DISEÑO Y SIMULACIÓN DE UN CIRCUITO INTEGRADO DE UNA RED NEURONAL APLICADO AL CONTROL DE VELOCIDAD DE MOTORES DC

DESIGN AND SIMULATION OF INTEGRATED CIRCUIT OF THE NEURONAL NETWORK APLICATION TO CONTROL OF MOTORS VELOCITY DC

Darío Utrilla Salazar[1]

RESUMEN

El objetivo de este trabajo, es el desarrollo de un dispositivo integrado, que permite controlar la velocidad de un motor de corriente continua (DC), utilizando una red neuronal. En el presente informe se describe un diseño desarrollado para controlar este tipo de motores DC utilizando el método de entrenamiento con algoritmos de retropropagación a través de la planta. Asimismo considerando que se puede entrenar a una red neuronal para controlar un proceso dinámico con parámetros variables. Plasmando en un dispositivo físico de un circuito integrado desarrollado con herramientas de diseño digital avanzado.

Palabras clave.- Redes neuronales, Control inteligente, Diseño digital avanzado.

ABSTRACT

The objective of this work, is the development of an integrated device, that allows to control the speed of a motor of DC (DC), using a neuronal network. In the present report a developed design is described to control this type of motors DC using the method of training with algorithms of retropropagacion through the plant. Also considering that can be trained to a neuronal network to control a dynamic process with variable parameters. Shaping in a physical device of an integrated circuit developed with tools of advanced digital design.

Key words.- Neuronal networks, Intelligent control, Advanced digital design.

INTRODUCCIÓN

En diversas aplicaciones del campo industrial, se requiere el uso del motor DC, debido a la facilidad para controlar su velocidad. En el presente trabajo se realiza el análisis del motor DC y su control de velocidad PWM, para lo cual se aplica el procesamiento con técnicas de las redes neuronales multicapa de alimentación directa, considerando algoritmos de retropropagación, el cual se emplea por la característica estática de las funciones a realizar, complementando el desarrollo de un I.C. en un FPGA utilizando técnicas de diseño digital avanzado.

REDES NEURONALES

La estructura básica de una neurona natural se muestra en la Fig. 1.

[1]Docente investigador de la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Ingeniería.

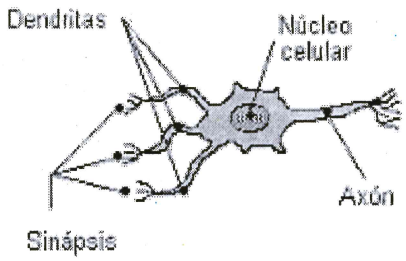


***Fig. 1** Estructura básica de una neurona.*

En el funcionamiento, se observa que cada neurona puede tener infinitas entradas llamadas Dendritas que condicionan el estado de su única salida, el Axón. Este Axón puede ir conectado a una Dendrita de otra neurona mediante la Sinapsis correspondiente, según se muestra en la Fig.2.

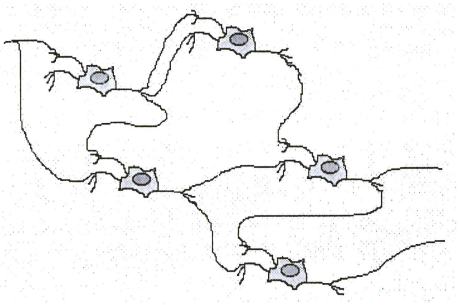


***Fig. 2** Organización de las redes neuronales.*

El Axón entrega un nivel eléctrico correspondiente a sus entradas y según la importancia se asigna a cada una de ellas. De esta forma, una neurona puede no reaccionar ante un nivel muy alto de una de sus entradas, o dar una salida muy favorable cuando otra de ellas está mínimamente activa.

En las primeras etapas de nuestra vida, cuando realizamos el aprendizaje de nuestros cerebros, entrenamos nuestras neuronas mediante el éxito o fracaso de una acción a unos estímulos sensoriales.

Cuando cierta acción realizada en respuesta a alguna entrada sensorial es exitosa (por ejemplo, al beber agua calmamos la sed), las conexiones sinápticas entre un grupo de neuronas se fortalecen, de manera que cuando se tiene una sensación sensorial parecida, la salida será la correcta. De esta forma se forman fuertes conexiones entre grupos de neuronas, que pueden servir para realizar otras acciones complejas. El esquema de una neurona artificial se muestra en la Fig. 3.

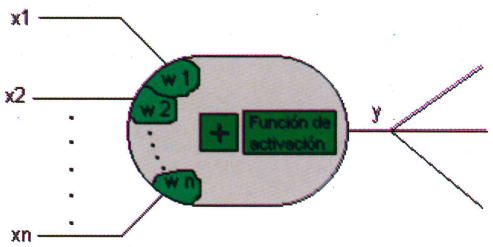
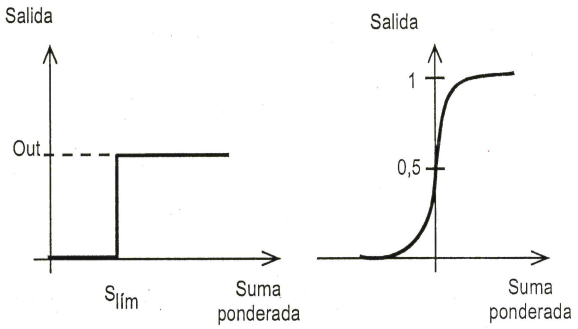


***Fig. 3** Esquema de una red neuronal.*

Esta neurona funciona de la siguiente manera: cada entrada x tiene su peso asociado w, que le dará más o menos importancia en la activación de la neurona. Internamente se calcula la suma de cada entrada multiplicada por su peso:

$$\text{Suma Ponderada} = \sum X_i \cdot W_i$$

Con este valor de suma ponderada se calcula una función de activación, que será la salida que dará la neurona. Las dos funciones de activación más usada son el Escalón y la Sigmoidea:



$\text{Salida} = \begin{cases} \text{Out si Suma} > S_{lim} \\ \text{Ø si Suma} < S_{lim} \end{cases}$	$\text{Salida} = \frac{1}{1+e^{-\text{suma}}}$
Función Escalón	Función Sigmoide

***Fig. 4** Funciones de activación más usadas.*

Principalmente se diferencian en que la Sigmoidea (llamada así por su forma de S) es diferenciable en todos sus puntos y la Escalón no.

EL PERCEPTRÓN UNICAPA

Un Perceptrón unicapa no es más que un conjunto de neuronas no unidas entre sí, de manera que cada una de las entradas del sistema se conectan a cada

neurona, produciendo cada una de ellas su salida individual:

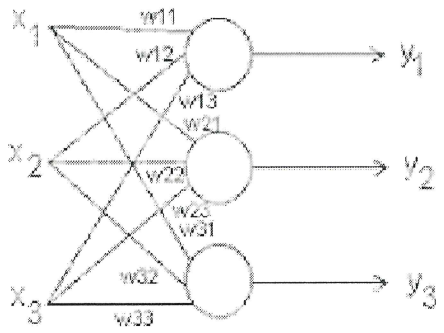


Fig. 5 Esquema del perceptron unicapa.

Como se ha mencionado, un conjunto de neuronas no sirve para nada si previamente no se le enseña qué función debe realizar.

Existen tres métodos de aprendizaje para un Perceptrón:

- Supervisado,
- Por Refuerzo y
- No Supervisado.

Aprendizaje supervisado.- Se presentan al Perceptrón, unas entradas con las correspondientes salidas que se quiere éste aprenda. De esta manera la red primero, calcula la salida que da ella para esas entradas y luego, conociendo el error que está cometiendo, ajusta sus pesos proporcionalmente al error que ha cometido (si la diferencia entre salida calculada y salida deseada es nula, no se varían los pesos).
Aprendizaje no supervisado.- Solo se presentan al Perceptrón las entradas y, para esas entradas, la red debe dar una salida similar.
Aprendizaje por refuerzo.- Se combinan los dos anteriores, y de cuando en cuando se presenta a la red una valoración global de como lo está realizando.

EL PERCEPTRÓN MULTICAPA

Esta estructura nació con la intención de dar solución a las limitaciones del Perceptrón clásico o unicapa, y supuso el resurgimiento del movimiento conexionista. Como su nombre indica, se trata de unos cuantos (dos o tres) perceptrones unicapa conectados en cascada, como en la Fig. 6

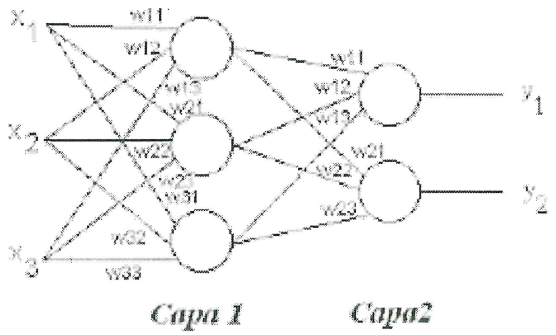


Fig. 6 Esquema del perceptrón multicapa.

El problema de este tipo de Perceptrón está en su entrenamiento, ya que es difícil modificar correctamente los pesos de la capa oculta (la capa 1 en la Fig. 6). Para poder hacer aprender cosas a un Perceptrón de este tipo, se implementó el algoritmo de *BackPropagation*, que tal como su nombre indica tiene la función de ir propagando los errores producidos en la capa de salida hacia atrás.

El proceso de aprendizaje tiene un gran coste de tiempo. Debido a eso, todavía no se ha estudiado a fondo. Las redes neuronales todavía se han de desarrollar mucho. Aún se debe estudiar para qué sirven realmente, conocer en qué tareas pueden resultar realmente útiles, ya que por ejemplo, es difícil saber cuánto tiempo necesita una red para aprender cierta tarea, cuántas neuronas se necesitan como mínimo para realizar cierta tarea, etc.

Las redes neuronales pueden llegar a ser algo realmente importante, pero todavía hace falta tiempo para estudiar cómo almacenan el conocimiento para desarrollar el hardware paralelo específico que requieren.

En la robótica, las redes neuronales también parecen prometer mucho, sobre todo en su sensorización, para que el robot sea capaz de generalizar lo que siente como estímulos individuales a considerar.

Tipos

Para el tratamiento con más detalle de redes neuronales referirse a [1 y 2]. En la identificación de los elementos no lineales del motor se utilizó una red multicapa, con una capa de entrada, una de salida y dos capas ocultas, tal como se muestra en

la Fig.7. Notar que en la salida de cada elemento de una capa se utiliza la función no lineal sigmoidal.

$$g = [(1-e^{-x})/(1+e^{-x})].$$

En la práctica, esta red ha sido utilizada para la identificación de sistemas estáticos.

Los pesos de la red se ajustan con el algoritmo de retropropagación [1, 2 y 3], que consiste en minimizar una adecuada función de error e entre la salida y de la red y una salida deseada y_d.

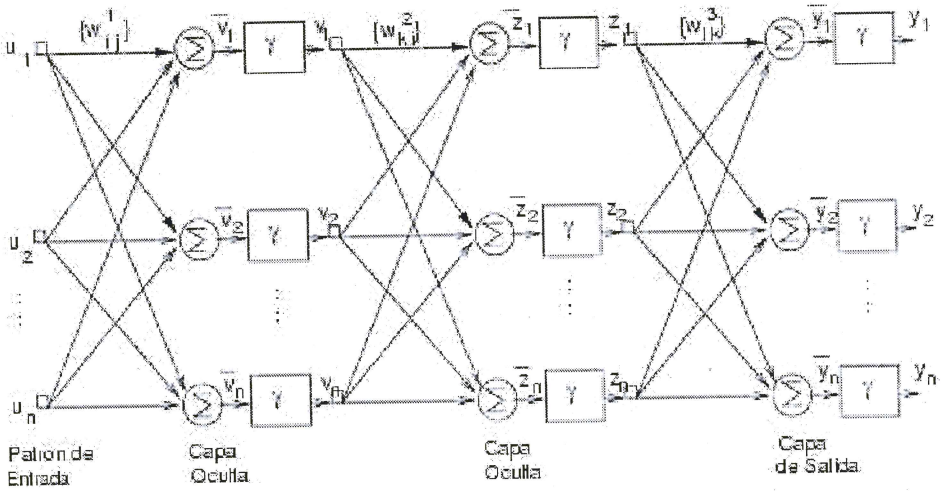


***Fig.** 7 Una red neuronal de tres capas.*

Las rutinas de entrenamiento de la red se programaron en C; las operaciones con matrices se implementaron en base a rutinas optimizadas *Blas* de Octave.

Las redes a utilizar poseen un elemento en la capa de entrada, 20 elementos en la primera capa oculta, 10 elementos en la segunda capa oculta, y, un elemento en la capa de salida.

DESCRIPCIÓN DEL SISTEMA A CONTROLAR

En esta sección se presenta una descripción del sistema implementado, (Fig. 8) y se realiza el modelamiento matemático de la planta.

Para el control, se usó un actuador que consiste de un motor DC de campo magnético permanente y escobillas conmutadas.

Para accionar al actuador se construyó un driver PWM, cuya etapa de potencia consiste de 4 conmutadores en configuración H y una lógica de disparo de tales conmutadores.

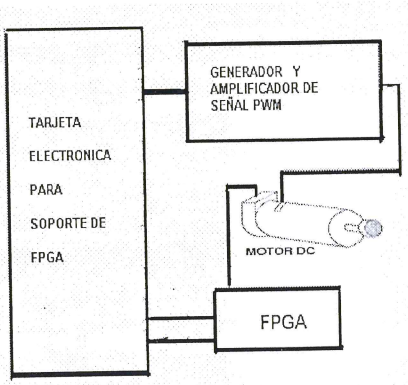


***Fig. 8** Esquema general del sistema implementado.*

Para el sensado de posición y del sentido de giro del motor se cuenta con un codificador óptico montado en el mismo. La información se envía por dos salidas seriales (vía trenes de pulsos desfasados 90 grados) a un FPGA XC3S200E de la familia XILINX, en donde fue integrado un circuito sensor de posición que consiste de un LS7083 y 4 contadores (74193). Estos elementos ocupan un 20% de la capacidad del FPGA [5].

DESARROLLO EXPERIMENTAL

El software para la implementación del circuito de control de motores DC en un FPGA fue el MAX+PLUS II de ALTERA, cuyo procedimiento es el siguiente:

a. Se inicia el programa ingresando al Max + Plus II:

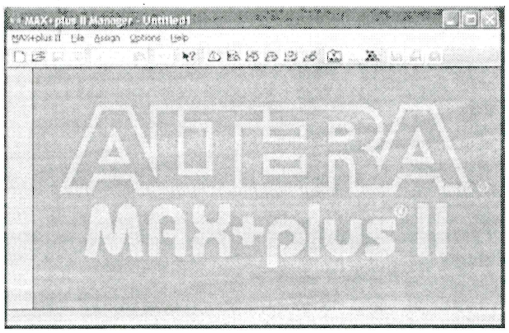


Fig. 9 *Presentación de Max+Plus II.*

b. Seleccionar el comando ***File – New.***

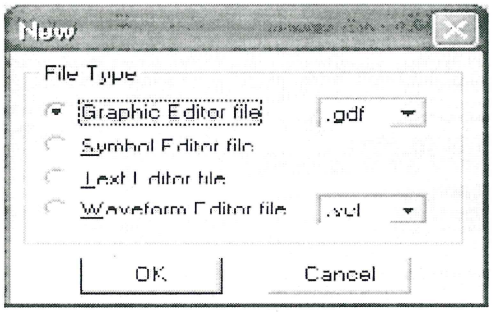
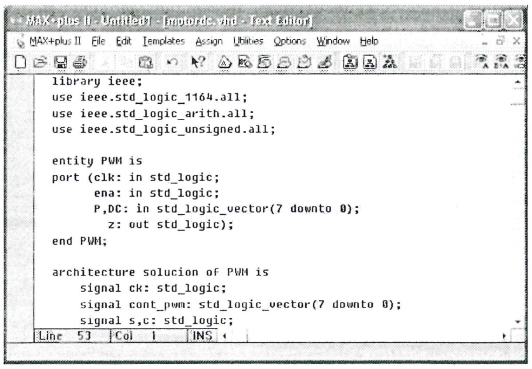


Fig. 10 *Apertura de nuevo proyecto.*

c. Ingresamos el siguiente programa del diseño.



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity PWM is
port (clk: in std_logic;
      ena: in std_logic;
      P,DC: in std_logic_vector(7 downto 0);
        z: out std_logic);
end PWM;

architecture solucion of PWM is
    signal ck: std_logic;
    signal cont_pwm: std_logic_vector(7 downto 0);
    signal s,c: std_logic;
```

Fig. 11 *Ingreso del programa texto.*

d. Guardar el programa asignando como nombre de archivo: ***motordc.vhd.***

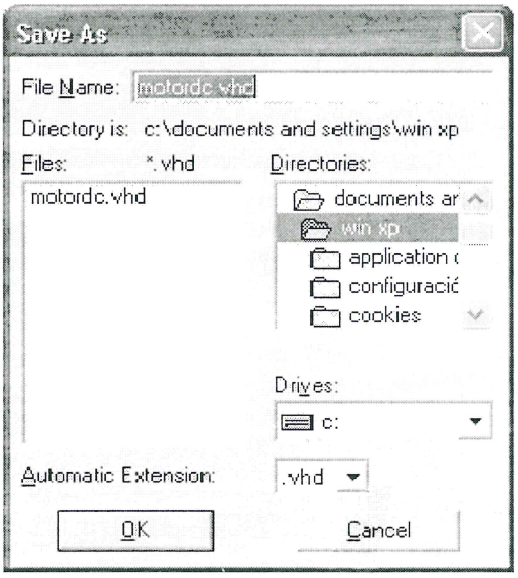


Fig.12 *Almacenamiento del programa.*

e. Para la compilación desprograma se selecciona: ***File-Project-Set Project to Current File***[4].

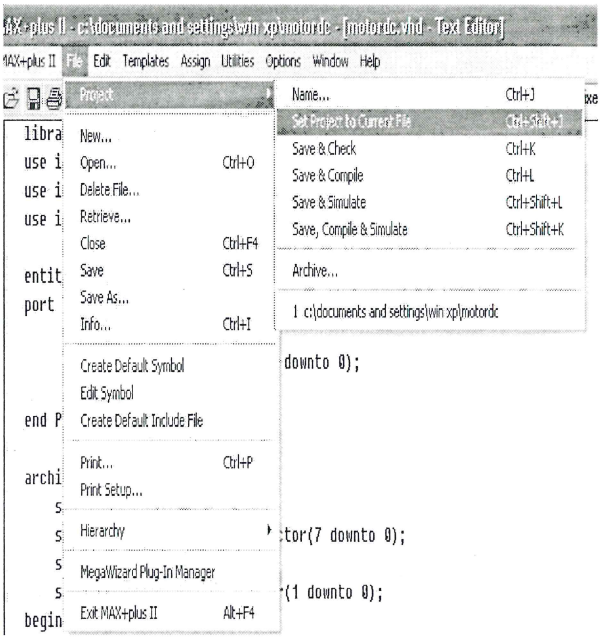


Fig. 13 *Procedimiento de compilación del programa.*

f. Ahora se puede compilar, Seleccionando: ***Max+Plus II – Compiler.***[4]

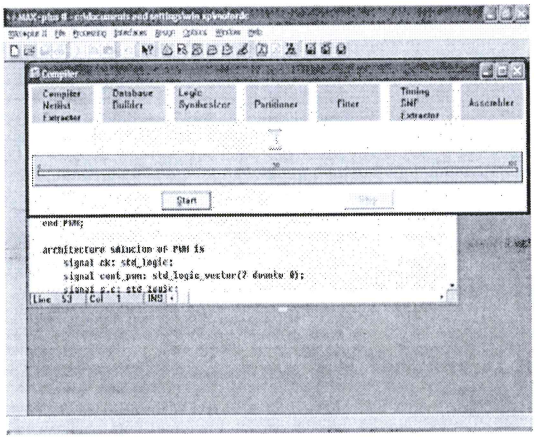


Fig. 14 Compilación del programa.

g. Luego de compilar y depurar errores, se obtiene el diagrama de bloques: seleccionando ***Max+Plus II-Simbol Editor.***

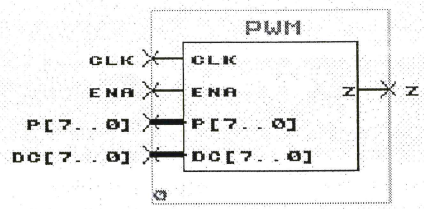


Fig. 15 Diagrama de bloques del diseño.

Donde las entradas son:

- CLK: Señal de reloj que se utiliza para llevar la cuenta del periodo y del ciclo de trabajo.
- ENA: Habilita la salida del generador PWM. Si no está habilitado la salida es cero (0).
- P[7..0]: Define el periodo de la señal PWM, al ser una entrada de 8 bits el periodo puede variar entre 0 a 255.
- DC [7..0]: Define el ciclo de trabajo de la señal PWM, al ser una entrada de 8 bits el periodo puede variar entre 0 a 255.
- Z: Es la salida del generador PWM.

RESULTADOS EXPERIMENTALES

En esta sección se muestran los resultados experimentales obtenidos de la simulación realizados con las herramientas (MAX+PLUS II de ALTERA). En el cual se muestra la respuesta del sistema para los diversos valores de Ciclo de Trabajo (10 los cambios de ciclo de trabajo [4 y 5].

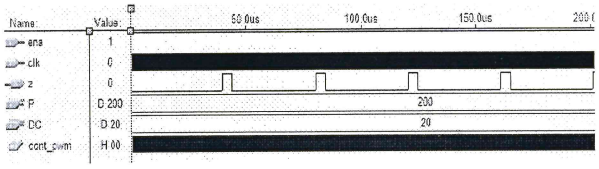


Fig. 16 Simulación de la señal PWM con un ciclo de trabajo del 10%

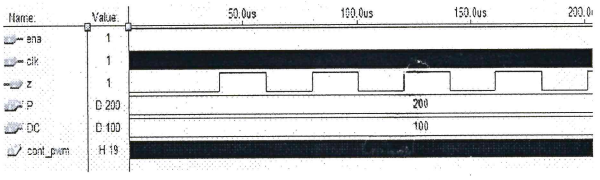


Fig. 17 Simulación de la señal PWM con un ciclo de trabajo del 50%

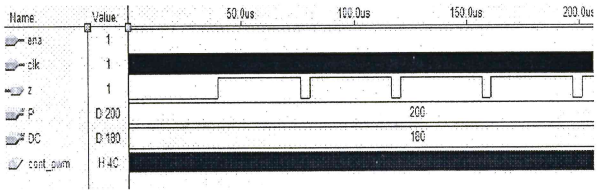


Fig. 18 Simulación de la Señal PWM, con un Ciclo de trabajo de 90 %.

CONCLUSIONES

En este proyecto se ha demostrado que las redes neuronales son capaces de identificar elementos no lineales tales como la fricción Estática y de Coulomb y la carga no lineal de la varilla (componente sinusoidal). También se ha demostrado que las redes neuronales convenientemente entrenadas pueden formar parte de los algoritmos de control, compensando las no linealidades más significativas del motor dentro de su rango de operación. Además con el desarrollo del proyecto basado en tecnología reconfigurable utilizando FPGA y el uso de herramientas de diseño digital avanzado, representados por el entorno EDA (Electronic Design Automation) son las herramientas de trabajo comúnmente empleadas en el diseño e implementación de sistemas digitales; permitiendo una gran

versatilidad y el manejo de interfaces graficas que incorporan las plataformas de diseño: ALTERA, Orcad, XILINX que permiten la instrumentación de dichos diseños y la implantación en los FPGA [5].

REFERENCIAS

1. **Freeman, J. A., Skapura, D. M.,** "Redes Neuronales, Algoritmos, Aplicaciones y Técnicas de Programación", Addison Wesley Iberoamericana, S. A. 2003.
2. **Haykin, S.**, "Neural Networks", Macmillan College Publishing Company. Inc 2004.
3. **Kumpati, S., Parthasarathy, K., Narendra,** "Identification and Control of Dynamical Systems Using Neural" Networks, IEEE Transactions on Neural Networks. Vol. I. No. 1, March 1998.
4. **Maxinez, D. G., Alcala, J.,** "VHDL El Arte de Programar Sistemas Digitales". 2004.
5. **Hamblen, J.O., Furman, M. D.**, "Rapid Prototyping of Digital Systems". Georgia Institute of Technology. 2000.

Correspondencia: dutrilla@uni.edu.pe

Recepción de originales: enero 2007
Aceptación de originales: marzo 2007