DISEÑO DE UNA ARQUITECTURA PARA LA IMPLEMENTACION DE UNA RED NEURONAL ARTIFICIAL PERCEPTRON MULTICAPA SOBRE UN FPGA

Manuel Monge Osorio, Mario Raffo Jara, Carlos Silva Cárdenas

Grupo de Microelectrónica de la Pontificia Universidad Católica del Perú
Sección de Electricidad y Electrónica

mmonge@pucp.edu.pe, mraffo@pucp.edu.pe, csilva@pucp.edu.pe

ABSTRACT

Neural networks are one of the most popular techniques that improve traditional methods in the area of information processing, and their best advantages are exploited in hardware implementation. Thus, this paper presents an architecture of a multi-layer perceptron (MLP) neural network for digit recognition. The design was done using VHDL and following a modular and generic description. It separates the integer and fractional parts of the synaptic weights to reduce the size of adders and multipliers in the MAC. Furthermore, by setting their parameters, the final design can implement different MLP neural networks to obtain the target task. The implementation was made over the development board DE2 based on Altera's EP2C35F672C6 Cyclone II FPGA with a PC serial interface for data interchange with the user.

1. INTRODUCCIÓN

Las redes neuronales son un paradigma de procesamiento de información que busca emular el comportamiento del sistema nervioso en el proceso de información [1]. Así, en la actualidad son utilizadas en muchas áreas debido a su gran flexibilidad y variedad de aplicaciones que presentan. Si bien la mayoría de estas se realizan en software (real-world applications), la importancia y vigencia de las implementaciones en hardware se mantiene y radica en aprovechar mejor las ventajas propias de las redes neuronales, que permiten un procesamiento en tiempo real y son usadas en aplicaciones que requieren un alto desempeño [1] [2]. Sin embargo, este tipo de implementación es aún un tema abierto a investigación, pues se debe, entre otros aspectos, optimizar la relación entre su eficiencia y el área utilizada en el chip [3].

El presente trabajo muestra el desarrollo de una red neuronal perceptron multicapa (MLP) aplicada al reconocimiento de dígitos manuscritos. La red MLP 49-25-10 fue entrenada en software con la herramienta Neural Networks Toolbox de Matlab. Para este proceso se utilizó la base de datos modificada del NIST (MNIST) [4], la cual cuenta con imágenes de dígitos manuscritos de 28x28 pixeles, que son umbralizadas antes de la entrada a la red neuronal.

Teniendo en cuenta estas dimensiones de la red, se diseñó la arquitectura combinando el planteamiento expuesto por Volnei A. Pedroni [5] con la idea propia de separar las partes enteras y decimales de los pesos sinápticos y umbrales encontrados en el entrenamiento. De esta forma se puede tener mayor exactitud con los decimales y mayor facilidad para realizar escalamientos si se tiene magnitudes muy elevadas en la parte entera.

Así, el diseño se realizó de manera modular y genérica para que pudiese ser adaptado fácilmente a un nuevo requerimiento con un re-entrenamiento de la red en software, extracción de los parámetros necesarios para el sistema e ingreso de los mismos en la descripción del circuito para su posterior síntesis e implementación en el FPGA.

2. DISEÑO DE LA RED NEURONAL MLP

El proceso de diseño de la red empezó por la elección de las dimensiones de la red. Primeramente, como se utilizó la base de datos del MNIST, la entrada del sistema es una imagen de 28x28 pixeles. Con el fin de reducir la cantidad de datos se realizó un proceso de umbralización para obtener una imagen de 28x28 bits. Esta imagen es dividida en segmentos de 4x4 bits, los cuales son sumados para obtener un vector de 49 elementos cuyos valores varían entre 0 y 16.

Dado que la red neuronal debe reconocer los dígitos decimales, se tendrá 10 neuronas en la capa de salida, con lo cual cada neurona indica si el dígito presentado a la entrada pertenece o no al dígito que representan. Para la cantidad de capas ocultas, se eligió una sola capa basándonos en el Teorema de Aproximación Universal (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989) [6]. La figura 1 muestra el diseño del sistema completo,

ISBN-13 978-968-7938-03-5

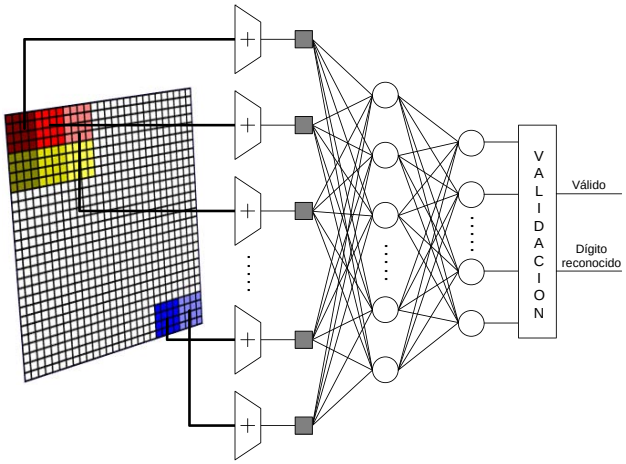

Figura. 1. Sistema completo: división de la imagen, red neuronal y etapa de validación.

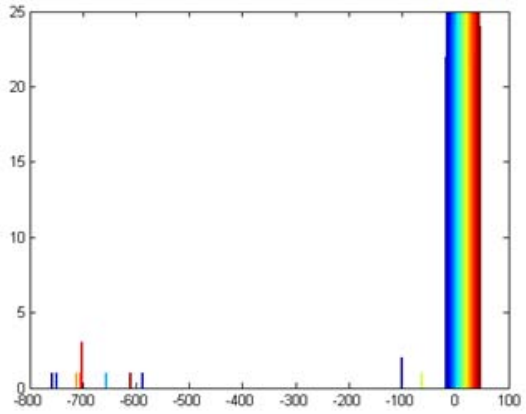

Figura. 2. Histograma de los valores obtenidos como pesos sinápticos.

en el cual se añade una etapa de validación de los resultados, en la cual se comparan las posibles neuronas ganadoras con un rango mínimo permisible que garantice que el patrón ha sido reconocido.

Con estas consideraciones se procedió al aprendizaje de la red neuronal en software. Se escogieron 25 neuronas para la capa oculta porque con esta configuración se obtuvo mejores resultados tanto en eficiencia como en dimensiones en comparación con otras formas de agrupar los bits de la imagen y con otras configuraciones de la red en cuestión. La eficiencia obtenida, en software, tras un entrenamiento de 100 *epochs* evaluando la base de datos de test del MNIST es de 92.11% con una valor máximo de un peso sináptico de -767.25. Se realizó un histograma de los valores obtenidos (figura 2) en el cual se observa que la mayor cantidad de los datos están concentrados en valores pequeños cercanos a cero con algunos *outliers*, los cuales son necesarios para el correcto funcionamiento de la red.

Observando el histograma, se aprecia la importancia de las cifras decimales (concentración de datos cercanos a 0) en los cálculos a realizar. De esta forma, en una representación de punto fijo, si quisiéramos una aproximación de centésimos se requerirían 7 bits para la parte decimal y, sumados a los 10 bits para la parte entera (valor máximo de 767.25) y 1 para la representación en complemento a 2, generarían 18 bits como entrada al multiplicador. En cambio, se pueden partir los pesos sinápticos en parte entera y decimal y trabajar ambas por separado y en paralelo. Así, se reducen los bits de entrada a los multiplicadores; pues para una precisión de centésimas se requieren de 7 bits más 1 bit para el signo y para la parte entera se mantiene lo anterior con 10 bits y 1 más para el signo.

Este proceso se explica a continuación para una neurona de *n* entradas, donde *p* es el vector de entrada de *i* elementos, *w* es el vector de pesos sinápticos de *i* elementos y *c* es el resultado de la suma de productos:

$$c = \sum_{i=1}^{n} p_i \cdot w_i = \sum_{i=1}^{n} p_i (w\_ent_i + w\_dec_i), \quad (1)$$

Tabla 1. Bits para la representación en complemento a 2 de los pesos sinápticos y umbrales de la red

Parte	Capa oculta		Capa de salida	
	w	b	w	b
Entera	11	4	9	4
Decimal	8	8	8	8

$$c = \sum_{i=1}^{n} p_i \cdot w\_ent_i + \sum_{i=1}^{n} p_i \cdot w\_dec_i, \quad (2)$$

La tabla 1 muestra los parámetros de los pesos y umbrales obtenidos tras el proceso de entrenamiento; donde *w* representa los pesos sinápticos y *b* los umbrales.

3. ARQUITECTURA DE UNA NEURONA

La arquitectura propuesta (figura 3) consta de un bloque funcional constituido por dos MAC's, un par de sumadores, un bloque adaptador que junta la parte entera y decimal del resultado y los acondiciona para evaluarla en la función de activación, en este caso la función sigmoide; y por un bloque seleccionador de los pesos sinápticos según la secuencia requerida. La ecuación 3 representa la operación que realiza la neurona; donde φ representa la función sigmoide, a_j la salida de la neurona *j* y *j* el número de neuronas por capa.

$$a_j = \varphi\left(\sum_{i=1}^{n} p_i \cdot w_{ij} + b_j\right), \quad (3)$$

Así, el contador permite la interacción de cada uno de los pesos sinápticos con el patrón de entrada mediante multiplexores. Dado que los valores de los pesos sinápticos y de los umbrales son constantes, se aprovecha esta característica al utilizar conexiones a '1' o '0' lógico envés de recursos lógicos como memorias.

La función sigmoide (figura 4) fue implementada según lo desarrollado en [7]. Para ello, los valores reales

ISBN-13 978-968-7938-03-5

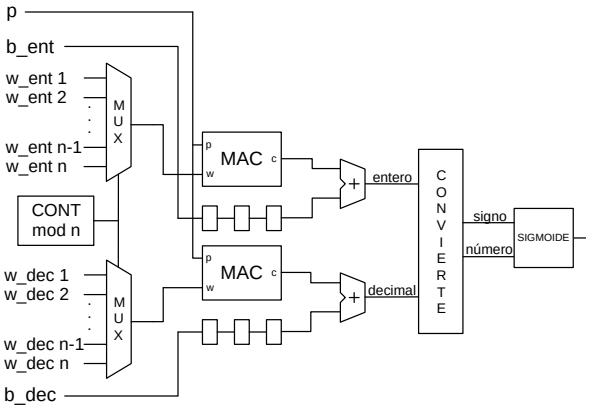

Figura. 3. Neurona Artificial.

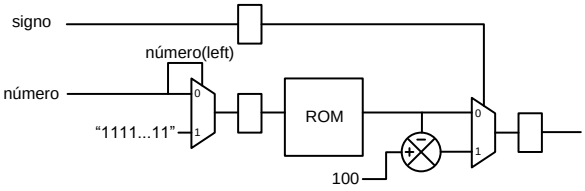

Figura. 4. Function Sigmoide.

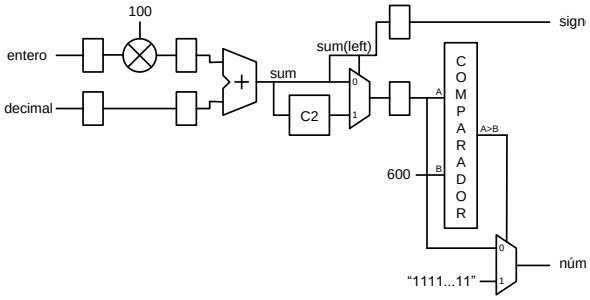

Figura. 5. Bloque Convierte.

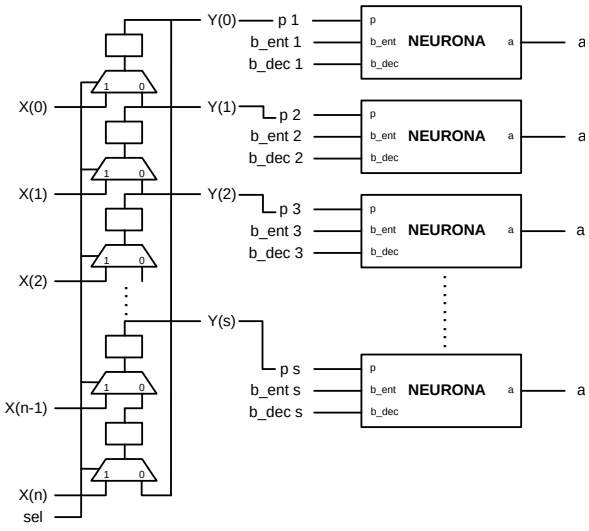

Figura. 6. Capa Neuronal: registro de desplazamiento circular unido con las neuronasde la capa.

de la función fueron escalados por un factor de 100 y redondeados a enteros. Además, el valor de saturación de la función se estableció en 5.11, con lo cual se utilizó una memoria de 512 posiciones, en la cual el valor de saturación corresponde a la posición 511.

Para esto, se requiere una etapa de adaptación que permita acoplar los resultados de las operaciones aritméticas divididas como lo muestra la figura 3 y los parámetros de entrada del bloque de la función de activación. De esta manera, el bloque *Convierte* realiza esta adaptación mediante la arquitectura mostrada en la figura 5. El resultado junta en un solo número escalado por 100 sus dos partes predecesoras (entera y decimal), y los muestra en dos salidas; siendo la primera el signo del número y la segunda el valor absoluto del mismo.

4. ARQUITECTURA DE LA RED NEURONAL

Teniendo el bloque funcional básico completo, la neurona artificial, la red se conforma en base a capas neuronales; en este caso, de una capa oculta de 25 neuronas y una capa de salida de 10 neuronas. La arquitectura de la capa neuronal se muestra en la figura 6. Está compuesta por un registro de desplazamiento circular que mediante la entrada de selección permite la carga de datos o la rotación de los mismos y por las neuronas respectivas que van acopladas a cada salida de la etapa anterior.

Para realizar esta unión, se debió tener en cuenta el ordenamiento de los pesos sinápticos en cada neurona según su posición en el arreglo circular. Del mismo modo, la activación de las banderas que permiten el funcionamiento de la capa neuronal ha sido considerada en este tipo de desplazamiento para no provocar errores de operaciones adicionales.

Respecto a la capa de salida, la neurona básica sufrió un cambio para adaptarse al rango valores que tiene en su entrada, de 0 a 100. Como estos representan valores escalados por 100, el bloque *Convierte* deja de multiplicar la parte entera y pasa a dividir entre 100 el número de la parte decimal que presenta en la entrada.

Los bloques adicionales permiten la umbralización y extracción de características de la imagen, y la validación y codificación del resultado obtenido en la red neuronal. Cada etapa del sistema está gobernada por un controlador (maquina de estados) que permite la interacción entre los distintos bloques.

5. RESULTADOS

El diseño de la red neuronal se realizó utilizando el lenguaje de descripción de hardware VHDL, considerando una descripción genérica y modular, y fue sintetizado usando las herramientas del Quartus II v.7.1. El dispositivo utilizado para la implementación fue el FPGA Cyclone II EP2C35F672C6 de Altera.

Los resultados del proceso de síntesis se muestran en la tabla 2, donde se observa la diferencia existente entre la neurona utilizada en la capa oculta y la de la capa de

ISBN-13 978-968-7938-03-5

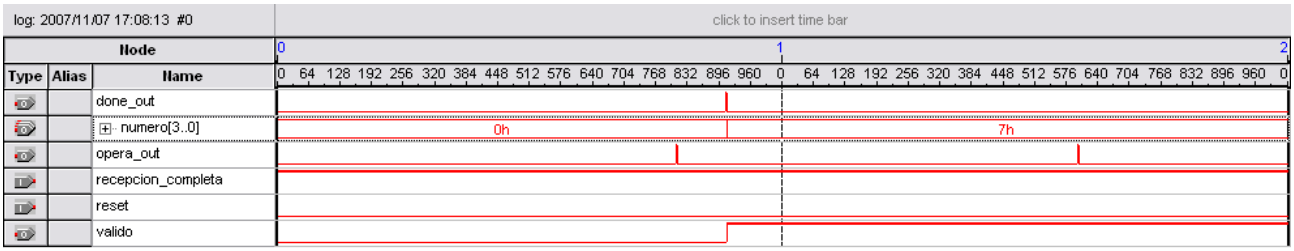

Figura. 7. Resultados del circuito implementado capturados por el Signal Tab II Logic Analyzer.

Tabla 2. Recursos utilizados en el FPGA

Recursos	Neurona		Red Neural	Sistema
	1	2		
Elementos lógicos	373	711	17188 (52%)	18511 (56%)
Registros	166	490	8680 (26%)	9439 (28%)
Bits de Memoria	4096	4096	143920 (30%)	148624 (31%)
Multiplicadores	1	2	45 (64%)	45 (64%)
Frecuencia máxima	146.78 MHz	116 MHz	104.12 MHz	100.97 MHz

salida. Esto se debe a la presencia de un divisor en la arquitectura, lo cual indica que es el bloque limitante para definir la frecuencia de operación de la red neuronal.

La red neuronal fue implementada en la tarjeta de desarrollo DE2 elaborada por Terasic con una frecuencia de 50 MHz. La figura 7 muestra los resultados obtenidos tras la implementación utilizando el Signal Tap.II Logic Analyzer.

Se observa que el circuito se comporta según lo esperado. Además, el tiempo requerido para reconocer un dígito es de 18.26us y que de ese tiempo tan solo 2.04us son utilizados por la red neuronal. Comparándolo con los 7.089ms que tarda el Matlab 7.1 en realizar el reconocimiento del dígito, podemos ver la principal ventaja que presenta la implementación en hardware de las redes neuronales frente a su contraparte en software.

6. CONCLUSIONES

El método aplicado en el diseño permite una fácil configuración del sistema puesto que todos los bloques presentes son genéricos y modulares. Así, la red neuronal puede ser re-entrenada para alcanzar los objetivos de la tarea requerida. De esta manera, con pocos cambios en su estructura, se pueden obtener diferentes redes neuronales.

Como se mencionó anteriormente, el tiempo de procesamiento requerido para reconocer un dígito es muy pequeño en comparación con lo requerido por una aplicación en software. Ello permite que nuestro diseño pueda ser utilizado en alguna aplicación de tiempo-real.

La principal ventaja de la arquitectura propuesta es el procesamiento paralelo que presenta la red neuronal; lo cual permite un rápido procesamiento, aunque ello incremente el área utilizada en el dispositivo. Sin embargo, puede ser usado en aplicaciones que requieran un mejor desempeño en velocidad que en área utilizada.

Además, la arquitectura puede ser modificada para obtener una versión serial o mixta entre la paralela y la serial. El compromiso que se genera entre la velocidad y el área utilizada en el dispositivo debe ser considerado en el diseño y ser escogido según la aplicación.

Otro método para reducir el área utilizada dentro del FPGA es usando su capacidad propia de reconfiguración. Ello permite, por ejemplo, tener en un momento la primera capa oculta, posteriormente, la segunda capa oculta en la misma área del dispositivo y finalmente la capa da salida con los resultados de toda la red neuronal. Este método es más adecuado para redes más grandes porque cada neurona contiene, al menos, un multiplicador; y el mejor desempeño es alcanzado utilizando los multiplicadores embebidos.

7. REFERENCIAS

[1] E.M. Ortigosa, A. Cañas, E. Ros, P.M. Ortigosa, S. Mota, J. Díaz, “Hardware description of multi-layer perceptrons with different abstraction level”, Microprocessors and Microsystems 30, pp 435–444, 2006.

[2] F. Smach, M. Atri, J. Mitéran, M. Abid, “Design of a Neural Networks Classifier for face detection”, Journal of Computer Science 2, Vol. 3, Pages 257 – 260, 2006.

[3] S. Vitabile, V. Conti, F. Gennaro, F. Sorbello, “Efficient MLP Digital Implementation on FPGA”, 8th Euromicro conference on Digital System Design 2005.

[4] Yann LeCun, León Bottou, Yoshua Bengio, and Patrick Haffner, “Gradient-Based Learning Applied to Document Recognition”, Proceedings of the IEEE, pp 9-10, 1998.

[5] Volnei A. Pedroni, “Circuit Design with VHDL”, MIT Press, 2004.

[6] Simon Haykin, “Neural Networks: a comprehensive foundation”, Macmillan, 1994.

[7] “Diseño e Implementación de una Arquitectura para el cálculo de la Función Sigmoide en un FPGA Cyclone II de Altera” M. Monge, J. Saldaña, C. Silva, “XIV INTERCON”,IEEE sección Perú, Piura, Agosto 2007

ISBN-13 978-968-7938-03-5