DISEÑO E IMPLEMENTACIÓN DE UNA RED NEURONAL EN UN FPGA PARA RECUPERACIÓN DE PATRONES

DESIGN AND IMPLEMENTATION OF A NEURAL NETWORK ON A FPGA FOR PATTERN RECOVERY

Aurelio Morales Villanueva[1], César Briceño Aranda[2]

RESUMEN

El presente trabajo de investigación muestra el diseño e implementación en hardware basado en dispositivos de lógica programable de alta complejidad como son los FPGAs, de una red neuronal artificial de Hopfield para la recuperación de patrones almacenados, a partir de patrones difusos. Se diseñaron dos redes neuronales artificiales (básica y avanzada), haciéndose uso de herramientas de diseño electrónico EDA y lenguaje de descripción de hardware.

Palabras clave.- Arreglo de compuertas programables en campo, red neuronal de Hopfield, Diseño electrónico automatizado, Lenguaje de descripción de hardware, Max+Plus II, Quartus II.

ABSTRACT

The present research shows the design and implementation of a Hopfield Artificial Neural Network (ANN) for pattern recovery, starting from noisy patterns, based on high complexity programmable logic devices like FPGAs. Two artificial neural networks were designed – one basic, and the advanced one – making use of EDA software tools and Hardware Description Language (HDL) .

Key words.- Field programmable gate array, Hopfield neural network, Electronic design automation, Hardware description language, Max+Plus II, Quartus II.

INTRODUCCIÓN

El problema planteado era la búsqueda de los algoritmos para el aprendizaje y el reconocimiento de patrones a partir de una base de conocimiento, usando técnicas de redes neuronales, y la implementación a nivel de prototipo de dichos algoritmos usando dispositivos de lógica programable. En el presente proyecto se diseñó la arquitectura de una red neuronal artificial digital que era capaz de reconocer patrones digitales de una base de datos de conocimiento, a partir de patrones difusos. En este caso se hizo un énfasis en la solución de los algoritmos de entrenamiento y recuperación de patrones, usando una solución basada estrictamente en hardware, utilizando dispositivos lógicos programables del tipo Field Programmable Gate Arrays (FPGA). Para el presente proyecto se tomó como problema, el reconocimiento de patrones digitales que previamente se almacenaron en la memoria de la red neuronal artificial. Estos patrones estaban formados por arreglos de píxeles blanco y negro que representan números.

FUNDAMENTO TEÓRICO

Es ampliamente conocido el auge que se ha tenido en las últimas décadas el desarrollo del ambiente de la computación, y en específico de las computadoras. Cada vez son más potentes, y capaces de realizar cálculos complejos en menos tiempo. Pero, a pesar del gran avance en la capacidad de cómputo de las computadoras y en

[1] Ing. MSc. Docente investigador de la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Ingeniería, [2]Ing. Electrónico, Docente investigador de la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Ingeniería.

especial de los microprocesadores, aun se necesita un gran esfuerzo computacional para resolver tareas como el reconocimiento de patrones, las que involucra también mucho tiempo de procesamiento, y que puede hacer inviable la solución a un determinado problema que debe ser resuelto en tiempo real. Esto se debe a que la información que viene del mundo real es masiva, en paralelo, muchas veces redundante, y otras veces imprecisa, con ruido, cuando las computadoras trabajan sobre datos precisos y de manera secuencial.

Entonces, una posible solución al problema es recurrir al procesamiento paralelo, empleando para ello muchos CPUs potentes, a fin de resolver los problemas mencionados. Pero el cerebro no trabaja en base a muchos CPUs potentes y tampoco se puede considerar una arquitectura del tipo Von Neumann; muy por el contrario, el procesamiento del cerebro es realizado por millones de procesadores elementales conocidos como neuronas, las cuales se interconectan conformando una red muy compleja, capaz de procesar una gran cantidad de información en paralelo.

Es aquí donde aparece el campo de investigación de las redes neuronales artificiales, las cuales, tratan de emular algunas de las características de procesamiento del cerebro. En el presente proyecto se diseñó la arquitectura de una red neuronal artificial digital que es capaz de reconocer patrones digitales de una base de datos de conocimiento, a partir de patrones difusos. En este caso se hizo un énfasis en la solución de los algoritmos de entrenamiento y recuperación de patrones, usando una solución basada estrictamente en hardware, utilizando dispositivos lógicos programables.

Red neuronal biológica

El elemento básico del sistema nervioso es conocido como la neurona. Solamente en la corteza humana existe entre 13 a 15 billones de neuronas, las cuales están organizadas en una estructura jerárquica de capas. La estructura de una neurona se muestra en la Fig. 1. La neurona está construida para la transmisión de señales en una determinada forma, excepto por su cuerpo, conocido como soma, que es el órgano de procesamiento de la neurona.

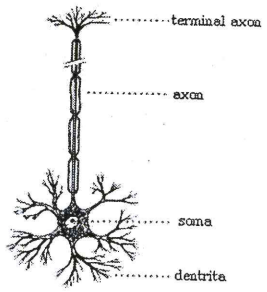

Fig. 1 Neurona biológica.

La neurona tiene los canales para entrada y salida de información, conocidos como dentritas y axón, respectivamente. El axón se ramifica en muchos terminales axón, los cuales, terminan en una membrana para contactarse con otras dentritas, a través de las sinapsis, como se muestra en la Fig. 2.

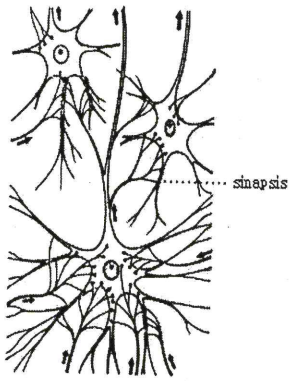

Fig. 2 Red neuronal biológica.

Desde el punto de vista funcional, las sinapsis están clasificadas en dos tipos: sinapsis excitadoras, las cuales habilitan impulsos a ser esparcidos en el sistema nervioso, y las inhibidoras, las cuales causan una atenuación del estímulo.

La transmisión de la información es posible desde que el soma y el axón están cubiertos con una membrana que es capaz de generar impulsos eléctricos bajo ciertas circunstancias. La comunicación entre neuronas es por lo general de manera química, a través de las sinapsis, las cuales, ajustan la intensidad de las señales. Las últimas neuronas en la red recolectan todas las señales de las otras neuronas, determinando si la excitación excede un determinado límite, produciendo un impulso, asegurando la propagación a su destino.

Red neuronal artificial

El modelo de una neurona artificial puede ser obtenido reformulando el funcionamiento de la neuronal biológica. La neurona artificial tiene generalmente n entradas, x_1, x_2, ..., x_n, que serían las señales que vienen de las dentritas. A cada una de estas señales le corresponde un peso w_1, w_2, ..., w_n. De acuerdo a lo mencionado en la sección anterior, cada sinapsis puede ser del tipo excitadora o inhibidora, que se traduce un peso positivo o negativo de la señal de entrada. Ver la Fig. 3.

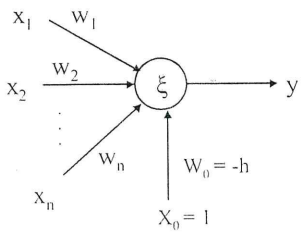

Fig. 3 Neurona artificial básica.

Así, la suma ponderada de las entradas representa el nivel de excitación de la neurona:

$$\xi = \sum_{i=1}^{n} w_i x_i \qquad (1)$$

El valor del nivel de excitación ξ después de alcanzar el nivel de umbral **h** (threshold) induce a una salida **y** de la neurona, que representa el impulso eléctrico generado por el axón. Para una salida no lineal de $\mathbf{y} = \sigma(\xi)$, después que el nivel de umbral es alcanzado es determinado por la función de activación. La función de activación más simple es la de nivel discontinuo o conocida como *hard limited activation function*, como se muestra a continuación:

$$\sigma(\xi) = \begin{cases} 1 \text{ si } \xi \geq h \\ 0 \text{ si } \xi < h \end{cases} \qquad (2)$$

Haciendo una manipulación conveniente, puede obtenerse que la función σ tenga un nivel de umbral **0**, y que el nivel de umbral actual con el signo opuesto sea entendido como un peso adicional, tal que $\mathbf{w_0} = -\mathbf{h}$ y la entrada asociado $\mathbf{x_0} = \mathbf{1}$. De esta forma obtenemos la fórmula matemática para la función de activación como se muestra a continuación:

$$y = \sigma(\xi) = \begin{cases} 1 \text{ si } \xi \geq 0 \\ 0 \text{ si } \xi < 0 \end{cases} \quad \text{donde } \xi = \sum_{i=0}^{n} w_i x_i \qquad (3)$$

Para construir una red neuronal artificial deberemos conectar las neuronas de tal forma que las salidas de las neuronas sean las entradas de otras neuronas, de manera análoga a como los terminales axones de una neurona biológica se conecta vía las conexiones sinápticas con las dentritas de otras neuronas.

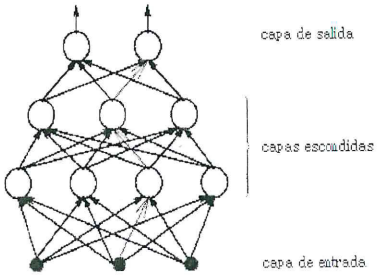

Fig. 4 Red neuronal multicapa.

El número de neuronas y la forma de cómo se interconectan las neuronas determinan la topología de la red neuronal artificial, entonces, se definirán las diferentes capas de la red neuronal: capa de entrada, una o más capas escondidas, y la capa de salida. En cada una de estas capas se tendrán una o más neuronas. Esto se puede apreciar en la Fig. 4.

Para el propósito del proyecto, se implementó una red neuronal artificial del tipo digital, y que trata de emular el accionar de las neuronas del cerebro ante la presencia de patrones difusos, para que ante esta información, sea capaz de recordar los patrones almacenados en su memoria.

Red neuronal de Hopfield

La Red Neuronal Artificial de Hopfield es una de las redes de una sola capa más importantes y ha influido en el desarrollo de una multitud de redes posteriores pudiendo trabajar con varias neuronas permitiendo conexiones recurrentes pero siempre dentro de la misma capa. La Red de Hopfield se conoce como una red auto-asociada, en la cual los valores de los nodos son actualizados de manera iterativa basados en un principio de cómputo local. El nuevo estado de cada nodo, depende únicamente de la sumatoria de sus entradas ponderadas en un tiempo dado.

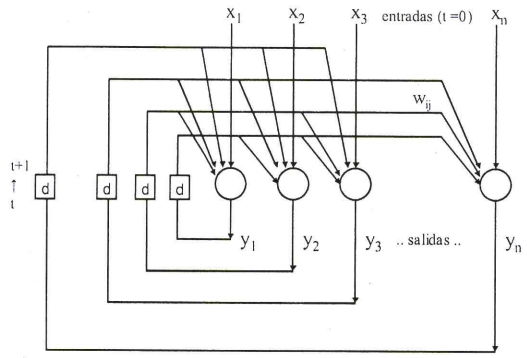

Fig. 5 Topología de la red neuronal de Hopfield.

La topología de la red neuronal de Hopfield se muestra en la Fig. 5. Cabe mencionar que la letra "*d*" representa un retardo (delay) de una iteración. Esto quiere decir que una salida y en el tiempo *t*, se convierte en la entrada *x* en el tiempo *t+1*, y donde w_{ij}, denota el peso de la conexión de la salida de la neurona *j* con la entrada de la neurona *i*. Asimismo, la entrada externa a la neurona *i* se denota como x_i. Notar que la salida de una neurona no se realimenta a sí misma. De esta manera, la suma ponderada net_i de las excitaciones en la neurona *i* puede ser expresada como sigue:

$$net_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} y_i + x_i \quad \text{para } i = 1, 2, \ldots, n \tag{4}$$

De esta manera, podemos expresar lo siguiente:

$$\begin{pmatrix} net_1 \\ net_2 \\ net_3 \\ \cdot \\ \cdot \\ \cdot \\ net_n \end{pmatrix} = \begin{pmatrix} 0 & w_{12} & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & w_{23} & \cdots & w_{2n} \\ w_{31} & w_{32} & 0 & \cdots & w_{3n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ w_{n1} & w_{n2} & w_{n3} & \cdots & 0 \end{pmatrix} * \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \tag{5}$$

es decir:

$$\boldsymbol{net = W * Y + X} \tag{6}$$

donde y_i se obtiene aplicando la función de activación de nivel discontinuo, de forma tal que:

$$y = \begin{cases} 1 \text{ si } net_i \geq 0 \\ 0 \text{ si } net_i < 0 \end{cases} \tag{7}$$

Entrenamiento de la red de Hopfield

El algoritmo de entrenamiento de la red neuronal de Hopfield consiste en el almacenamiento de los pesos w_{ij}. Para esto, asumamos que se desea almacenar p patrones en forma de vectores bipolares, $\mathbf{S^{(m)}}$ donde $m = 1, 2, ..., p$. El algoritmo de almacenamiento de la matriz de pesos W usando la regla de Hebb, se obtiene a partir de la siguiente fórmula [1]:

$$W = \sum_{m=1}^{p} S^{(m)} S^{(m)t} - pI \quad \text{para } m=1,2, \ldots, p \tag{8}$$

o expresado de otra forma:

$$w_{ij} = (1 - \delta_{ij}) \sum_{m=1}^{p} S_i(m) S_j(m) \tag{9}$$

donde δ_{ij} denota la función delta de Kronecker, tal que $\delta_{ij} = 1$ si $i=j$, y $\delta_{ij} = 0$ si $i \neq j$. Notar que i y j varían de $1, 2, ..., n$, y los pesos w_{ii} tienen valor cero.

La información de los vectores unipolares para el algoritmo de almacenamiento de pesos de los vectores $\mathbf{S^{(m)}}$ donde $m = 1, 2, ..., p$, necesitan ser modificados de tal manera que el componente **"-1"** reemplace a los elementos **"0"** en los vectores unipolares originales. Esto puede ser realizado fácilmente reemplazando las entradas del vector unipolar original $\mathbf{S^{(m)}}$ por $\mathbf{2S_i^{(m)} - 1}$ donde $i = 1, 2, ..., n$. De esta forma, podemos expresar los pesos w_{ij} como sigue:

$$w_{ij} = (1 - \delta_{ij}) \sum_{m=1}^{p} (2S_i^{(m)} - 1)(2S_j^{(m)} - 1) \tag{10}$$

Este algoritmo, usando la regla de Hebb, fue usado para el diseño e implementación de la red neuronal básica en un FPGA. Pero, para el desarrollo e implementación de la red neuronal avanzada en un FPGA se usó el algoritmo basado en la matriz pseudo inversa, que fue propuesta para el modelo de Hopfield por Personnaz y otros [2].

Este algoritmo hace el uso del cálculo de la pseudo inversa de una matriz, y es válida para patrones aleatorios y no aleatorios, ortogonales y no ortogonales y permite un almacenamiento de un número de patrones igual al número de neuronas, aunque cada mínimo global casi no se diferencia con el resto. La regla de Hebb es un caso particular

de la regla de la pseudo inversa. Si la matriz de pesos sinápticos de la red de Hopfield es W y $\mathbf{S}^{(m)}$ es uno de los p patrones a memorizar de n neuronas, se debe cumplir la siguiente fórmula [3]:

$$\mathrm{Sgn}\left[W * \mathbf{S}^{(m)}\right] = \mathbf{S}^{(m)} \quad \text{para } m = 1, 2, \ldots, p \qquad (11)$$

trabajando con neuronas bipolares {-1, +1}. Una situación más restrictiva a partir de la condición genérica anterior sería la siguiente:

$$W * \mathbf{S}^{(m)} = \mathbf{S}^{(m)} \qquad (12)$$

Si colocamos los p patrones $\mathbf{S}^{(m)}$ a memorizar como columnas de una matriz G, esta matriz tendrá una dimensión de n x p, por lo que la ecuación (12) se puede re-escribir como sigue:

$$W * G = G \qquad (13)$$

El objetivo es encontrar la matriz W que cumpla con la ecuación (13). Si la matriz G fuera una matriz cuadrada de determinante no nulo, ésta sería invertible, de tal forma que la matriz W se puede expresar como sigue:

$$W = G * G^{-1} \qquad (14)$$

Pero para el caso general, la matriz G no es cuadrada, con lo que la matriz W se obtiene de la siguiente fórmula:

$$W = G * G^{+} \qquad (15)$$

Donde G^{+} es la matriz pseudo inversa de la matriz G, que representa la generalización de la inversión de una matriz cuando esta es rectangular.

Recuperación de patrones en la red de Hopfield

Para red neuronal de Hopfield existen dos algoritmos de recuperación de patrones:

- Actualización dinámica síncrona o paralela
- Actualización dinámica asíncrona

Actualización dinámica síncrona o paralela.- Las salidas de todas la neuronas se actualizan a la vez. Esto puede ser expresado como se muestra a continuación:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}^{k-1} = \sigma \left(\begin{bmatrix} 0 & w_{12} & w_{13} & \cdots & w_{1n} \\ w_{21} & 0 & w_{23} & \cdots & w_{2n} \\ w_{31} & w_{32} & 0 & \cdots & w_{3n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ w_{n1} & w_{n2} & w_{n3} & \cdots & 0 \end{bmatrix} * \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}^{k} + \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \right) \quad \text{para k = 0, 1, ...} \qquad (16)$$

donde los super índices k y $k+1$ denotan las iteraciones en el tiempo. Tomar en cuenta que en $\sigma[]$ se aplica la función **sgn( )** en cada fila escalar de la matriz encerrada en corchetes. Para efectos del caso más general, se asume que existe un vector de entrada X que todo el tiempo se aplica a la entrada de la red neuronal, pero para efectos de recuperación de patrones a partir de uno ruidoso, se dejará de lado el vector X, y por el contrario, el vector con el patrón ruidoso será el correspondiente al vector Y en la iteración 0, es decir Y^0, como se muestra a continuación.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}^{k+1} = \sigma \left(\begin{bmatrix} 0 & w_{12} & w_{13} & \cdots & w_{1n} \\ w_{21} & 0 & w_{23} & \cdots & w_{2n} \\ w_{31} & w_{32} & 0 & \cdots & w_{3n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ w_{n1} & w_{n2} & w_{n3} & \cdots & 0 \end{bmatrix} * \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}^{k} \right) \quad \text{para } k = 0, 1, \ldots \qquad (17)$$

Tomar en cuenta que cada actualización del vector Y define un punto de un hipercubo de 2^n vértices en el espacio de n dimensiones, donde cada vértice representa la combinación binaria de valores (verdad y falso).

Una desventaja de la actualización dinámica síncrona es que podemos encontrar entre dos actualizaciones consecutivas, un patrón y su complemento, llegando a lo que se conoce como dos mínimos locales, pero que no son un mínimo global.

Actualización dinámica asíncrona.- La actualización en cada iteración ocurre en una sola neurona, de tal forma que ahora la fórmula para la actualización se puede expresar como sigue:

$$y_i^{k+1} = \text{sgn}\left(\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} y_j^k\right) \quad \text{para } i = 1, 2, \ldots, n \quad \text{y} \quad k = 0, 1, \ldots \tag{18}$$

donde el índice k denota la iteración en tiempo discreto, y el índice i denota el número de la neurona que se actualiza. De hecho, el índice i debe generarse de manera random. El índice j varía de 1 a n, recordando que cuando $i = j$, w_{ij} vale cero, si se usó la regla de Hebb en el entrenamiento. De esta manera, y tomando en cuenta que el vector inicial es Y^0, las actualizaciones del vector Y serán como sigue:

$$\begin{aligned}
&\text{1ra. Actualización: } Y^1 = [y_1^0\ y_2^0 \ldots y_m^1 \ldots y_p^0 \ldots y_q^0 \ldots y_n^0]^t \\
&\text{2da. Actualización: } Y^2 = [y_1^0\ y_2^0 \ldots y_m^1 \ldots y_p^2 \ldots y_q^0 \ldots y_n^0]^t \\
&\text{3ra. Actualización: } Y^3 = [y_1^0\ y_2^0 \ldots y_m^1 \ldots y_p^2 \ldots y_q^3 \ldots y_n^0]^t
\end{aligned} \tag{19}$$

donde las neuronas m, p y q se escogen de manera random. En el peor de los casos, se tendrán n iteraciones, que es igual al número de neuronas.

Para el desarrollo del proyecto, el algoritmo seleccionado para la recuperación de patrones estuvo basado en la actualización dinámica asíncrona de las neuronas.

DISEÑO E IMPLEMENTACIÓN

Para el propósito del proyecto se diseñaron e implementaron dos tipos de redes neuronales: una red neuronal básica y otra red neuronal avanzada. La red neuronal básica se diseñó e implementó siguiendo la regla de entrenamiento de Hebb, mientras que la red neuronal avanzada se diseñó e implementó usando la regla de la matriz pseudo inversa.

Red neuronal artificial básica.- Para el desarrollo del prototipo de red neuronal artificial básica se utilizó la tarjeta educacional UP1 de Altera [4], que incluye al dispositivo EPF10K20RC240-4, el cual, proporciona 1,344 registros, 12,288 bits en memoria, y 1,152 celdas lógicas.

En las Figs. 6 y 7 se muestran los diagramas de bloques que implementan el algoritmo de hardware para el entrenamiento de la red neuronal artificial básica.

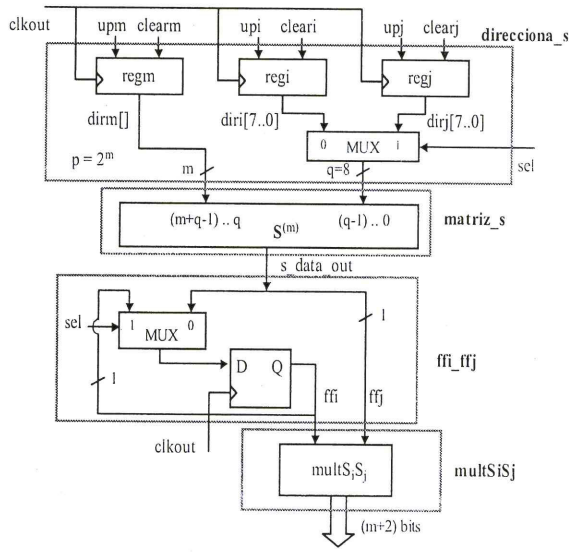

Fig. 6 Diagrama de bloques que implementa el algoritmo de entrenamiento de la red neuronal usando la regla de Hebb (1 de 2).

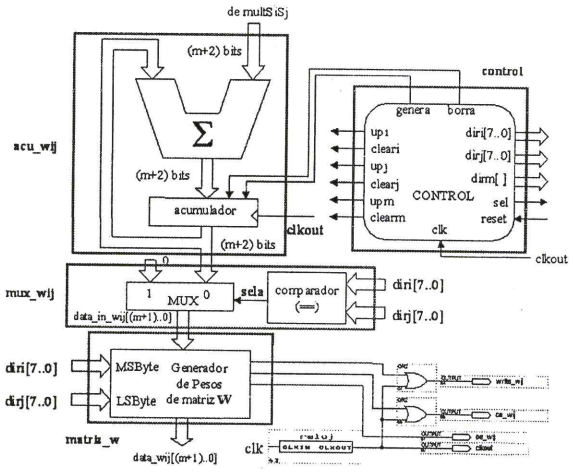

Fig. 7 Diagrama de bloques que implementa el algoritmo de entrenamiento de la red neuronal usando la regla de Hebb (2 de 2).

Para el desarrollo de los archivos de diseño de la red neuronal artificial básica, se usó la herramienta EDA Max+Plus II [5].

El primer diseño implementa el algoritmo en hardware que implementa el entrenamiento de la red, según la regla de Hebb (ver Figs. 6 y 7). El segundo diseño implementa el algoritmo en hardware para la recuperación de patrones, usando

la regla de actualización dinámica asíncrona. Los bloques del segundo diseño se mostrarán para la red neuronal artificial avanzada, desde que a nivel conceptual los bloques son los mismos para la red neuronal básica y avanzada.

Para la implementación de la red neuronal artificial básica, la matriz de conexiones sinápticas ó matriz de pesos W, estuvo compuesta de un elemento de memoria NVRAM externo al dispositivo FPGA. La razón de utilizar una memoria externa al FPGA se debió a que la cantidad de pesos w_{ij} generados era de 256 filas por 256 columnas, es decir, un total de 65,536 conexiones sinápticas, y desde que el dispositivo EPF10K20RC240-4 solo puede almacenar 12,288 bit de memoria, no sería capaz de albergar todos las conexiones sinápticas.

Durante el período de entrenamiento de la red, se generaron los pesos w_{ij} que fueron almacenados en la memoria NVRAM, y durante el período de recuperación de patrones, se usaron los pesos w_{ij} almacenados, a fin de recuperar los patrones originales. Para la implementación de la memoria que almacena los pesos w_{ij} se escogió la memoria del fabricante "Texas Instruments Inc"., modelo BQ4016MC-70 la cual posee una capacidad de almacenamiento de 1M x 8 bits [6]. Las salidas de las neuronas se visualizaron en una pantalla VGA, haciendo uso de la interfaz que proporciona la tarjeta UP1 de Altera.

Red neuronal artificial avanzada

Para el desarrollo del prototipo de red neuronal artificial avanzada se utilizó la tarjeta educacional DE2 de Altera [7 y 8], que incluye al dispositivo Cyclone II EP2C35F672C6, el cual, proporciona 33,088 registros, 483,840 bits de memoria y 33,216 celdas lógicas. En las Figs. 8, 9, y 10 se muestran los bloques que implementan el algoritmo de hardware para la recuperación de patrones a partir de patrones ruidosos. Estos bloques son válidos para la red neuronal avanzada y básica.

Para el desarrollo de los archivos de diseño de la red neuronal artifical avanzada, se usó la herramienta EDA Quartus II de Altera [9 y 10], implementándose el algoritmo en hardware para la recuperación de patrones, usando la regla de actualización dinámica asíncrona.

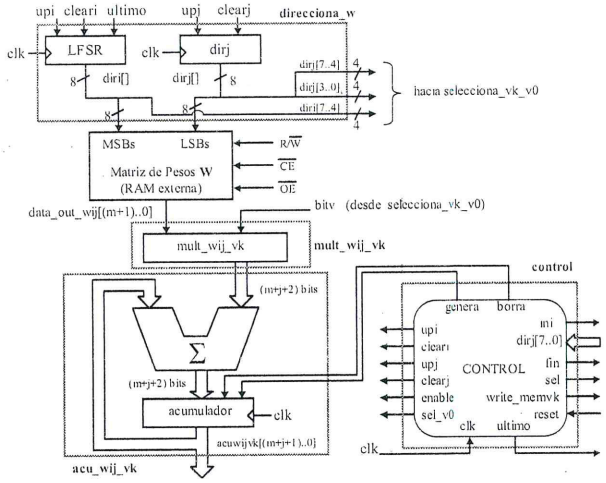

Fig. 8 Diagrama de Bloques que implementa la recuperación de patrones (1 de 3).

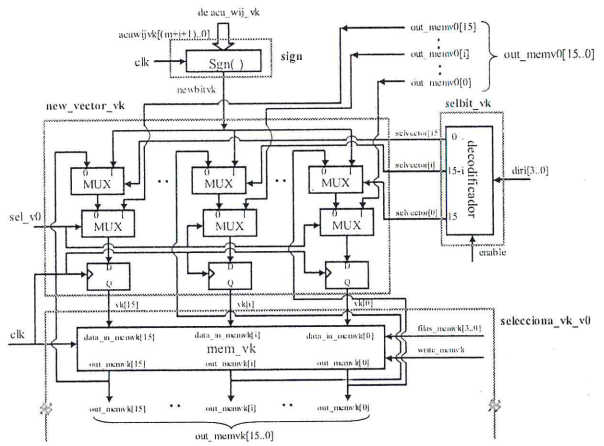

Fig. 9 Diagrama de Bloques que implementa la recuperación de patrones (2 de 3).

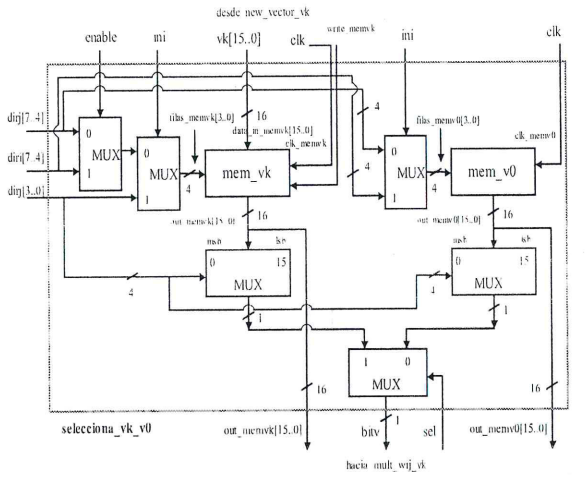

Fig. 10 Diagrama de bloques que implementa la recuperación de patrones (3 de 3).

Para la implementación de la red neuronal artificial avanzada, la matriz de conexiones sinápticas ó matriz de pesos W, estuvo compuesta de un elemento de memoria externo al dispositivo FPGA. Para esto, se utilizó la memoria semiconductora SRAM que posee la tarjeta educacional DE2 de Altera, esto debido a que se generaron 256 filas x 256 columnas de conexiones sinápticas, y cada peso tenía 16 bits de información, con lo que se tendría 65,536 ó 64K palabras de 16 bits, es decir, 1'048,576 bits de RAM, cuando la cantidad máxima de bits de RAM que alberga el dispositivo Cyclone II EP2C35F672C6, es de 483,840 bits de RAM. Las salidas de las neuronas se visualizaron en una pantalla VGA.

RESULTADOS

Para el análisis de los resultados experimentales, el parámetro de estudio fue la energía de la red neuronal, la cual fue observada a lo largo del período de recuperación de patrones ruidosos. La energía de la red neuronal se define por la siguiente expresión [1]:

$$\mathbf{E = - (1/2)\ V^{T} * W * V} \qquad (20)$$

donde V es el vector columna que contiene el valor de todas las neuronas (valores bipolares –1 y +1) y V^T es su transpuesta (vector fila).

La fórmula anterior se puede expresar de otra manera:

$$E = -(1/2)\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} v_i v_j \qquad (21)$$

donde se toma en cuenta que $w_{ij} = 0$ para $i = j$. Esto quiere decir que nos estaremos basando en el estudio original de la red de Hopfield, usando la regla de Hebb, la cual, se basa en que la matriz de conexiones sinápticas es simétrica ($w_{ij} = w_{ji}$, $i \neq j$) y que todos los elementos de la diagonal son ceros.

Esta restricción será dejada de lado cuando se trabaje con la red de Hopfield cuyo aprendizaje está basado en el método de la pseudo inversa de la matriz, pero la fórmula (20) seguirá siendo válida y en la fórmula (21) los índices i y j pueden ser iguales. Los resultados experimentales deben demostrar que la red neuronal implementada ante la presencia de un patrón ruidoso debe evolucionar hacia un mínimo de energía que representa uno de los patrones almacenados, de tal forma que:

$$\mathbf{E((t)) \ ?\ E(V(t+1))} \qquad (22)$$

donde t y $t+1$ son iteraciones en el tiempo. El valor final de energía, al final del período de recuperación, debe corresponder a un mínimo de energía que corresponde a uno de los patrones almacenados en la red.

Resultados para red neuronal artificial básica

En la Fig. 11 se muestra la implementación de la red neuronal artificial básica, y algunas pruebas realizadas.

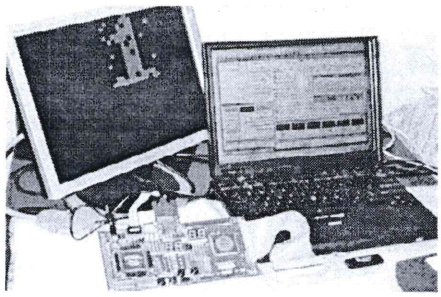

Fig. 11 Recuperación de patrón "1" a partir de un patrón ruidoso con la red neuronal artificial básica.

La actualización de una neurona se realiza en 39 μs, con lo que la actualización de las 256 neuronas se realiza en un tiempo de 9.984 ms y su efecto no se notaría en la pantalla. Para esto, se modificó el diseño, a fin de actualizar el total de las neuronas en un tiempo de 15 segundos, con lo que la actualización de cada neurona se visualiza cada 58.6 ms.

Para una perfecta recuperación de patrones ruidosos aplicando la regla de Hebb, los patrones a ser almacenados en la red deben ser ortonormales, donde la cantidad de patrones a almacenar p, es menor o igual que la cantidad de neuronas, es decir, $p \leq N$. Es decir, los vectores a almacenar en la red deben ser mutuamente ortogonales, es decir, que $V^T_r * V_s = 0$ para $r \neq s$. Bajo esta condición cada patrón almacenado representa un mínimo global absoluto, y la red converge a uno de los patrones ante la presencia de ruido. Para el

proyecto, los patrones no son mutuamente ortogonales, y no representan mínimos globales, pero sí mínimos locales y la recuperación de patrones estará muy condicionada a la cantidad de ruido a que se exponga inicialmente a la red.

Para la red neuronal básica se realizaron 2 pruebas en función de la cantidad de patrones almacenados. Para la primera prueba los patrones almacenados en la red son cuatro, los cuales se muestran en la Fig. 12.

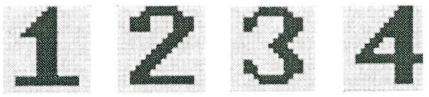

Fig. 12 *Patrones para la primera prueba en la red neuronal básica.*

La Fig. 13, 14, 15 y 16 muestran los resultados obtenidos, donde el nivel de ruido generado es random.

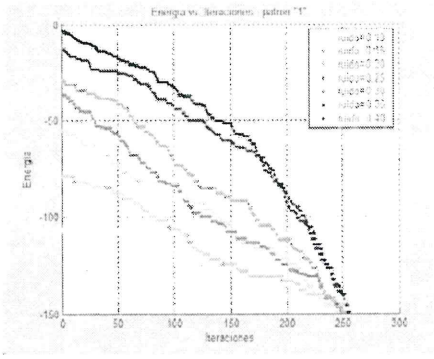

Fig. 13 *Energía vs. iteraciones para recuperación del patrón "1".*

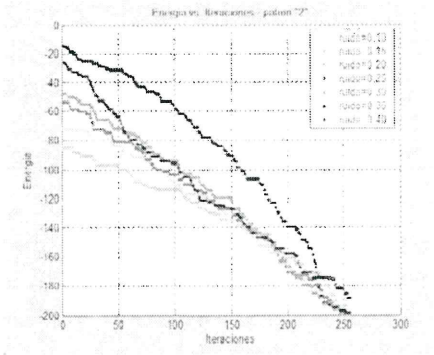

Fig. 14 *Energía vs. iteraciones para recuperación del patrón "2".*

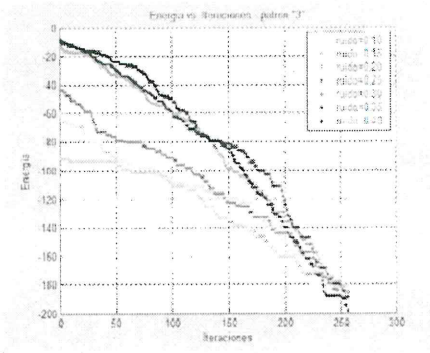

Fig. 15 *Energía vs. iteraciones para recuperación del patrón "3".*

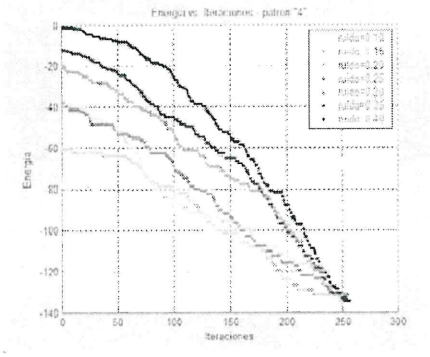

Fig. 16 *Energía vs. iteraciones para recuperación del patrón "4".*

Resultados para red neuronal artificial avanzada

Para el caso de la red neuronal avanzada la matriz de pesos W tiene una diagonal cuyos valores w_{ij} $(i=j)$ son diferentes de cero, pero sigue siendo simétrica, es decir, $w_{ij} = w_{ji}$, $i \neq j$. Los resultados mostraron que el desempeño es mejor que la red neuronal basada en la regla de Hebb. En este caso, los patrones almacenados se muestran en la Fig. 17.

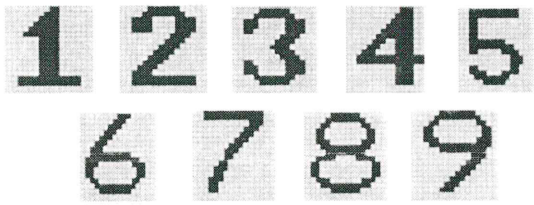

Fig. 17 *Patrones almacenados la en red neuronal avanzada.*

Las Figs. 18, 19, 20, 21, 22, 23, 24, 25 y 26 muestran los resultados obtenidos.

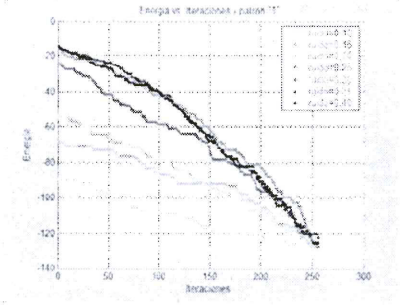

Fig. 18 *Energía vs. iteraciones para recuperación del patrón "1".*

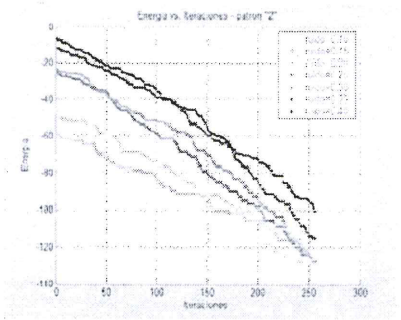

Fig. 19 *Energía vs. iteraciones para recuperación del patrón "2".*

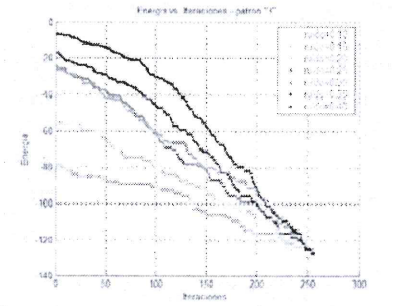

Fig. 20 *Energía vs. iteraciones para recuperación del patrón "3".*

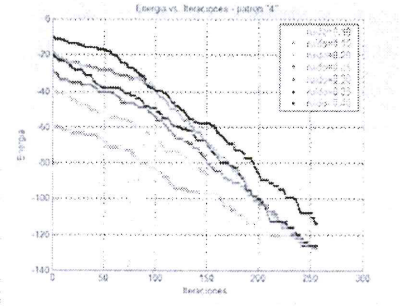

Fig. 21 *Energía vs. iteraciones para recuperación del patrón "4".*

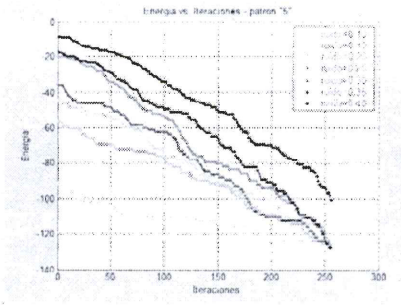

Fig. 22 *Energía vs. iteraciones para recuperación del patrón "5".*

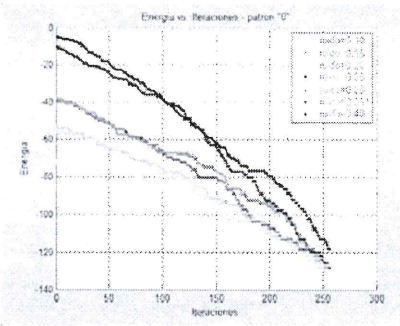

Fig. 23 *Energía vs. iteraciones para recuperación del patrón "6".*

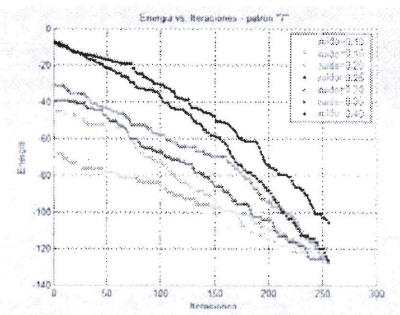

Fig. 24 *Energía vs. iteraciones para recuperación del patrón "7".*

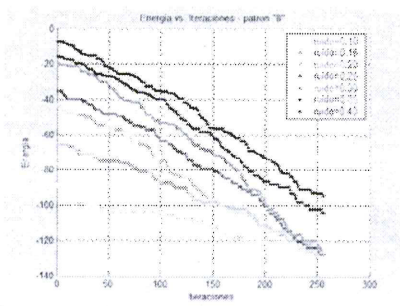

Fig. 25 *Energía vs. iteraciones para recuperación del patrón "8"*

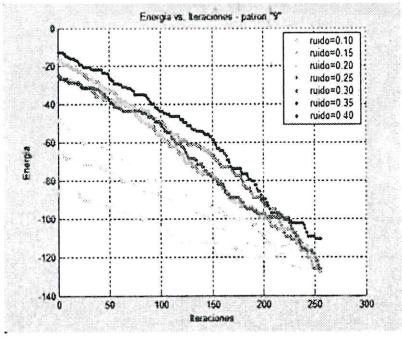

Fig. 26 Energía vs. iteraciones para recuperación del patrón "9".

La actualización de una neurona se realiza en 12.9μs con lo que la actualización de las 256 neuronas se realiza en un tiempo de 3.3024 ms y su efecto no se notaría en la pantalla. Para esto, se modificó el diseño, a fin de actualizar el total de las neuronas en un tiempo de 15 segundos, con lo que la actualización de cada neurona se visualiza cada 58.6 ms. El nivel de ruido generado es random. Por otro lado, tenemos que la energía de todos los patrones sin ruido es el mismo, con un valor de –128.00.

La Fig. 27 muestra la implementación de la red neuronal artificial avanzada sobre la tarjeta DE2 de Altera.

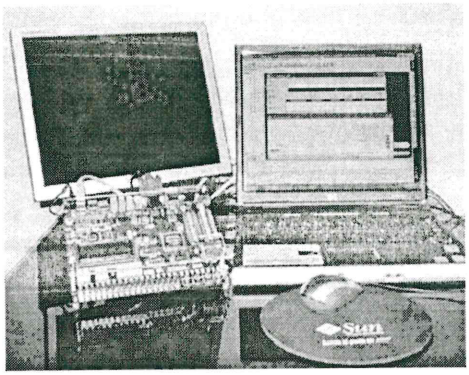

Fig. 27 Recuperación de patrón "1" con ruido en la red neuronal artificial avanzada.

CONCLUSIONES

En la red neuronal de Hopfield, la cantidad de patrones que pueden recuperarse aceptando una cantidad de error está limitada a un 13.8% de la cantidad de neuronas **N**, es decir, **0.138N** según Hertz y otros [11]. Sin embargo, para una mejor recuperación según McEliece y otros [12], el número de patrones a almacenar debe ser menor que **N/(4lnN)**. En base a esto, el límite inferior de patrones a almacenar en la red, tomando en cuenta que N = 256, sería 11.54, que para efectos de un número entero de patrones debe ser 11, y el límite superior sería 35.328 o 35 patrones. Lo anterior es válido cuando los patrones son mutuamente ortogonales. Como los patrones no son mutuamente ortogonales, se encuentra lo siguiente:

a) La cantidad de patrones que se pueden almacenar son menores que lo indicado anteriormente.
b) Para el caso de la red neuronal básica, la cual usó la regla de Hebb, la energía de cada patrón almacenado no es la misma, y tampoco no son los mínimos globales.
c) Más aun, aumentando de 4 a 5 patrones almacenados en la red neuronal básica, empeora el desempeño de la red, ya que ahora los patrones "2", "3" y "5" tienen bastante parecidos.
d) Se comprobó que la red de Hopfield es una red estable, y que siempre trata de converger a un punto de energía mínima.

Para el caso de la red neuronal avanzada podemos llegar a las siguientes conclusiones:

e) Es posible almacenar una mayor cantidad de patrones que la red neuronal de Hopfield basada en la regla de Hebb.
f) Los valores de energía mínimos globales corresponden a la energía de cada uno de los patrones a ser almacenados en la red, siendo un valor común.
g) Lo mencionado en f) garantiza que en el mejor de los casos se recupere uno de los patrones originales ya que tienen la menor energía de la red (mínimo global).
h) La red neuronal avanzada también es estable y es capaz de soportar mayor nivel de ruido inicial que la red neuronal básica.

Con relación a la implementación de las redes neuronales en FPGA obtenemos las siguientes conclusiones:

i) Fue posible diseñar lógica basada en FPGAs para la implementación de algoritmos cuyas reglas admiten que las señales evolucionen en el tiempo sin una interacción externa, tal como la red

recurrente de Hopfield que implementa una memoria autoasociativa.

j) La mayor limitante en la implementación física de la red de Hopfield está en la matriz de conexiones sinápticas W, como se muestra en la Tabla 1, donde se puede apreciar que la velocidad de crecimiento en la cantidad de direcciones para la matriz W es superior a la velocidad de crecimiento de la cantidad de bits de resolución (pixeles) para representar los patrones.

Tabla 1. Matriz W vs. cantidad de pixeles

Cantidad de pixeles	Direcciones de memoria, Matriz W
128 = 16 x 8	2^{14} = 16K
256 = 16 x 16	2^{16} = 64K
512 = 32 x 16	2^{18} = 256K
1024 = 32 x 32	2^{20} = 1M
2048 = 64 x 32	2^{22} = 4M
4096 = 64 x 64	2^{24} = 16M

k) A pesar de lo indicado anteriormente, la implementación de la red de Hopfield en un FPGA mostró que es posible establecer la recuperación de patrones en un tiempo muy corto, del orden de mseg.

l) Este último representa una ventaja sobre una implementación basada en software y abre un campo de investigación sobre particionamiento de aplicaciones que pueden ser optimizadas en hardware a manera de acelerar procesos complejos, como el de recuperación de patrones.

Para un detalle de los resultados y los programas en lenguaje de descripción de hardware para la implementación de las redes neuronales básica y avanzada, remitirse al informe final del proyecto [13].

REFERENCIAS

1. **Zurada, J. M.,** "Introduction to Artificial Neural Systems", PWS Publishing Company, pp. 325-354, USA, 1992.
2. **Personnaz, L., Guyon, I., Dreyfus, G.,** "Collective computational properties of neural neworks: New learning mechanisms", Physical Review A, 34, 5 pp. 4217-4228, USA, 1986.
3. **Martín del Brío, B., Sanz Molina, A.,** "Redes Neuronales y Sistemas Difusos" 2da Edición, Alfaomega Grupo Editor, pp. 123-142, México, 2002.
4. **Altera Corporation**, "Altera UP1 University Program Design Laboratory Package User Guide", USA, 1997.
5. **Altera Corporation**, "Altera Max+Plus II Getting Started Manual", USA, 1997.
6. **Texas Instruments, Inc.** "BQ4016/BQ4016Y 1024Kx8 Nonvolatile SRAM Data Sheet", USA, 1999.
7. **Altera Corporation**, DE2 Development and Educational Board, USA, 2007: http://university.altera.com/materials/boards/unv-de2-board.html.
8. **Altera Corporation**, DE2 Development and Education Board User Manual, ver 1.4, USA, 2006: http://university.altera.com/materials/boards/DE2_UserManial.pdf
9. **Altera Corporation**, "Quartus II software for education", USA, 2007: http://university.altera.com/materials/software/unv-quartus2.html
10. **Altera Corporation**, "Introduction to the Quartus II Software Manual", USA, 2006.
11. **Hertz, J., Krogh, A., Palmer, R.**, "Introduction to the Theory of Neural Computation" Addison-Wesley, pp. 11-70, USA, 1991.
12. **McEliece, R., Posner, E., Rodemich, E., Venkatesh, S.**, "The Capacity of the Hopfield Associative Memory" IEEE Trans. Information Theory IT-33(4) pp. 461-482, USA, 1987.
13. **Morales, A., Briceño, C.**, "Diseño e implementación de una red neuronal en un FPGA para recuperación de patrones" Informe Final, Instituto de Investigación FIEE-UNI, Lima - Perú, 2008.

Correspondencia: amorales@uni.edu.pe

Recepción de originales: Junio 2007
Aceptación de originales: Setiembre 2007