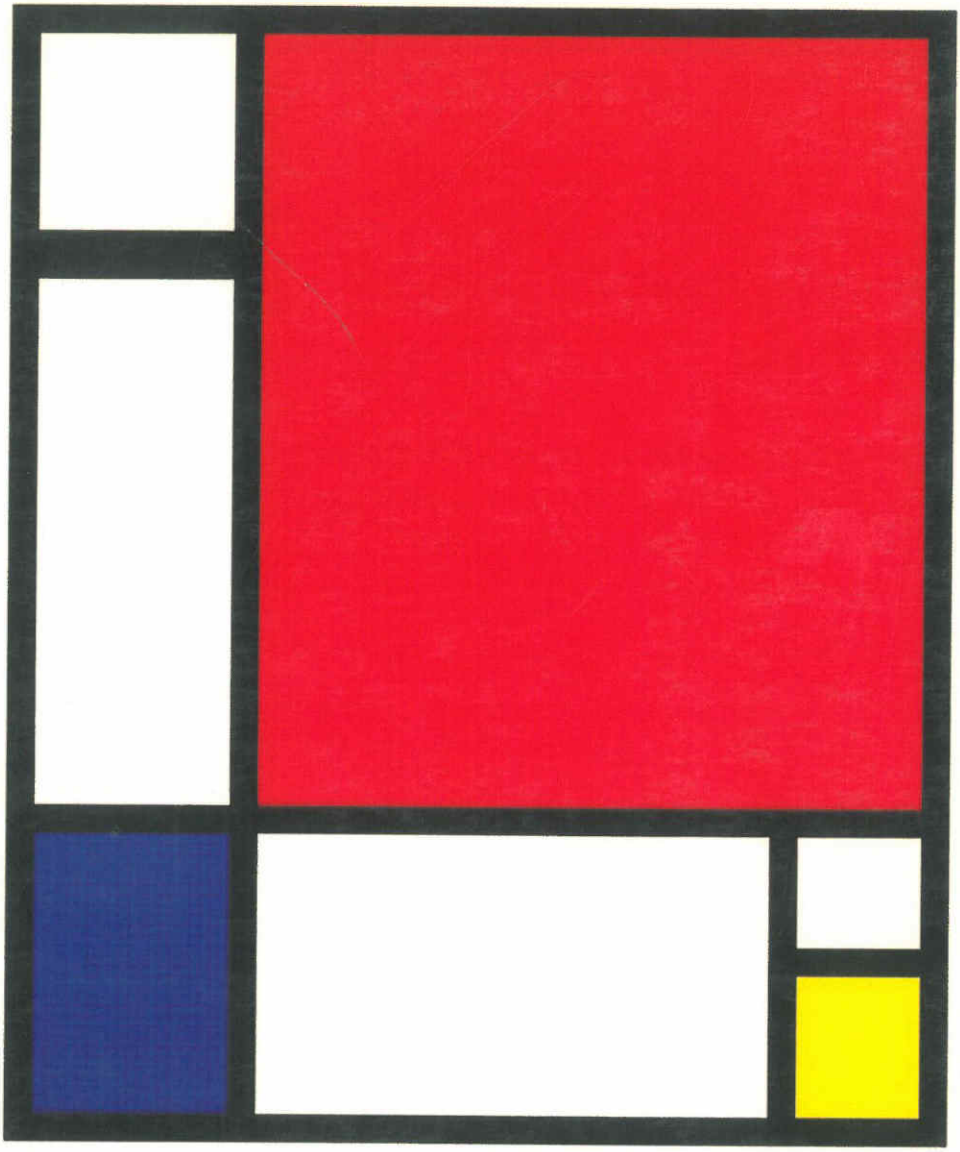


COMPUTACIÓN RECONFIGURABLE & FPGAs

Eduardo Boemo
Francisco Gómez Arribas
Sergio López-Buedo
Gustavo Sutter
(Eds.)

COMPUTACIÓN RECONFIGURABLE & FPGAS

Artículos seleccionados de las III Jornadas de Computación Reconfigurable y Aplicaciones Madrid, 10-12 de Septiembre de 2003

Eduardo Boemo Scalvinoni
Francisco Gómez Arribas
Sergio López-Buedo
Gustavo Sutter Capristo

Editores

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Título: CONFIGURACIÓN RECONFIGURABLE & FPGAs

I.S.B.N.: 84-600-9928-8
Depósito Legal: SE-2914-2003

Editores: E. Boemo, F. Gomez-Arribas, S. López-Buedo, G. Sutter
Escuela Politécnica Superior.
Universidad Autónoma de Madrid.
28049 Cantoblanco
Madrid - España.
www.ii.uam.es

Impresión: Publicaciones Digitales S.A.
C/ San Florencio, 2, 41018, Sevilla - España
Tel: 902 405 500 / (+34) 954 583 425 - Fax: (+34) 954 583 205
info@publidisa.com / www.publidisa.com

Capítulo 6: Redes Neuronales

Capítulo 7: Sistemas de Control

Implementación de Redes Neuronales con FPGAs para el Reconocimiento del Habla

Ortigosa E. M.[1], Ortigosa P. M.[2], Cañas A.[1], Ros E. [1], Carrillo R. R.[1] y Agís R.[1]

[1]Dept. de Arquitectura y Tecnología de Computadores,
ETS Ingeniería Informática. Universidad de Granada, E-18071 Granada, Spain
{eva, acanas, eros, rcarrillo, ragis}@atc.ugr.es
[2] Dept. de Arquitectura de Computadores y Electrónica,
Universidad de Almería, E-04120 Almería, Spain, ortigosa@ual.es

Resumen. En este trabajo se exploran distintas posibilidades de implementaciones de un perceptrón multicapa en FPGAs para el reconocimiento del habla. Los diseños presentados se han definido utilizando dos niveles de abstracción distintos: nivel de transferencia de registros (con VHDL) y un nivel algorítmico de descripción (con Handel-C). Se presenta un estudio de los costos de las diferentes alternativas consideradas en términos de área de silicio, velocidad y recursos computacionales.

1 Introducción

Una Red Neuronal Artificial (RNA) es un sistema de procesamiento de información inspirado en la forma en que los sistemas nerviosos procesan la información. Una RNA se puede configurar para una aplicación específica, como reconocimiento del habla o clasificación de datos, por medio del aprendizaje. Este proceso, como en la biología, modifica los pesos sinápticos entre las unidades computacionales o neuronas.

En este trabajo se realiza un estudio de la viabilidad de implementación y la eficiencia de una RNA implementada con hardware reconfigurable (FPGA) para su utilización en sistema embebidos, como sistemas de reconocimiento del habla portátiles. Entre los distintos tipos de RNA nos hemos centrado en el Perceptrón Multicapa.

Para definir correctamente los parámetros de la red neuronal es necesario concentrarse en un problema concreto, por ejemplo, un sistema de marcado de números de teléfono controlado con la voz (esto puede ser de interés para conductores de automóviles que deben mantener la atención en la carretera mientras conducen). Para esta aplicación el conjunto de palabras a reconocer son los números del 0 al 9.

El reconocimiento automático del habla es un proceso mediante el cual las señales acústicas del habla se relacionan con texto [1]. La primera etapa de cualquier sistema de procesamiento de voz es el muestreo de la señal acústica. Pero estas muestras no se suelen utilizar como entradas directas al sistema de reconocimiento. Normalmente se realiza un preprocesamiento mediante el cual se extraen algunas características utilizando bancos de filtros y otras técnicas. La siguiente etapa es el reconocimiento de fonemas, grupos de fonemas o palabras.

La red neuronal que más se utiliza como clasificador es el perceptrón multicapa, para el que se han desarrollado múltiples algoritmos de aprendizaje [2]. El perceptrón multicapa consiste en una red neuronal organizada en capas de elementos de procesamiento. Normalmente se compone de tres o más capas: una de entrada que recibe las variables de entrada para su clasificación, una o más capas ocultas y una capa de salida con un elemento por cada clase (Fig. 1.a). Cuando se presenta un patrón de entrada, los elementos de la red realizan distintos cálculos en capas sucesivas y los resultados se van propagando hasta los elementos de salida. Normalmente, el procesamiento completo de un patrón de entrada termina con la selección de un solo elemento de salida activo y el resto a cero.

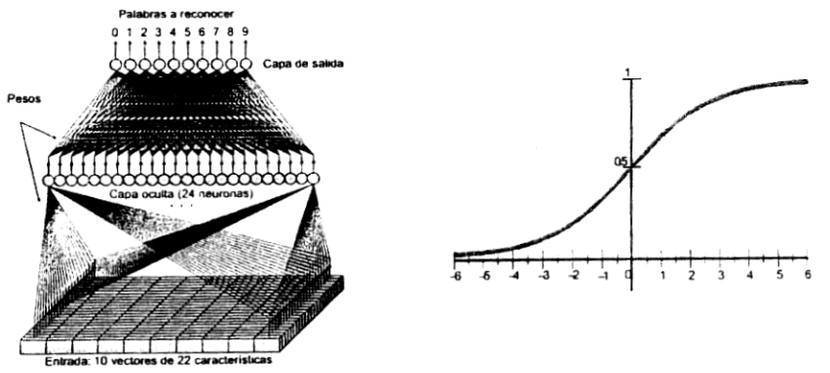
Cada elemento de procesamiento de una capa se conecta con todas las neuronas de la capa siguiente y la anterior. Las conexiones están moduladas por pesos que definen el comportamiento de la red y se ajustan durante el entrenamiento siguiendo un algoritmo de entrenamiento supervisado llamado de *"backpropagation"* [2].

Tras el proceso de entrenamiento, se presentan nuevos patrones de entrada. En las distintas capas, cada elemento de procesamiento calcula la suma ponderada de todas sus entradas (1),

$$sum_i = \sum_j w_{ij} S_j \ , \tag{1}$$

donde w_{ij} es el peso que conecta el nodo j con el nodo i y S_j es la salida del nodo j. La salida de un elemento i es $S_i = f\,(sum_i)$, que se transmite a los nodos de la siguiente capa. Este procesamiento se realiza en las sucesivas capas hasta que se genera un vector de salida. La función f denota la función de activación de cada elemento de procesamiento. Normalmente se utiliza una función sigmoidea (Fig. 1.b).

$$f(sum_i) = \frac{1}{1 + e^{-sum_i}} \ . \tag{2}$$



(a) (b)

Fig. 1. (a) Ejemplo de perceptrón para reconocimiento de palabras. (b) Función sigmoidea de activación

2 Alternativas de Implementación Hardware

En la aplicación propuesta necesitamos un perceptrón de 220 entradas (10 vectores de 22 características extraídas en la capa de preprocesamiento) y 10 elementos de salida (correspondientes a las diez palabras que se pretenden reconocer). Tras probar diferentes configuraciones, los mejores resultados (96.83% de clasificación correcta) se obtuvieron con 24 elementos en la capa oculta (Fig. 1.a).

Se ha utilizado coma fija con representación en complemento a dos para todos los cálculos y diferente número de bits para el almacenamiento de datos (entradas, pesos, función de activación, salidas, etc.). Es necesario limitar el rango de las diferentes variables:

- Entradas a la red y salidas de la función de activación. Ambas deben tener el mismo rango para poder definir estructuras multicapa sin necesidad de particularizar distintos tipos de neuronas. Se han escogido 8 bits.
- Pesos sinápticos de 8 bits.
- Entradas a la función de activación, que se define con una tabla de consulta (LUT) que almacena los valores útiles. Aparentemente se necesitan 23 bits para la entrada a esta función, pero debido a que se utiliza una función sigmoide (Fig. 1.b), la mayoría de valores se repiten (son uno o cero) y sólo hay que almacenar una pequeña zona de transición (aproximadamente un 1% de los valores).

Teniendo en cuenta todos estos aspectos relativos a la discretización, el modelo obtiene resultados similares a la simulación de la red con doble precisión. Los resultados se degradan en menos de un 1%.

La estrategia de almacenamiento principal se basa en el uso de los módulos RAM, de tal forma que las entradas, las salidas y los pesos asociados se almacenan en estos módulos RAM.

En esta sección se describen la versión secuencial y paralela de una arquitectura tipo perceptrón multicapa, utilizando dos niveles de abstracción diferentes: transferencia entre registros (con VHDL) y modo algorítmico (con Handel-C).

2.1 Nivel de Transferencia entre Registros (VHDL)

Se ha utilizado VHDL estándar como lenguaje de descripción de hardware para diseñar el perceptrón desde un nivel de transferencia entre registros. Todos los diseños se han realizado con la herramienta FPGA Advantage 5.3, de Mentor Graphics [3].

Antes de describir el diseño en detalle se ha analizado el tipo de operaciones que se requieren. Como se deduce de la ecuación (1), las operaciones básicas que realiza cada neurona consisten en multiplicaciones de las entradas (señales sinápticas) con los pesos y la suma de todas estas contribuciones. La Fig. 2 muestra la estructura básica de una unidad funcional (o elemento de procesamiento) para realizar estas operaciones. Esta unidad de procesamiento consiste en un multiplicador de 8 bits y un sumador acumulador de 23 bits. Para conectar la salida del multiplicador de 16 bits con el sumador de 23 bits se necesita una extensión de signo. Precisamos de 23 bits para acumular la multiplicación de 220 entradas con sus pesos.

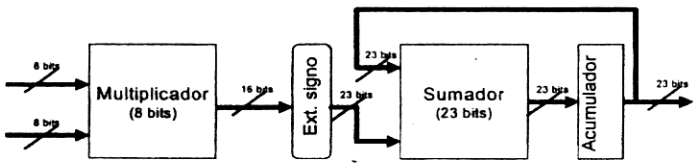


Fig. 2. Estructura básica de la Unidad Funcional

Versión secuencial

La arquitectura secuencial se ha diseñado para estimar el mínimo área necesario para implementar el perceptrón, aunque esto exige un tiempo de computación largo. Esta versión consta de una sola unidad funcional que realiza las operaciones correspondientes a todas las neuronas de la red de forma secuencial (Fig. 3.)

Las entradas de la unidad funcional son las entradas sinápticas y sus pesos que se almacenan en módulos RAM separados. En particular, hay un módulo que guarda todos los pesos, y dos módulos que guardan los valores de las entradas sinápticas, uno para los correspondientes a la capa de entrada y otro para los de la capa oculta. Se han utilizado dos módulos diferentes para poder leer de un módulo asociado a la capa de entrada y escribir la salida de una neurona de la capa oculta en el mismo ciclo de reloj.

La salida de la unidad funcional se conecta con el módulo de la función de activación. La salida de este módulo se guarda en el bloque RAM de la neurona oculta o de la neurona de salida, dependiendo de la neurona que se haya computado.

La dirección de la memoria RAM a la que se accede se calcula utilizando un contador de 8 bits y uno de 5. El contador de 8 bits direcciona el módulo RAM que almacena las entradas sinápticas (para leerlas) mientras que el contador de 5 bits las direcciona para escritura.

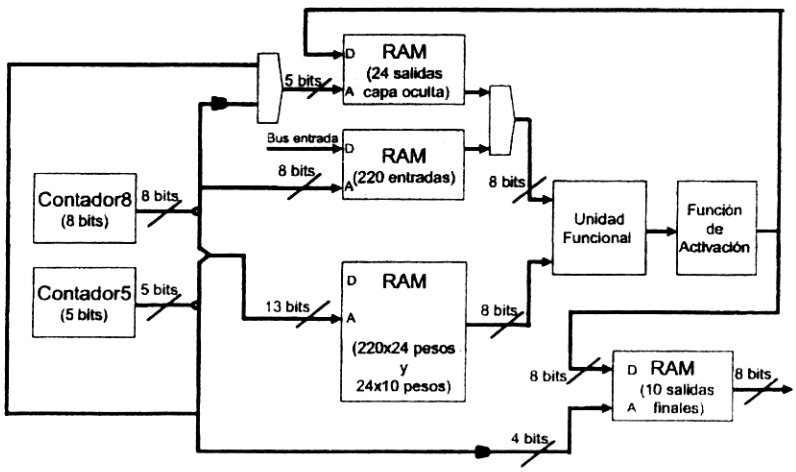


Fig. 3. Estructura de un perceptrón multicapa secuencial

La dirección de 13 bits de la RAM de los pesos se calcula concatenando las direcciones de los dos contadores, de tal forma que los bits más significativos corresponden al contador de 5 bits (véase la Fig. 3).

Versión Paralela

La arquitectura paralela que se describe a continuación implementa un paralelismo de nodos o elementos de computación de cada capa, es decir se implementa una unidad funcional por cada neurona en la capa con mayor número de ellas. Con esta estrategia, las neuronas de una capa trabajan en paralelo y por lo tanto generan sus salidas simultáneamente. Esto no es completamente paralelo porque las salidas de diferentes capas se obtienen en serie.

Para el perceptrón que se describe en este trabajo se necesitan 24 neuronas en la capa oculta, por lo tanto se requieren 24 unidades funcionales. Todas ellas trabajarán en paralelo cuando se calculen las salidas de la capa oculta y sólo 10 de ellas se utilizarán cuando se compute la capa de salida. La Fig. 4 muestra la estructura básica de esta versión paralela.

Como todas las unidades funcionales trabajan en paralelo para cada entrada sináptica, necesitan acceder a sus pesos asociados simultáneamente. Por lo tanto la RAM de pesos debe ser privada para cada unidad funcional. Por ello, se han implementado 24 módulos RAM para los pesos.

Cada unidad funcional necesita algún almacenamiento local para el dato de salida porque la transmisión de datos entre capas es serie. Se ha decidido utilizar 24 registros (DFF en la figura) paralelos, en vez del módulo RAM de la versión secuencial, porque esta opción permite reducir el tiempo de escritura. Los datos de estos registros paralelos se introducen en un único módulo de función de activación mediante un multiplexor 24:1 cuyas señales de selección son las salidas del contador de 5 bits. La

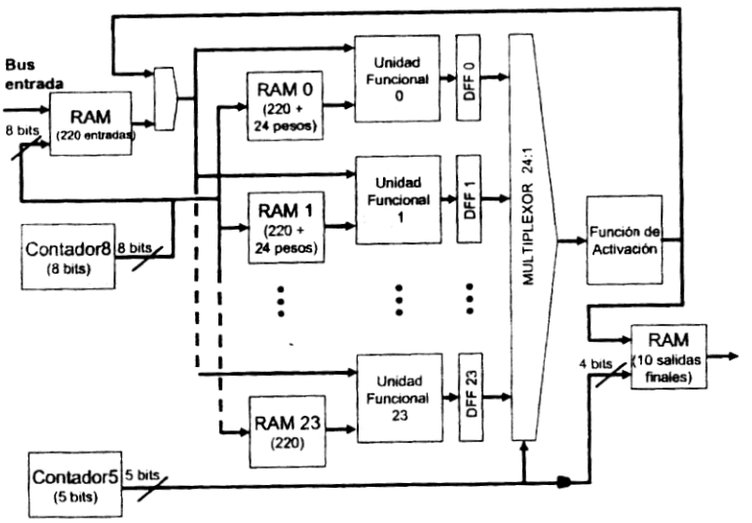


Fig. 4. Estructura de la versión paralela del perceptrón multicapa

salida de la función de activación es utilizada tanto como señal sináptica para las neuronas de la capa de salida o guardada en el módulo RAM de salida.

2.1 Nivel de descripción alto (Handel-C)

El diseño de alto nivel se ha definido utilizando Handel-C [4] como lenguaje de descripción de hardware. Handel-C es un lenguaje para la implementación rápida de algoritmos. Todo el proceso de diseño se ha realizado con la herramienta de desarrollo DK1 de Celoxica [4].

Versión Secuencial
El perceptrón computa las señales sinápticas de cada neurona en la capa oculta procesando cada entrada secuencialmente, y posteriormente, las salidas obtenidas se procesan de igual manera en la distintas capas hasta que se alcanza la salida. Todas las multiplicaciones se realizan secuencialmente.

Versión Paralela
En este caso, todas las neuronas de una misma capa computan sus resultados simultáneamente, en paralelo, excepto al acceder a la función de activación que se realiza en serie. Los diseños paralelos se han definido utilizando la directiva "par" de Handel-C que fuerza la implementación de circuitos dedicados para ciertas partes del algoritmo, para su computación paralela. La utilización óptima de la directiva "par" requiere un análisis previo del conjunto de operaciones y transferencias de información que se pueden realizar en paralelo. Por ejemplo, el diseñador debe saber que los módulos RAM sólo permiten en acceso a una palabra en cada ciclo de reloj.

3 Resultados

Los sistemas se han diseñado utilizando los entornos de desarrollo FPGA Avantage y DK1.1 para extraer los ficheros EDIF. Todos los diseños se han proyectado finalmente en una FPGA VirtexE 2000, utilizando la herramienta Xilinx Foundation 3.5i [5].

La Tabla 1 muestra los resultados de implementación obtenidos después de sintetizar los diseños descritos con VHDL tanto de la versión secuencial como la paralela. Los resultados se caracterizan con los siguientes parámetros: número de *slices*, número de bloques de memoria RAM embebida (EMB RAM), periodo mínimo de reloj, número de ciclos de reloj requeridos para la evaluación de cada vector de entrada (220 componentes de entrada) y el tiempo total consumido para la evaluación de un vector de entrada.

Como era de esperar, los resultados indican que la versión secuencial con una sola unidad funcional requiere menos recursos que la versión paralela con 24 unidades funcionales. Por otro lado la versión paralela es unas 20 veces más rápida que la versión secuencial. De esta forma, aprovechamos el paralelismo inherente al esquema de computación de las redes neuronales artificiales.

Diseño	# slices	% slices	# EMB RAMs	% EMB RAMs	Reloj (ns)	# Ciclos	Tiempo de Eval.(μs)
Serie	379	1.5	19	11	49.024	5630	276.005
Paralelo	1614	8.5	26	16	53.142	258	13.710

Tabla 1. Características de implementación de los diseños con VHDL

La Tabla 2 presenta los resultados obtenidos después de sintetizar las versiones secuenciales y paralelas del perceptrón descrito con Handel-C. Como se comentó en la sección 2.1, la obtención de un sistema eficiente requiere de un análisis de las distintas opciones. Cuando se programa el perceptrón se pueden considerar distintas alternativas con respecto al almacenamiento de datos: (a) sólo se utiliza RAM distribuida para todo el diseño, (b) los pesos asociados a las señales sinápticas se guardan en EMB RAMs, mientras que el resto de datos se almacena en memoria distribuida, (c) sólo se utilizan EMB RAMs para el almacenamiento de todos los datos.

Los resultados de la Tabla 2 muestran que independientemente de la opción de memoria escogida, la versión paralela requiere más recursos hardware que la versión secuencial. Por otro lado la versión paralela es unas 20 veces más rápida que la versión secuencial como ocurría en los diseños descritos con VHDL. La opción de memoria dependerá de las especificaciones de tiempo y de área del sistema. Pero los diseños realizados con Handel-C pueden cambiarse de forma sencilla para ajustarse a los requerimientos de la aplicación.

La comparación entre las Tablas 1 y 2, muestra que la implementación a nivel de transferencia de registros (definido con VHDL) de un sistema da lugar a mejores prestaciones en área y tiempo de computación. Pero la mayor ventaja de la descripción de alto nivel de sistemas es el corto tiempo de diseño y modificación. De esta forma, es conveniente comentar que el tiempo de diseño de la versión secuencial utilizando Handel-C ha sido unas 10 veces más corto que el que se utilizado para el diseño con VHDL. Además, para la versión paralela, los cambios necesarios utilizando Handel-C han sido relativamente pequeños (introduciendo las directivas *"par"*) mientras que la versión descrita con VHDL ha precisado del diseño de una nueva unidad de control.

Diseño	# slices	% slices	# EMB RAMs	% EMB RAMs	Reloj (ns)	# ciclos	Tiempo de Eval. (μs)
(a) Serie	2582	13	0	0	50.620	5588	282.864
Paralelo	6321	32	0	0	58.162	282	16.402
(b) Serie	710	3	24	15	62.148	5588	347.283
Paralelo	4411	22	24	15	59.774	282	16.856
(c) Serie	547	2	36	22	65.456	5588	365.768
Paralelo	4270	22	36	22	64.838	282	18.284

Tabla 2. Características de las implementaciones de los diseños definidos con Handel-C. (a) Sólo RAM distribuida. (b) EMB RAMs y RAM distribuida. (c) Sólo EMB RAMs

4 Conclusiones

Hemos presentado la implementación FPGA de un perceptrón multicapa para el reconocimiento del habla. Se han realizado los diseños de la versión secuencial y paralela utilizando distintos niveles de abstracción: nivel de transferencia de registros (definido con VHDL) y nivel algorítmico (definido con Handel-C). Los resultados muestran que los sistemas diseñados desde el nivel de transferencia de registros utilizando VHDL son más compactos y rápidos. Sin embargo, el tiempo de diseño es mucho más corto cuando se utiliza un nivel de descripción algorítmico.

Para la aplicación de reconocimiento del habla se obtienen niveles de clasificación de un 96.83% con unos tiempos de computación entre 14 a 16 microsegundos por muestra. De esta forma, cumple de sobra las restricciones temporales de la aplicación. Por lo tanto, la implementación presentada puede verse como un diseño de bajo coste (todo el sistema, incluso la versión paralela se podría programar en un dispositivo de bajo costo) y podría utilizarse como sistema embebido en una plataforma de reconocimiento del habla portátil (para sistemas controlados por voz).

Agradecimientos

Este trabajo ha sido financiado por CICYT TIC2002-00228 y SpikeFORCE (IST-2001-35271).

Referencias

1. Rabiner, L. and Juang, B.H.: Fundamentals of Speech Recognition, Prentice-Hall (1993)
2. Widrow, B., Lehr, M.: 30 years of adaptive neural networks: Perceptron, Madaline and Backpropagation. Proceedings of the IEEE, vol. 78, no. 9, pp.1415-1442 (1990)
3. Mentor Graphics, http://www.mentor.com/fpga/techpapers/index.cfm
4. Celoxica, http://www.celoxica.com/technical_library/handel-c/default.asp
5. Xilinx, http://www.xilinx.com/literature/index.htm