computacional* retomó el camino basándose en estructuras copiadas de la naturaleza. Las redes neuronales (artificiales) emulan de manera simplificada el funcionamiento de las redes biológicas. La investigación ha estado dirigida primordialmente a simulación en computador de los algoritmos que se proponen, lo que se aleja del comportamiento real de las neuronas biológicas. Este artículo muestra un desarrollo en el que se implementó un sistema para diseñar, simular, crear e interactuar con redes neuronales en un dispositivo físico, para acercarse al concepto biológico. El dispositivo de trabajo (un FPGA) es un arreglo de elementos lógicos, cuyos puntos de interconexión pueden ser programados, lo que modela mejor la evolución de un cerebro biológico.

Palabras clave
Inteligencia computacional, diseño de redes neuronales, implementación de redes neuronales, sistemas bio-inspirados, FPGA, VHDL.

Abstract
From the beginning of the computational revolution there was interest for the intelligent machines. After the failure of the searching methods of Artificial Intelligence applied to real problems, a new approach, Computational Intelligence, recaptured the road basing on structures copied from nature. The (artificial) neural networks emulate in a simplified way the operation of the biological networks. Research has been directed primarily to computer simulation

* Ingeniero Electricista de la Universidad del Valle en 1991 y Magíster en Automática de la misma universidad en 1997. Docente de la Facultad de Ingeniería en la Pontificia Universidad Javeriana Cali, coordinador del área de Sistemas Digitales y Microprocesadores. Miembro del Grupo de Automática y Robótica (GAR). Correo electrónico: jparra@puj.edu.co.

** Ingeniero electrónico de la Pontificia Universidad Javeriana Cali. Miembro del Grupo de Automática y Robótica (GAR). Correo electrónico: dramoz@usa.net.

*** Ingeniera electrónica de la Pontificia Universidad Javeriana Cali. Miembro del Grupo de Automática y Robótica (GAR). Correo electrónico: atigreros@usa.net.

of the proposed algorithms, which moves away from the real behavior of the biological neurons. This article shows a development of a system for designing, simulating, creating, and interacting with neural networks implemented in a physical device, to come closer to the biological concept. The target device, a FPGA, is an arrangement of logical elements, whose interconnection points can be programmed, which better models the evolution of a biological brain.

Keywords
Computational intelligence, neural networks design, neural networks implementation, bio-inspired systems, FPGA, VHDL.

1. Introducción

El uso de redes neuronales ocupa hoy por hoy un lugar muy importante en la solución de problemas complejos, como: clasificación de patrones [1], modelamiento y control [4] y procesamiento de señales [5]. La mayoría de las implementaciones de redes neuronales son simuladas en computador, subutilizando una máquina poderosa en la ejecución de una sola labor y restringiendo la movilidad del sistema.

La implementación de redes neuronales en FPGA presenta una posible solución a este problema [2] [9]. Teniendo en cuenta que los niveles de integración y las prestaciones de las FPGA aumentan consistentemente cada año, es viable suponer que en un futuro cercano esta aproximación hará válido el desarrollo de sistemas bioinspirados soportados por hardware reconfigurable [7].

El presente artículo describe un sistema que le permite al diseñador de redes neuronales validar y verificar sus diseños en hardware reconfigurable, sin tener que conocer la arquitectura de dicho hardware ni los lenguajes de descripción para programarlos, en contraste con herramientas similares [10].

2. Consideraciones de diseño

Una red neuronal artificial (NN) es una estructura compuesta por un conjunto de elementos denominados neuronas, relacionadas entre sí mediante diferentes valores de conexiones. La red genera patrones de salida como respuesta a impulsos o entradas, en los que la relación entre las entradas y las salidas es no lineal [6]. A continuación se muestra un diagrama Top-Down que sirve de base conceptual para describir el proyecto.

Diagrama 1. Diagrama Top-Down de una red neuronal

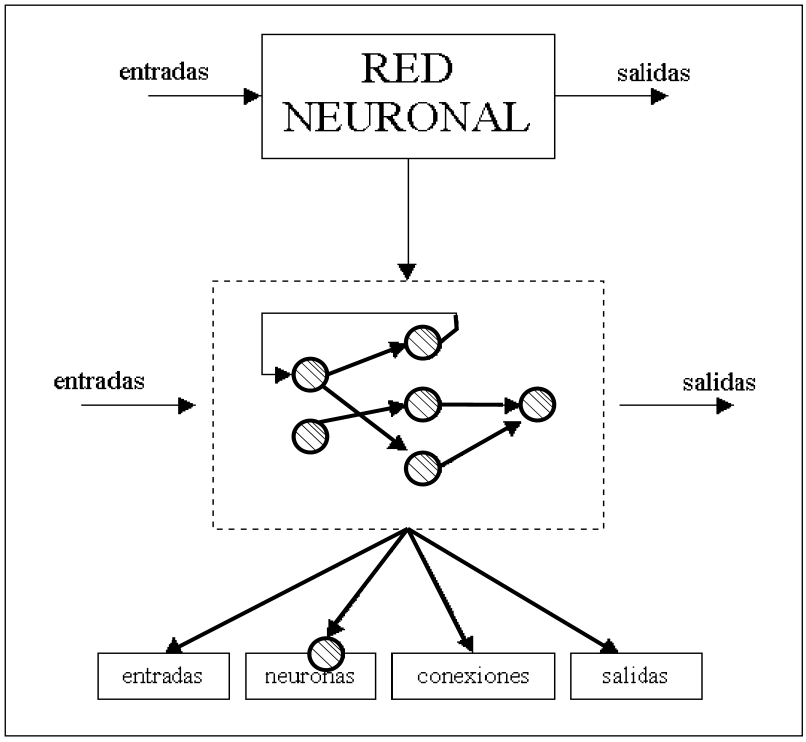
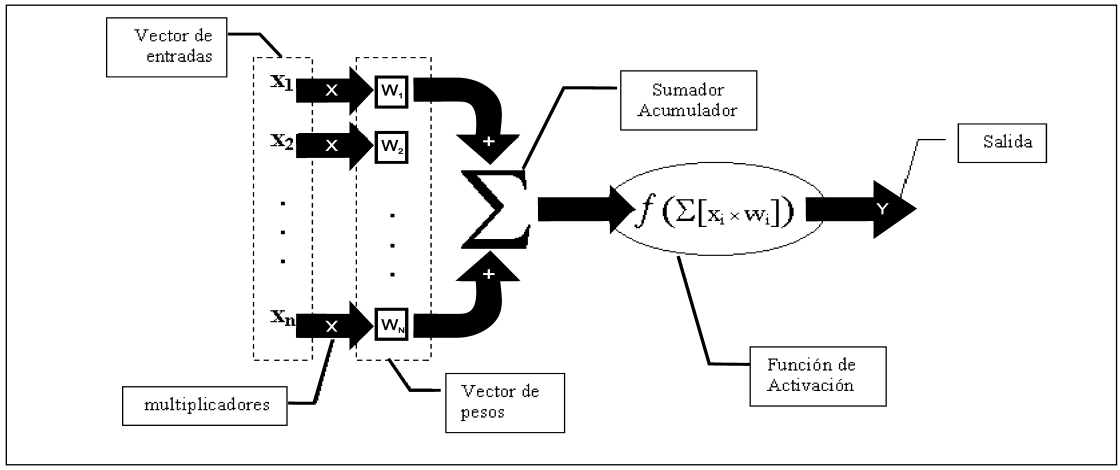


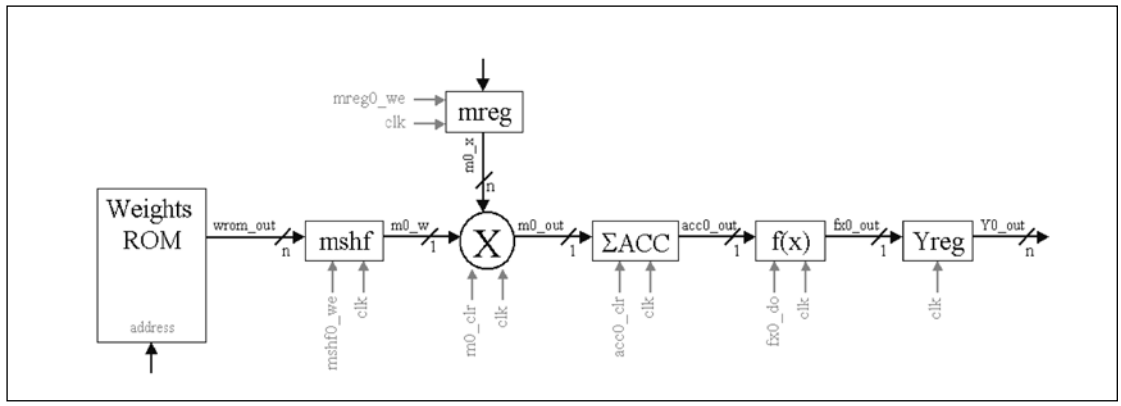
Diagrama 2. Modelo matemático de una neurona



Tomando la neurona como la entidad básica de una red neuronal, se plantea la estructura del diagrama 3, a partir del modelo matemático del diagrama 2. La representación de la información numérica se hace en punto fijo, teniendo en cuenta las restricciones que impone el hardware [8].

Debido al costo en recursos de tiempo y espacio que consume la implementación de un multiplicador en lógica programable, éste se convierte en la parte esencial de la implementación de una NN. Para el trabajo se tuvieron en cuenta diferentes estructuras y arquitecturas de multiplicadores, las cuales fueron comparadas con base en la información obtenida a partir de la herramienta MAX+PLUS II [11]. Este programa permite obtener la información de espacio utilizado por una implementación y su frecuencia máxima de trabajo.

Diagrama 3. Modelo estructural de una neurona



Puesto que se buscaba una implementación de bajo costo de recursos en el FPGA, se optó por una implementación serial por paralelo, utilizando una variante del algoritmo de Booth [3]. En la siguiente tabla se puede observar diferentes implementaciones de multiplicadores, en los que la implementación escogida sobresale en casi todas las características evaluadas.

Tabla 1. Comparación de diferentes implementaciones de multiplicadores

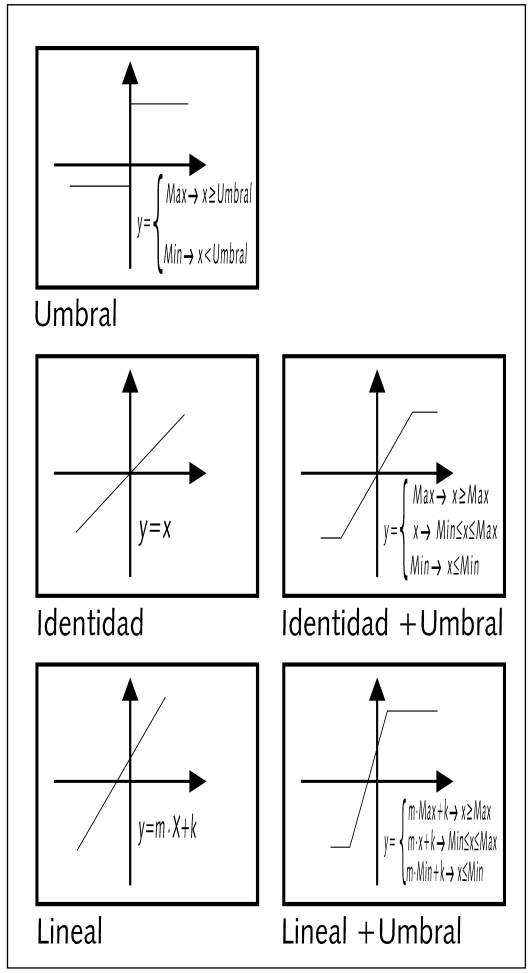
Implementación (16x16 Signed Multiplier)	LCs (Logic Cells)	DFFs (Flip–Flops)	Frec. Max. Reloj (MHz)	Entrega de Respuesta (MHz)
A*B (IEEE VHDL' 93)	417	0	22.7	22.7
Parallel by serial	84	50	20.20	1.3 (20.20/16)
Serial by parallel	47	32	125	3.9 (125/32)
Lpm_mult (4 stage pipe)	417	308	75.75	18.9 (75.75 / 4)

Se implementaron las siguientes funciones de activación (ver gráfica 1):

1. Umbral o *Threshold*.
2. Identidad.
3. Lineal.
4. Identidad con umbral.
5. Lineal con umbral.

Todos los parámetros de las funciones pueden ser personalizados: niveles de umbral, pendiente y corte con el eje *y*.

Gráfica 1. Funciones de activación implementadas

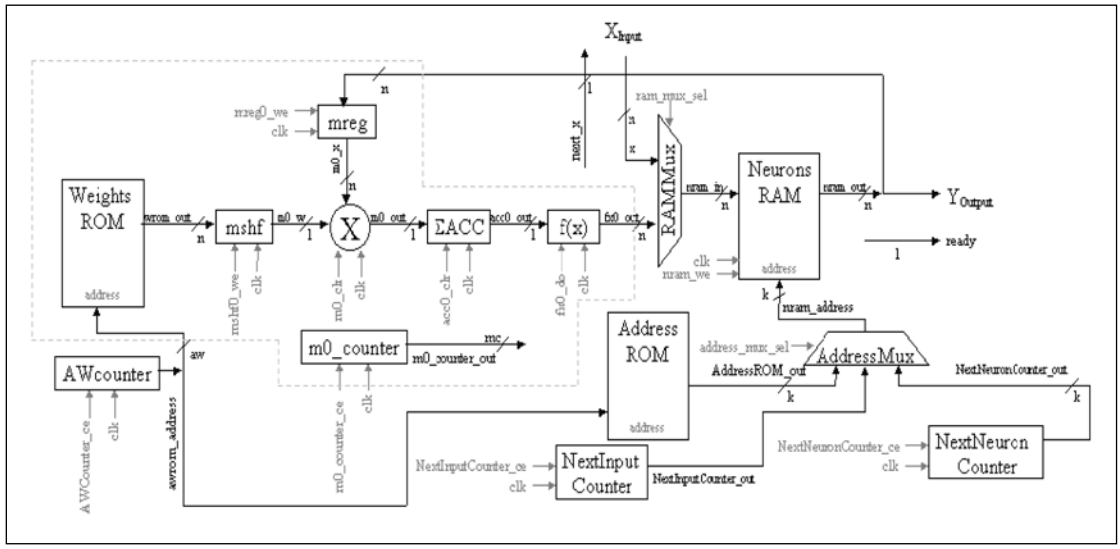


3. Implementación del sistema

A partir de la estructura de una neurona, se diseña el sistema completo mediante la adición de una memoria RAM que almacena el resultado del cómputo de una neurona y las entradas *x* a la red y una memoria ROM para almacenar la topología de la red.

La gráfica 2 muestra el sistema completo y la gráfica 3 muestra el algoritmo de la implementación.

Gráfica 2. Estructura para la implementación de una red neuronal



Gráfica 3. Algoritmo de la Implementación

Si $(AWCounter = SiguienteNeurona)$
$\Rightarrow \sum Acc = 0$

Si (WLC) --se finalizo una multiplicación
$\Rightarrow op1 = NeuronsRAM[AddressROM]$
$op2 = WeightsROM[AWCounter]$
$\sum Acc = \sum Acc + op1 \cdot op2$
$AWCounter = AWcounter + 1$

Si $(AWCounter = NeuronaAnterior + TotalPesosNeurona + 1)$
$\Rightarrow f(x) = \sum Acc$ -- realizar función de activación

Si $(AWCounter = NeuronaAnterior + TotalPesosNeurona + 2)$
$\Rightarrow NeuronsRAM[NextNeuronCounter] = Yreg$
$Nram\_we = Enable$
$NextNeuronCounter = NextNeuronCounter + 1$

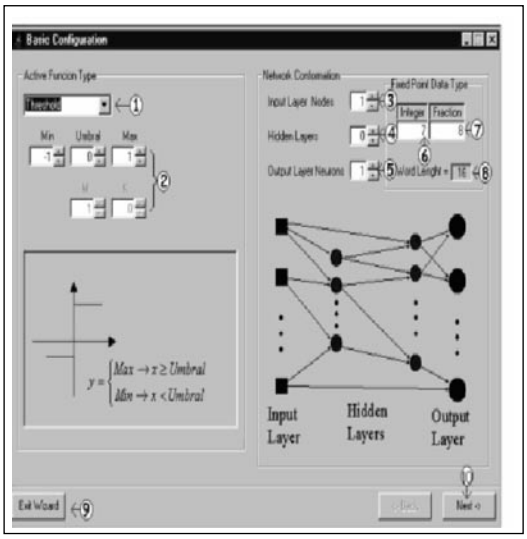
Mientras (no hay resultados de neurona)
$\Rightarrow NeuronsRAM[(NextInputCounter)] = X_{Input}$
$NextInputCounter = NextInputCounter + 1$

4. Software para ingresar el diseño de la red

El software para el diseño de la red por parte del usuario tiene como fin principal crear un archivo tipo VHDL que cumpla con las características de los lenguajes para descripción de hardware, para así poder implementar a través de éste la red neuronal en un FPGA.

Se le da al usuario la mayor libertad posible para generar la red que más se acomode a sus necesidades; es decir, el usuario puede crear la topología (ver figura 1), escoger entre diferentes funciones de activación, determinar el tamaño de palabra de los datos y, en el momento de generar el archivo en VHDL, poder escoger la ubicación de las memorias para almacenar los datos (externas o internas a la arquitectura), además permite realizar la simulación de la red implementada en el PC y monitorear su funcionamiento directamente en el FPGA.

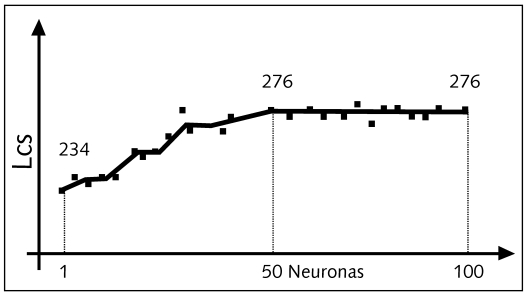
Figura 1. Pantalla de ejemplo de la aplicación



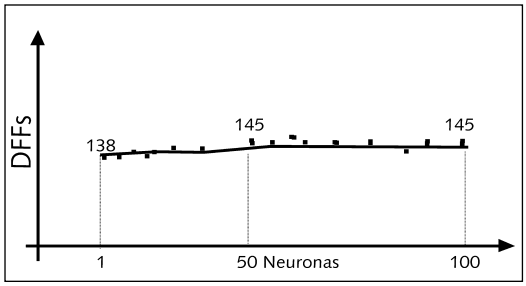
5. Análisis de resultados

Se probó la red ingresando topologías arbitrarias con distintas cantidades de entradas, salidas, capas ocultas, etc. Se crearon redes para resolver problemas básicos como la función OR, la función XOR y la función Suma aritmética. Durante el desarrollo de las pruebas se logró comprobar que es posible realizar la implementación de una red neuronal de grandes dimensiones sin que haya un incremento significativo de recursos.

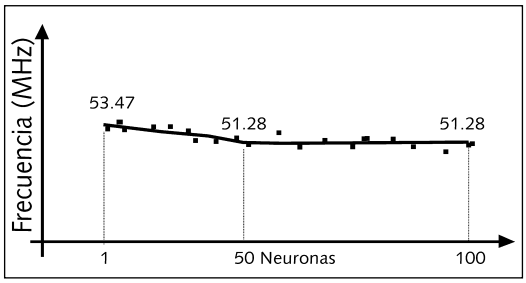
Gráfica 4. Incremento de LCs vs Neuronas



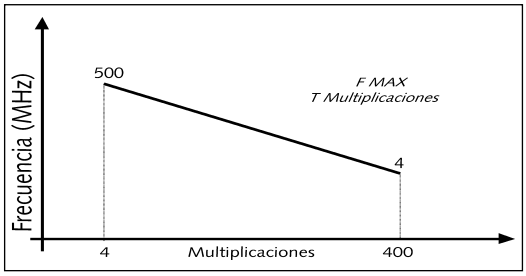
Gráfica 5. DFFs vs Neuronas



Gráfica 6. Decremento de la frecuencia máxima



Gráfica 7. Decremento de la velocidad de cómputo de una red neuronal



En las gráficas 4 a 7 se puede observar las variaciones de los recursos y las velocidades de cómputo respecto a la variación del tamaño de la red implementada.

6. Conclusiones

Es importante, al implementar redes neuronales en hardware, tener un balance

adecuado entre el uso de recursos y la velocidad de procesamiento, de tal manera que sea posible implementar una gran variedad de redes neuronales en el dispositivo destino. En este caso, con un FPGA de mediana complejidad como el FLEX10K10 se pudo implementar diversas topologías de redes neuronales. Esto posiblemente no se hubiera logrado de haber optado por otro tipo de implementación, basado, por ejemplo, en sistemas paralelos, lo cual hubiese sido más rápido, pero con mayor costo de recursos.

La implementación de sistemas digitales basados en procesamiento serial presenta la ventaja de disminuir al máximo la cantidad de recursos necesarios para su implementación a costa de disminuir la velocidad de procesamiento global; esto, en ciertos casos, puede ser compensado con la replicación de algunas partes constituyentes del sistema. Este tipo de sistemas puede ser ventajoso en FPGA de poca capacidad, cuando los recursos son escasos o en casos en los que se requiera un ingreso de datos rápido para grandes frecuencias, como por ejemplo los sistemas de comunicación.

La forma de representar los números y de hacer operaciones con ellos, cuando se trata de números reales, es fundamental para el éxito de un diseño en FPGA. La opción de utilizar el formato punto fijo en sistemas digitales implementados en FPGA es la más adecuada, ya que permite realizar implementaciones más rápidas con un bajo costo en espacio.

Las funciones de activación en una red neuronal constituyen una base fundamental en la calidad de respuesta que se pueda esperar de la red. En ciertos casos, un análisis adecuado mediante varias pruebas puede permitir disminuir los recursos necesarios para implementar una red neuronal. Por ejemplo, si bien es cierto que la respuesta de una función lineal puede ser mejor que la respuesta de una función umbral, para algunos casos, el costo de la segunda en rapidez y espacio es mayor que la primera, influyendo significativamente en las características con las que se requieran los resultados.

Es importante, al implementar redes neuronales en hardware, tener un balance adecuado entre el uso de recursos y la velocidad de procesamiento, de tal manera que sea posible implementar una gran variedad de redes neuronales en el dispositivo destino.

Referencias bibliográficas

[1] Arias, E. y Torres, H. (Febrero 2000). *Sistemas inteligentes en un chip utilizando FPGAs: Aplicaciones a la visión por computadora.* Cholula, México: X Congreso Internacional de Electrónica, Comunicaciones y Computadoras, pp. 37-41.

[2] Baratta, Bo, Caviglia, Diotalevi y Valle. (1998). *Microelectronic Implementation of ANN. 5th Electronic Devices and Systems Intl Conf.*. Brno, Rep. Checa.

[3] Efthymiou, W. (2004). *An Asynchronous, Iterative Implementation of the Original Booth Multiplication Algorithm*. 10th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'04) pp. 207-215

[4] Fontalba, González y Sandoval. (1996). *Realización de una Red Neuronal en una FPGA para Detección de Velocidad*. España: XI Congreso de Diseño de Circuitos Integrados, Sitges.

[5] Fu, L. (1994). *Neural Networks in Computer Intelligence*. S.d. McGraw-Hill.

[6] Haykin, S. (1992). *Neural Networks, a Comprehensive Foundation.* S.d.: Macmillan College Publishing Company.

[7] Lenne, P. (1997). *Digital Connectionist Hardware: Current Problems and Future Challenges.* Lanzarote, España: International Work-Conference on Artificial and Natural Neural Networks, IWANN'97, pp. 688-713.

[8] Labrose, J. (1998). "Fixed Point Arithmetic for Embedded Systems". *C/C++ Users Journal: Advanced Solutions for C/C++ Programmers*, Vol. 16, No 2. p.21-28.

[9] Pérez-Uribe, A. (1999). FAST: *A Neural Network with Flexible Adaptable-Size Topology.* Extraído de la World Wide Web: http://lslwww.epfl.ch/pages/tutorials/dnn_adapt

[10] Tosini, Acosta y Boemo. (2001). *NNGgen: un sistema de generación automática de redes neuronales digitales para FPGA.* Montevideo, Uruguay: VII Workshop IBERCHIP.

Infografía

[11] Altera Corporation. Extríado del World Wide Web: http://www.altera.com