PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UNA ARQUITECTURA PARA UNA RED NEURONAL ARTIFICIAL PERCEPTRON MULTICAPA SOBRE UN FPGA APLICADA AL RECONOCIMIENTO DE CARACTERES

Tesis para optar el Título de Ingeniero Electrónico

Presentado por:

MANUEL ALEJANDRO MONGE OSORIO

Lima - PERÚ

2008

RESUMEN

El presente trabajo realizó el diseño genérico y modular de una red neuronal artificial perceptron multicapa MLP orientada al reconocimiento de dígitos manuscritos en un FPGA mediante el lenguaje de descripción de hardware VHDL. El entrenamiento de esta red se realizó externo al chip, en software, mediante la herramienta de Redes Neuronales del Matlab 7.1 y utilizando como imágenes de entrenamiento la base de datos modificada del NIST (MNIST database). Con esto, se logra que el FPGA se dedique solamente a la tarea de reconocimiento, mas no al aprendizaje de la red. Si se quisiera que se cumpla con otra aplicación, bastará con su re-entrenamiento en software para obtener los parámetros necesarios e introducirlos en su descripción y configuración.

Según esto, el diseño se desarrolló en tres partes. La primera muestra el proceso seguido para encontrar los parámetros requeridos por la red neuronal, como son sus dimensiones, pesos sinápticos y umbrales. La segunda consiste en el diseño de la neurona artificial, bloque fundamental de la red y en la cual, debido a la importancia de los decimales y a la presencia de valores elevados en los pesos sinápticos, se trabajó separando las partes enteras y decimales. De esta manera se reducen las dimensiones de los multiplicadores, factor importante en el desempeño. Finalmente, la tercera muestra la arquitectura de una capa neuronal y las consideraciones a tener en cuenta para realizar la unión entre los datos de entrada y las neuronas componentes de la capa. También, se desarrollaron bloques adicionales para que la red neuronal pueda interactuar con la imagen de entrada y para que el resultado obtenido sea codificado y validado.

La red neuronal fue implementada en la tarjeta de desarrollo DE2 basada en el FPGA Cyclone II EP2C35F672C6 de Altera con una frecuencia de 50 MHz. En esta etapa se utilizaron las herramientas disponibles en el Quartus II v. 7.1., así como el Signal Tap II Logic Analyzer. Se obtuvo que el tiempo requerido para reconocer un dígito es 18.26us y que de este tiempo, tan solo 2.04us son utilizados por la red neuronal. En comparación con los 7.089ms que demora una aplicación en Matlab 7.1, podemos observar la principal ventaja presente en las implementaciones en hardware de las redes neuronales.

A Margarita… mi madre… y más que una madre.

A Jesús Alberto y Jesús Augusto… grandes hermanos.

A mis incondicionales Cynthia, Lucía, Romina, Michael, Marcelo, Alonso… amigos como pocos.

A mi asesor Dr. Carlos Silva, por toda la confianza y todo el apoyo mostrado.

A mi querido Grupo de Microelectrónica (GuE), por haberme permitido descubrir mi verdadera vocación.

A Julio César Saldaña, Mario Raffo, Jorge Luis Lagos, Gerard Santillán, Walter Calienes, Héctor Villacorta, Heiner Alarcón, Erick Raygada, Jorge Lucio Tonfat, Joel Muñoz, Jorge Benavides, José Daniel Alcántara, José Francisco Quenta; por todos los consejos y enseñanzas, por todo el apoyo, paciencia y sincera amistad.

A mis profesores… por todos sus consejos y enseñanzas.

A todos… Gracias!

"Caminante, son tus huellas
el camino y nada más;
caminante, no hay camino,
se hace camino al andar.

Al andar se hace camino
y al volver la vista atrás
se ve la senda que nunca
se ha de volver a pisar.

Caminante no hay camino
sino estelas en el mar…"

Antonio Machado

ÍNDICE

INTRODUCCIÓN

Las redes neuronales son una teoría de procesamiento de información perteneciente a la inteligencia artificial que busca emular el comportamiento del sistema nervioso en el procesamiento de la información. Se basa en las múltiples interconexiones que se generan entre sus elementos básicos, denominados neuronas. De esta forma, según esta interacción pueden formar diferentes tipos de arreglos para cumplir una tarea específica.

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, la importancia y vigencia de las implementaciones en hardware se mantiene y radica en aprovechar mejor las ventajas propias de las redes neuronales y en que son usadas en aplicaciones que requieren un alto desempeño.

El presente trabajo muestra el desarrollo en hardware de una red neuronal artificial perceptron multicapa (MLP) aplicada al reconocimiento de dígitos manuscritos. La red MLP 49-25-10 fue entrenada en software con la herramienta de redes neuronales del Matlab (Neural Networks Toolbox). Para este proceso se utilizó la base de datos modificada del NIST (MNIST) que cuenta con 60000 imágenes binarias representativas de dígitos manuscritos de 28x28 pixeles, las que son umbralizadas antes de la entrada a la red neuronal.

Teniendo en cuenta estas dimensiones de la red, se diseña la arquitectura combinando el planteamiento expuesto por Volnei A. Pedroni 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 realiza 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. Adicionalmente, se pretende que esta arquitectura de propósito general sirva de base para futuras investigaciones y trabajos en el tema. De esta forma estaremos contribuyendo con la difusión de estas técnicas en nuestra industria.

CAPÍTULO 1

ASPECTOS GENERALES SOBRE LA IMPLEMENTACIÓN DE REDES NEURONALES

1.1. Procesos y problemas de elevada complejidad

En la actualidad, los nuevos requerimientos de la sociedad en función de sus nuevas necesidades de tratamiento de información y cómputo, no son cubiertos a cabalidad por los computadores vigentes. Incluso no son lo suficientemente eficaces para resolver tareas comunes de la actividad humana, pese al avance tecnológico alcanzado y a su continuo desarrollo.

Sin embargo, gracias a este avance de la tecnología y a la atmósfera de necesidad generada se han desarrollado nuevas teorías de procesamiento y se han retomado algunas ya estudiadas, las cuales resuelven dichos inconvenientes mediante la aplicación de nuevos algoritmos y nuevos paradigmas. Dentro de ellas se encuentran las Redes Neuronales, las cuales presentan un gran número de aplicaciones a través de su implementación tanto en hardware como en software.

Asimismo, es conveniente aplicar dichas tecnologías en nuestro país, para que de esta forma podamos contribuir en su desarrollo y modernización.

1.1.1. Utilización y aplicaciones de las Redes Neuronales

Las redes neuronales son utilizadas para el procesamiento de señales, reconocimiento y clasificación de patrones, síntesis y reconocimiento de voz, control de procesos y equipos, predicción, aproximación, direccionamiento, en el campo de la medicina, los negocios, en el ámbito industrial, social y científico. Algunas aplicaciones son los sistemas de control, sistemas de predicción, sistemas de seguridad, identificación, videoconferencia, procesamiento de documentos, reconocimiento de patrones como voz, caracteres, objetos.

La potencialidad presente en las redes neuronales y la existencia de un mercado y de clientes por satisfacer, debido a las necesidades actuales, genera una sostenibilidad económica que permite la implementación de las aplicaciones teniendo en cuenta su costo de inversión tecnológica.

1.2. Implementaciones de las redes neuronales

Las redes neuronales se implementan tanto en hardware como en software. Ambos tipos de desarrollo presentan un campo de acción definido, el cual depende de los requerimientos específicos de la aplicación; por lo cual, ella debe ser analizada para realizar una correcta elección. Un análisis costo-beneficio también debe llevarse a cabo para contar con una visión global de la aplicación.

Sin embargo, y pese a que la mayoría de implementaciones se realizan en software, la importancia y vigencia de la implementación en hardware se mantiene y radica en aprovechar mejor las ventajas propias de las redes neuronales, y en que sus aplicaciones necesitan de un elevado desempeño, y características inherentes al desarrollo en hardware, que no presenta la implementación en software [1] [2].

1.2.1. Implementación en Hardware

Si bien la implementación de redes neuronales en software se esta convirtiendo en la más apropiada y utilizada para la resolución de aplicaciones reales (real-world applications), las implementaciones sobre hardware siguen siendo esenciales e importantes debido a que permiten un procesamiento en tiempo real y que son utilizadas en pequeñas áreas específicas donde un muy alto rendimiento es requerido (por ejemplo: física de alta energía), en aplicaciones embebidas de redes simples (por ejemplo: chips de reconocimiento de voz), y en sistemas neuromórficos, los cuales implementan directamente una función deseada (por ejemplo: touchpad y silicon retinas)[1], aprovechando al máximo las ventajas inherentes del paralelismo masivo de las redes neuronales.

Sin embargo, la implementación en hardware de redes neuronales es aun un tema abierto a investigación, pues se debe optimizar la relación entre su eficiencia y exactitud, y el área utilizada en el chip [3]. De esta forma, su etapa de diseño no se encuentra completamente desarrollada y provoca deficiencias en comparación a la implementación en software.

Cabe mencionar que las limitaciones propias del software y la aparición de dispositivos lógicos programables más capaces vuelven necesarias a las implementaciones en hardware pese a sus propias limitaciones como son un mayor costo, mayor tiempo de diseño y mayor tiempo de puesta en el mercado (time-to-market).

1.2.2. Implementación en Software

Como ya mencionamos, las implementaciones en software vienen ganando aceptación debido a sus ventajas sobre las implementaciones en hardware en aplicaciones reales, como por ejemplo el reconocimiento de caracteres dentro de un sistema de restauración de libros o dentro de un sistema en el cual se cuenta con un computador que realiza distintas labores no críticas. Otra aplicación se da en el campo de control de procesos, o en las predicciones y aproximaciones, donde el tiempo de procesamiento no es un requerimiento del sistema. También se aprecia en el campo de la medicina, donde se utiliza como ayuda para el diagnóstico de enfermedades debido a su capacidad de detectar y clasificar patrones (señales del electroencefalograma o electrocardiograma, o diagnóstico por imágenes). En todos estos casos, las aplicaciones se realizan sobre software en un computador [1].

Dentro de las ventajas que presenta se encuentran las herramientas de diseño (Toolbox) y contar con herramientas de programación de alto nivel, que permiten un diseño en menor tiempo. Sin embargo, presenta limitaciones debido a que recarga al procesador (CPU) en una sola aplicación que le lleva “mucho tiempo”, pues tareas como clasificación y localización son de alto consumo de procesador; lo cual disminuye el desempeño global del sistema completo. Además, según la aplicación, no permite aplicaciones en tiempo real; pues la mayoría se realizan en hardware porque aprovechan mejor todas las ventajas de las redes neuronales [2].

CAPÍTULO 2

REDES NEURONALES Y SU IMPLEMENTACIÓN EN HARDWARE

2.1. Estado del arte

2.1.1. Presentación del asunto de estudio

Las nuevas teorías de procesamiento de información buscan cubrir las carencias que muestran los sistemas actuales de cómputo en la resolución de diversos problemas, los cuales son cada vez más complejos y sofisticados, debido a la gran cantidad de variables que deben manejar y a la dificultad intrínseca del problema. La inteligencia artificial es un claro ejemplo de esto, y de cómo dichas teorías van desplazando a las actuales pues presentan una mejor eficacia en estos campos.

Dentro de este grupo, se encuentran las redes neuronales artificiales, las cuales pretenden emular el comportamiento del cerebro en el procesamiento de información. Estas redes se conforman de varios elementos de procesamiento simples llamados neuronas, quienes se encuentran conectadas entre sí y procesan la información con un paralelismo masivo, logrando resolver varios y diversos problemas en distintas aplicaciones de diferentes áreas.

Para tal fin, las redes neuronales son implementadas tanto en software como en hardware según la aplicación. De esta forma, si es una actividad estática, se cuenta con un computador y el tiempo de procesamiento no es crítico, una aplicación en software será la adecuada; mientras que si se requiere un procesamiento veloz o en tiempo real, se necesita de un equipo portátil, ligero y de pequeñas dimensiones, una implementación en hardware es la mejor opción.

Según esto, el desarrollo de redes neuronales implementadas en hardware presenta un amplio campo de acción tanto a nivel industrial como comercial. La gran diversidad de aplicaciones hace indispensable contar con una arquitectura base, con la cual sea factible realizar posteriores trabajos de investigación para aplicaciones específicas.

Por lo tanto, el presente estudio busca el diseño de una arquitectura de una red neuronal, la cual será la base de posteriores desarrollos en el mismo campo y en el campo de los sistemas neuromórficos. Según dichos requerimientos, la arquitectura debe estar realizada en base a módulos y además debe tener diversas aplicaciones. De esta forma, se diseñará una arquitectura de una red neuronal conocida como Perceptron Multicapa o “Multi-Layer Perceptron” y, para probar su

correcto funcionamiento, se escogió una de sus principales aplicaciones, y que además se encuentra muy difundida, como es la etapa de reconocimiento del carácter de un sistema de reconocimiento óptico de caracteres o “Optical Character Recognition” (OCR).

2.1.2. Estado de la investigación

Para el proceso especifico del reconocimiento del carácter, dentro de las implementaciones hechas en hardware, existen diversas formas y métodos para su realización. Esto se debe también a la diversidad de la tecnología actual y a su continuo avance, lo cual permite el desarrollo de teorías ya existentes y de nuevas formas de procesamiento.

Entre estas podemos mencionar las redes neuronales, las cadenas ocultas de Markov, el voto mayoritario, sistemas basados en microprocesadores, transformada de Fourier, transformada de wavelet, métodos estadísticos de Bayes, árboles de decisión, lógica difusa, algoritmos genéticos, etc. Dentro de las cuales, destaca en este campo la implementación del sistema mediante redes neuronales, pues presentan un mejor desempeño, logrando tasas de éxito superiores al 99% para caracteres impresos [4], y capacidades de aprendizaje y adaptación. Por estas razones es el método más utilizado en la actualidad, encontrándose tanto en dispositivos específicos como en programas para el reconocimiento de caracteres.

La razón por la cual se desarrolla este modelo es por la poca eficiencia de los métodos tradicionales de procesamiento para resolver las tareas comunes de la actividad humana, pese a contar con una mayor capacidad que el cerebro humano [5]. Las neuronas, a pesar de ser mucho más lentas que las compuertas lógicas (pues los eventos en un chip ocurren en el orden de nanosegundos; mientras que los neuronales en milisegundos), procesan mejor la información y en mucho menor tiempo [6]. Esto lo logran debido al gran número de interconexiones o sinapsis que existen entre ellas mismas. Según esto, las Redes Neuronales Artificiales (ANN: Artificial Neural Networks) o simplemente Redes Neuronales (NN: Neural Networks) buscan modelar el comportamiento del sistema nervioso biológico a partir del conocimiento del mismo y de la biología de la neurona; utilizando sus respectivos modelos matemáticos.

Así, las redes neuronales son un procesador distribuido masivamente en paralelo que tiene la tendencia natural de almacenar conocimiento de forma experimental y

volverla disponible para su uso [7]. Su unidad de procesamiento de información es la neurona, la cual es fundamental para el funcionamiento de la red. Consta de las siguientes partes:

- Sinapsis o interconexiones, cada una caracterizada por un “peso sináptico” o fuerza de conexión. Dicho peso será positivo si la sinapsis es excitadora y será negativo si es inhibidora.
- Un sumador para sumar las señales entrantes multiplicadas por sus respectivos pesos sinápticos.
- Una función de activación, para limitar la amplitud de la salida de la neurona.

La figura 1 muestra las representaciones de una neurona biológica y una neurona artificial.

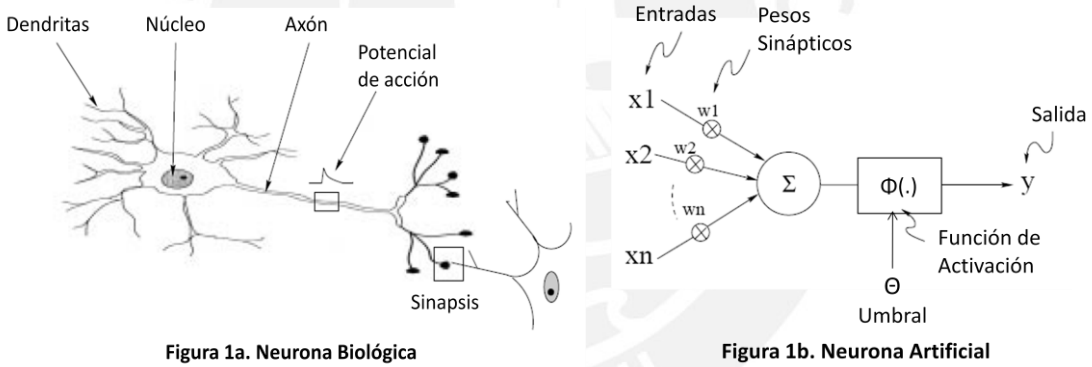


Figura 1a. Neurona Biológica

Figura 1b. Neurona Artificial

Figura 1. Neurona biológica y Neurona Artificial [8].

Su utilización satisface las demandas de los sistemas actuales, cuyos procesos son muy complejos y requieren de una mayor eficacia en su resolución. Las características de las redes neuronales que permiten lo dicho anteriormente son las siguientes [5] [6] [9]:

- Estructura distribuida masivamente paralela.
- Habilidad de aprendizaje.
- Habilidad de generalización.

- No linealidad.
- Mapeo entre entrada-salida.
- Adaptabilidad.
- Procesamiento de información local.
- Respuesta de confiabilidad.
- Información contextual.
- Memoria distribuida.
    - Memoria a largo plazo se encuentra en los pesos sinápticos.
    - Memoria a corto plazo corresponde a las señales enviadas por las neuronas.
- La fuerza de la sinapsis puede ser modificada por la experiencia.
- Los neurotransmisores pueden ser excitadores o inhibidores.
- Tolerancia a fallas.
- Capacidad de implementación VLSI.
- Uniformidad de análisis y diseño.
- Analogía neurobiológica.

Estas ventajas son aprovechadas en su mayoría por los sistemas actuales y son el motivo principal por la cual se prefieren frente a los métodos de procesamiento tradicionales. Por esto, las aplicaciones y usos de las redes neuronales son muy diversas y se encuentran en varios campos como son la robótica, control, reconocimiento de patrones, medicina, síntesis de voz, reconocimiento de voz, aproximación de funciones, predicción, optimización, reconstrucción de patrones, negocios y el entendimiento del procesamiento del cerebro humano [5][9][10].

El reconocimiento de caracteres es un caso particular del reconocimiento de patrones, razón por la cual dichas redes neuronales son capaces de realizar dicha labor; y, como ya mencionamos, son las más utilizadas debido a su gran eficiencia

y confiabilidad. En este tipo de tarea, se tiene un número fijo de categorías o clases dentro de las cuales las muestras deben ser clasificadas. De esta forma, la ventaja que presenta este método reside en el hecho que se pueden separar regiones no lineales de decisión tan complicadas como se requiera dependiendo del número de neuronas y/o de la cantidad de capas [7].

Las arquitecturas que realizan dicha tarea son las redes de una sola capa, las redes de múltiples capas o multicapas (MLP: MultiLayer Perceptron), las redes de base radial (RBF: Radial Basis Function network), redes ART (Adaptive Resonance Theory models) y Neocognitron [5] [9]. Sin embargo, la red más utilizada para esta tarea es la MLP, debido a que utiliza el algoritmo de aprendizaje que se encuentra actualmente más extendido, el algoritmo o regla BackPropagation [7].

En cuanto al hardware, las redes neuronales se encuentran implementadas de diversas formas, ya sean analógicas o digitales. En las primeras, se puede citar a los dispositivos analógicos programables, como los FPAA (Field Programmable Analog Array); mientras que en las segundas destacan los dispositivos lógicos programables como los FPGA (Field Programmable Gate Array), donde también se desarrollan los conceptos de FPNA (Field Programmable Neural Array) y FPNN (Field Programmed Neural Network), con los cuales el diseño de redes neurales más complejas se facilita en gran medida [11]. Finalmente, se tiene la tecnología ASIC (Application Specific Integrated Circuit, Circuito Integrado de Aplicación Específica) en donde, ya sea analógico o digital, se obtienen chips con un diseño y aplicación específica; y en este caso, de una red neuronal.

La tabla 1 muestra el compromiso existente entre algunos parámetros del diseño y algunas tecnologías para la implementación de redes neuronales. Observamos que los FPGAs poseen buenas prestaciones en la mayoría de los campos, siendo un dispositivo adecuado para dicha aplicación.

Tabla 1. Dispositivos apropiados/inapropiados para la implementación de redes neuronales [11].

	ASIC analógico	ASIC digital	FPGA	Basado en procesador	Computador paralelo
Velocidad	+++	++	+	-	+
Área	+++	++	+	-	--
Costo	--	--	++	++	--
Tiempo de Diseño	--	--	++	+++	+
Confiabilidad	--	+	++	++	++

-- : muy desfavorable
- : desfavorable
+ : favorable
++ : muy favorable
+++ : extremadamente favorable

En la actualidad, los FPGAs son muy utilizados en diversas aplicaciones, y también han sido usados para el diseño e implementación de redes neuronales de diversas arquitecturas. La más utilizada es la red multicapa (MLP) y el algoritmo de aprendizaje Backpropagation, por las razones mencionadas previamente y por las características inherentes propias del FPGA y los componentes que presenta; los cuales facilitan su implementación. Así, es utilizado para el reconocimiento de caracteres, dígitos, caras, firmas, etc.

2.1.3. Síntesis sobre el asunto de estudio

Los nuevos requerimientos de los procesos actuales y la incapacidad mostrada por los computadores para resolverlos generan la necesidad de buscar nuevos métodos para el procesamiento de información. Dentro de este ámbito, las Redes Neuronales Artificiales surgen como una alternativa eficaz ante este inconveniente, pues presentan un mejor desempeño en un menor tiempo y una gran variedad de aplicaciones. Para lograr tal fin, dichas redes neuronales deben pasar por un proceso de aprendizaje (entrenamiento) y luego por un proceso de prueba (test) para que finalmente sean colocadas en la aplicación deseada.

Por tal motivo, el presente estudio propone el diseño sobre un FPGA de una arquitectura de una red neuronal para que sea el precedente de posteriores desarrollos tanto en el campo de las redes neuronales como en el campo de los sistemas y circuitos neuromórifcos.

La arquitectura neuronal planteada es el perceptron multicapa, conocido como MLP, debido a que es una red genérica y resuelve un gran número de problemas en diversas áreas. Adicionalmente, para verificar su funcionamiento, será aplicado a la etapa de reconocimiento del carácter de un sistema de reconocimiento óptico de caracteres (OCR).

El diseño se llevará a cabo con el lenguaje de descripción de hardware VHDL (Very High Speed Integrated Circuit Hardware Description Language, Lenguaje de Descripción de Hardware de Circuitos Integrados de Alta Velocidad), el cual permite describir circuitos y sistemas digitales mediante el desarrollo de un código que es transparente frente a las diversas herramientas EDA (Electronic Design Automation). Además, se utilizará un entorno de desarrollo proporcionado por el fabricante que permite la interacción entre el diseñador y el FPGA.

Un punto importante que cabe resaltar es el hecho que el diseño depende del FPGA a utilizar, pues ellos presentan diferentes estructuras, componentes y limitaciones según la familia a la cual pertenezcan y la empresa que los desarrolla. De esta forma, debemos conocer a fondo los recursos que presenta el dispositivo para tenerlos en cuenta durante el desarrollo del diseño.

Finalmente, se obtendrán conclusiones a partir de los resultados obtenidos durante todo el proceso y a la finalización del mismo; los cuales serán útiles para futuros estudios e investigaciones en el mismo campo y en temas relacionados.

2.2. Reconocimiento del carácter dentro de un OCR

Un Reconocedor Óptico de Caracteres OCR (Optical Character Recognition) es un sistema que convierte la imagen de textos escritos o impresos en una forma tal que el computador pueda procesarlos posteriormente. Generalmente, se almacenan como caracteres ASCII dentro de un archivo de texto. Consta, básicamente, de las siguientes etapas: adquisición de la imagen, análisis del formato, segmentación, extracción de características y reconocimiento del carácter [12] [13].

La adquisición de la imagen es el proceso por el cual la imagen del texto es obtenida, por lo que es el punto de inicio de todo el proceso. Generalmente se emplean escáneres, pero también pueden emplearse cámaras [12] [13].

Seguidamente, el análisis del formato consiste en identificar los títulos, subtítulos, columnas, gráficos y formas del documento para realizar una adecuada lectura. De esta forma se evitan resultados erróneos [12] [13].

Posteriormente, la segmentación es el proceso que consiste en la separación o partición de un objeto en varias partes de menor magnitud. En el caso de la separación de caracteres, involucra reconocer los límites entre un carácter y otro y generar una nueva imagen para cada carácter segmentado. En este proceso, puede incluirse una etapa de normalización de la imagen, al poder existir diferencias en los tamaños de los caracteres [12] [13].

Después se prosigue con la extracción de características, la cual consiste en la obtención de parámetros que permitan realizar adecuadamente los respectivos análisis de las imágenes. Una primera tarea es la umbralización, que determina si un punto pertenece o no al carácter; si es así, le asigna un 1 lógico, caso contrario un 0 lógico. Posteriormente, se calcula la altura, el ancho, el área, la relación entre ancho y alto, la orientación; es decir, las características del carácter [12] [13].

Finalmente se realiza el reconocimiento del carácter, que consiste en la clasificación del carácter dentro de las clases posibles que lo definan como tal. En otras palabras, es la etapa que nos dice que carácter es y nos da a la salida dicho carácter en un formato que sea manipulable por la computadora, como es el ASCII [12] [13].

Esta última etapa es una de las más importantes, pues se encarga de realizar la clasificación, y es de quien depende el éxito o fracaso del proceso; convirtiéndola en factor importante de la eficiencia total del sistema.

2.3. Redes Neuronales

Una Red Neuronal Artificial es un paradigma de procesamiento de información que busca emular el comportamiento en el cual el sistema nervioso biológico procesa la información [1].

Adicionalmente, podemos decir que una red neuronal es un procesador distribuido paralelamente masivo que posee la tendencia natural de almacenar el conocimiento experimental y volverlo utilizable. Se asemeja al cerebro en dos aspectos:

- El conocimiento es adquirido por la red neuronal a través de un proceso de aprendizaje.
- Las fortalezas de las interconexiones, conocida como “pesos sinápticos” son usadas para almacenar el conocimiento [6].

Como ya se menciono, la unidad de procesamiento de la red es la neurona artificial, y su adecuado funcionamiento se logra a través del proceso de entrenamiento o aprendizaje. Según el ordenamiento de las neuronas de la red y del tipo de aprendizaje utilizado, se encuentran diversas clases de redes neuronales.

Según su arquitectura, pueden ser redes de una sola capa, redes multicapa y redes recurrentes. Según su método de aprendizaje se tiene dos tipos:a) según el paradigma usado puede ser supervisado, no supervisado o híbrido; y b) según el algoritmo de aprendizaje, este puede ser de error-corrección, Boltzmann, Hebbian, competitivo. Además, las funciones de activación utilizadas son la identidad, escalón unitario, sigmoide, tangente hiperbólica, de probabilidad Gaussiana [5] [6].

2.3.1. Redes Multicapa

Es aquella red que posee una capa de entrada, una capa de salida y una o más capas ocultas. Existen la red Perceptron Multicapa (Multi-Layer Perceptron, MLP) y la red de Base Radial (Radial-Basis Function Networks, RBF).

La red MLP "es una generalización del perceptron y surgió como consecuencia de las limitaciones de dicha arquitectura en lo referente al problema de la separabilidad no lineal" [15]; mientras que la red RBF posee una única capa oculta y sus neuronas tienen un carácter local; es decir, cada una de ellas se activa en una

región diferente del espacio de patrones de entrada, lo cual se debe al uso de las llamadas funciones de base radial como función de activación [14].

Estas redes multicapa se utilizan en aplicaciones como reconocimiento de patrones, aproximación de funciones, predicción, control [9].

2.3.2. Red Perceptron Multicapa (MLP)

Es aquella red que contiene una o más capas ocultas, muestra un alto grado de conectividad y sus funciones de activación son diferenciables [6].

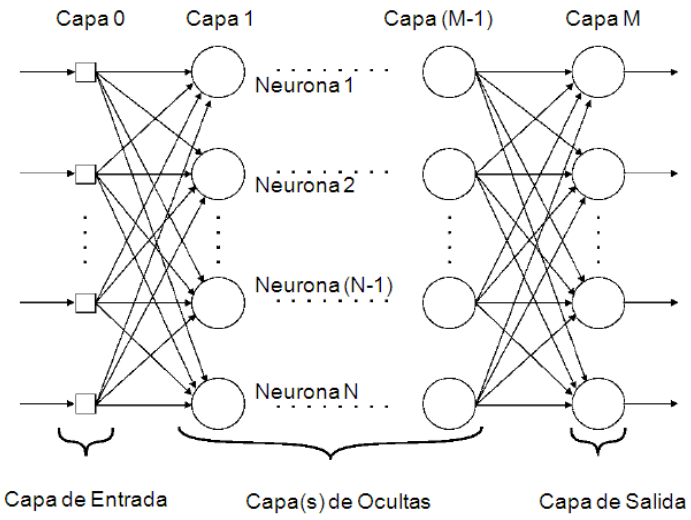


Figura 2. Red Neuronal Perceptron Multicapa (MLP) [11].

Se encuentra conformada por un arreglo de perceptrones, los cuales son la forma más simple de una red neuronal, inventada por Rosenblatt en 1958. El Perceptron consiste básicamente en una neurona simple con pesos sinápticos ajustables y un umbral de activación, y es utilizado para clasificar patrones linealmente separables.

El principio y sustento del perceptron se encuentra en el Teorema de convergencia del Perceptron:

“Dadas dos clases C1 y C2, dichas clases son linealmente separables si existe un vector de pesos sinápticos configurado para tal fin. También, si se conoce que dos clases C1 y C2 son linealmente separables, entonces existe un vector de pesos sinápticos que lo permita. Para lograr tal fin, los pesos sinápticos son ajustados (actualizados) según el error generado, las entradas y el parámetro razón de aprendizaje” [6].

La figura 3 muestra dos conjuntos de datos perteneciente a dos clases que son linealmente separables. Este diagrama ilustra el teorema de convergencia del perceptron, pues es este último quien establece la separación de las clases según sus pesos sinápticos.

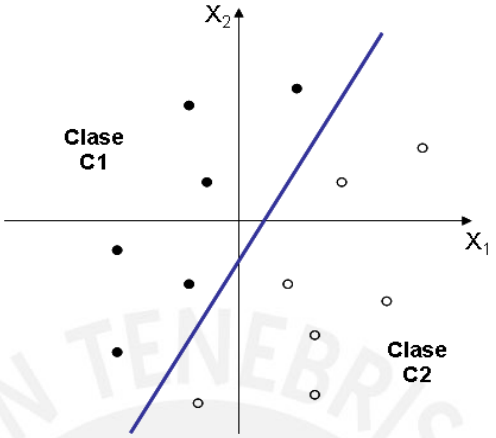


Figura 3. Teorema de Convergencia del Perceptron [6].

2.3.3. Paradigmas de Aprendizaje

Son las formas en la cuales se realiza el proceso de aprendizaje de la red neuronal. Pueden ser supervisado, no supervisado e híbrido [9] [15].

Es supervisado cuando la información es presentada a la red en forma de patrones de entrada y los resultados deseados son conocidos previamente. Un supervisor (profesor) verifica la salida de la red y la compara con la esperada. Por lo tanto, minimiza esta diferencia mediante la modificación de los pesos sinápticos.

Es no supervisado cuando la información es presentada a la red bajo forma de patrones de entrada y los resultados no son conocidos previamente, pues no requiere de la salida correcta a cada patrón de entrada. Este método explora la relación o correlación existente entre los patrones de los datos y los organiza agrupándolos en diferentes categorías según características en común.

El paradigma híbrido combina los aprendizajes supervisado y no supervisado aplicándolos en distintas capas de la red; es decir, algunos pesos sinápticos son ajustados a través del aprendizaje supervisado y los otros mediante el aprendizaje no supervisado [9] [15].

2.3.4. Aprendizaje Error-Corrección

En este método se genera la señal de error mediante la comparación entre la salida de la red ante el estímulo de un ejemplo y la señal deseada. Así, el objetivo es minimizar esta señal de error a través de la "función de costo" (cost function). De esta forma, el aprendizaje es estrictamente un problema de optimización mediante el ajuste de los pesos sinápticos [6].

Entre los algoritmos de aprendizaje que utilizan este método se encuentran el algoritmo LMS (Least Mean Square, Mínimo Error Cuadrático Medio), Back-propagation, Adaline, Madaline.

2.3.5. Algoritmo Backpropagation

Consiste en dos procesos, uno hacia adelante (forward pass) y uno hacia atrás (backward pass) [6].

En el primero, se coloca a la entrada de la red una señal de entrenamiento, la cual atraviesa la misma generando una salida que es comparada con la salida deseada, produciéndose una señal de error por cada neurona de la capa de salida.

En el segundo, dicha señal de error es propagada desde la salida hacia atrás, donde se realiza la actualización y modificación de los pesos sinápticos en función de la razón de aprendizaje, el gradiente local y la entrada de la neurona específica. Este proceso termina según un criterio de parada, el cual permite un error mínimo en la red.

2.4. Dispositivo lógico programable FPGA

Un Dispositivo Lógico Programable (Programmable Logic Device, PLD) es un chip de propósito general para la implementación de circuitos lógicos. Contiene un conjunto de elementos lógicos agrupados de diversas maneras. Puede ser visto como una “caja negra” que contiene compuertas lógicas e interruptores programables, los cuales permiten a las compuertas conectarse entre sí para formar cualquier circuito lógico requerido [16].

Se clasifican según el grado de complejidad y cantidad de recursos lógicos presentes en el dispositivo. Así, existen SPLD (Simple PLD), dentro de los cuales

están los PLA (Programmable Logic Array), PAL (Programmable Array Logic), PLA/PAL Registradas (Registered PLA/PAL), GAL (Generic PAL); CPLD (Complex PLD); FPGA (Fiel-Programmable Gate Array) [16][17].

Dentro de este contexto, un FPGA (Arreglo de puertas programable en campo) es un PLD que soporta implementaciones de circuitos lógicos y sistemas digitales relativamente grandes y son programables en campo, es decir, en el lugar del diseño, junto al diseñador [16]. Presenta una arquitectura virgen, la cual puede ser configurada a necesidad, logrando una gran diversidad de aplicaciones. El almacenamiento de estas conexiones se realiza en una SRAM en la mayoría de los casos, pero también puede realizarse mediante FLASH o antifuse [16][17].

2.4.1. Arquitectura genérica

Básicamente, un FPGA contiene bloques lógicos, recursos de interconexión y unidades de entrada y salida [16] [17].

Los bloques lógicos están conformados por tablas de verdad (Look-Up Table, LUT), la cual es un circuito programable que permite implementar cualquier función lógica de un número determinado de entradas, por flip-flops y multiplexores.

Para controlar las conexiones necesarias entre los bloques lógicos, los recursos de interconexión y las unidades de entrada y salida se programan dichas conexiones mediante técnicas de programación, siendo la más utilizada la tecnología de celdas SRAM.

Adicionalmente, según el FPGA, este puede contar con elementos adicionales y bloques dedicados como memorias, PLL, multiplicadores, MAC's, DSP's, microprocesadores.

2.4.2. Arquitectura del FPGA CYCLONE II de ALTERA

Es un FPGA basado en un proceso SRAM de 1.2V, 90nm con una alta densidad de elementos lógicos (Logic Elements, LE's) entre 4608 y 68416, y hasta 1.1Mbits de memoria RAM embebida [18].

Además de los elementos básicos, cuenta con bloques dedicados adicionales como bloques de memorias RAM embebidas de 4K, multiplicadores embebidos de 18x18, PLL's y unidades de entrada/salida avanzadas. La cantidad de cada uno de ellos depende del dispositivo elegido, pues estas características están presentes en toda la familia.

La disposición de sus componentes se muestra en la figura 4. Se aprecia los bloques de memoria, los multiplicadores embebidos, los PLL's, los arreglos lógicos y los bloques de entrada/salida (Input-Output Element, IOE).

Los arreglos lógicos están estructurados en base a los arreglos de bloques lógicos (Logic Array Blocks, LAB's), que son un arreglo de LE's, y a los recursos de interconexión locales (Local Interconnect) [18].

Además, se cuentan con filas y columnas de interconexión para realizar conexiones entre LE's de diferentes LAB's. La estructura del LAB se muestra en la figura 5.

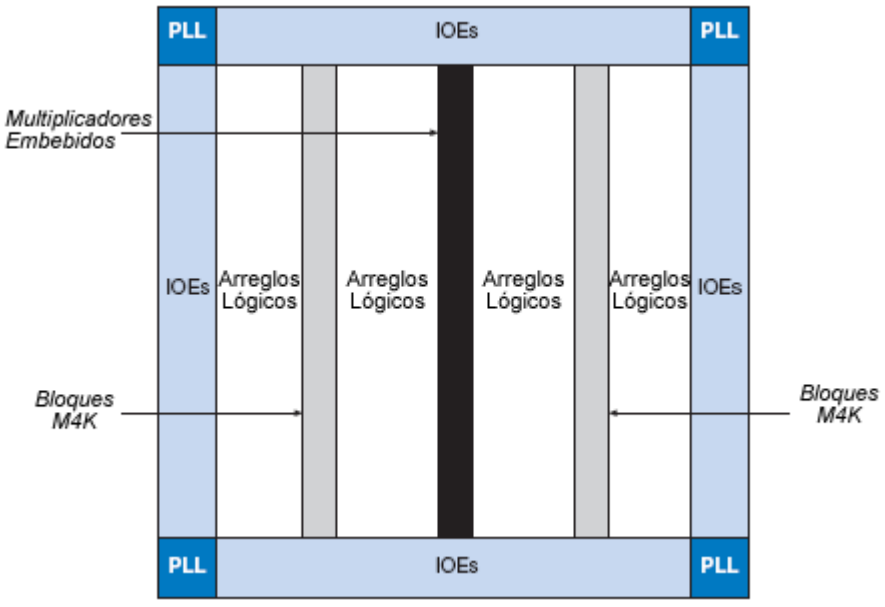


Figura 4. Diagrama de bloques del FPGA Cyclone II EP2C20 [18].

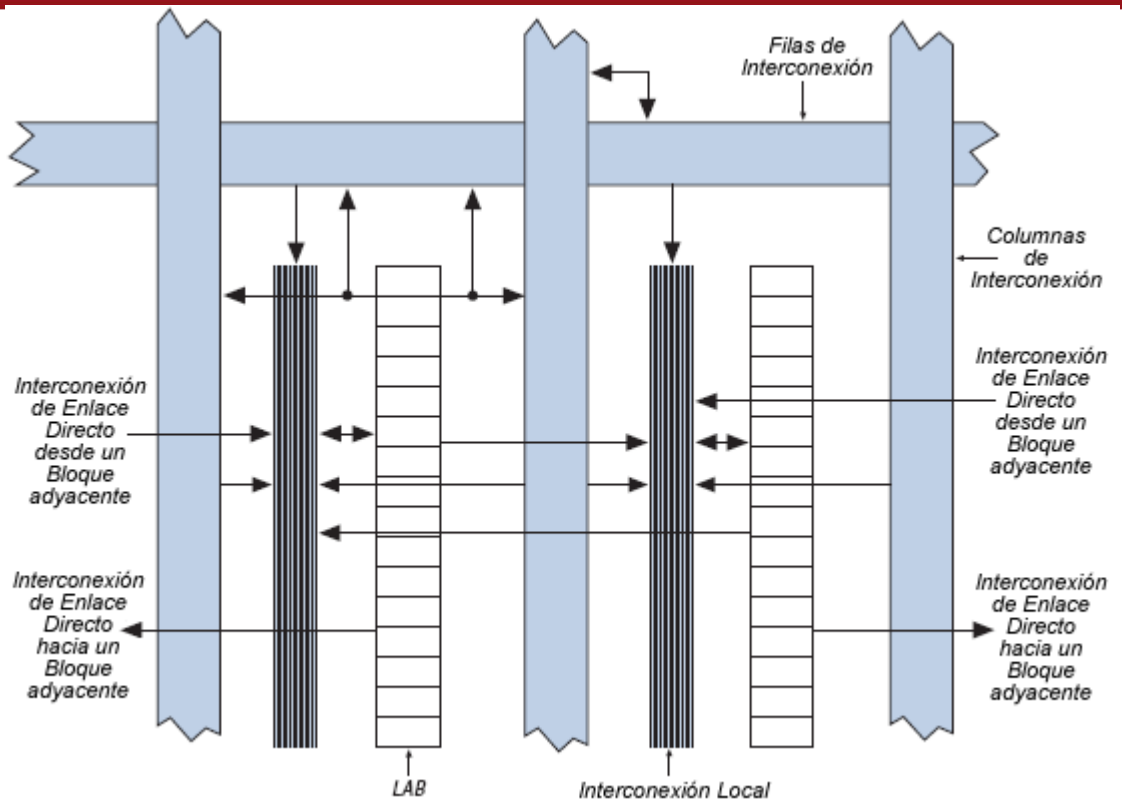


Figura 5. Estructura del LAB del FPGA Cyclone II [18].

2.5. Lenguaje de descripción de hardware VHDL

Es un Lenguaje de Descripción de Hardware (HDL: Hardware Description Language) que describe el comportamiento de un circuito o sistema electrónico; cuyo circuito o sistema físico (real) puede ser implementado en el dispositivo respectivo.

Es utilizado principalmente, junto con Verilog, para implementar circuitos en FPGA's, CPLD's y en el campo de los ASIC's, siendo este diseño portable y reusable.

Cabe mencionar que las sentencias VHDL son inherentemente concurrentes, es decir, se ejecutan simultáneamente. Solamente las que se encuentren dentro de procesos, funciones o procedimientos son ejecutadas de manera secuencial.

Para realizar esta labor, utiliza las denominadas herramientas EDA (Electronic Design Automation) creadas especialmente para las tareas de síntesis, implementación y simulación de circuitos descritos. Algunas son ofrecidas dentro de los entornos de desarrollo de los fabricantes (Quartus II, MAXPLUS II, ISE)

mientras que otros son desarrollados y ofrecidos por compañías EDA especializadas como Leonardo Spectrum (sintetizador de Mentor Graphics), Synplify (sintetizador de Synplicity) y ModelSim (simulador de Mentor Graphics) [17].

2.6. Modelo teórico

El constante incremento de las capacidades de procesamiento requeridas para los procesos y necesidades actuales conlleva al desarrollo de nuevas teorías y tecnologías que puedan suplir las carencias provocadas por la falta de eficacia de las técnicas empleadas. Las redes neuronales son capaces de resolver gran parte de estos problemas mediante su implementación tanto en hardware como en software, pues son utilizadas en diversos campos como son el control, la industria, el reconocimiento de patrones, la medicina, los negocios y demás. Sin embargo, existen aplicaciones en las cuales necesariamente se requiere de una implementación en hardware para que dicha tarea pueda ser llevada a cabo.

Así, los dispositivos (hardware) donde implementemos dichas redes deben ocupar el área mínima posible con las mayores prestaciones posibles para garantizar una mejor eficiencia; consiguiendo mayor velocidad y confiabilidad al utilizar hardware digital dedicado.

Dentro de dichos dispositivos mencionados, destacan los FPGA's, pues pueden ser configurados según la necesidad y los requerimientos del sistema; y también pueden ser reconfigurados, permitiendo una gran flexibilidad tanto en el diseño como en la implementación. Para configurar el FPGA utilizaremos los llamados lenguajes de descripción de hardware (HDL), y específicamente VHDL. Un posterior avance sería el desarrollo de un circuito integrado de aplicación específica (ASIC) mediante la tecnología VLSI, pues el diseño del sistema en un FPGA es un paso previo para este fin. De esta forma, se alcanzaría un chip con mejores prestaciones en aspectos como velocidad, área ocupada y consumo de potencia.

2.6.1. Definiciones Operativas

El presente diseño de una red neuronal en un FPGA debe ser evaluado para medir su desempeño y verificar su funcionamiento. De esta forma, los criterios más

importantes son los referidos a la efectividad del sistema y a la arquitectura planteada.

- **Estructura de la red neuronal**

Indica el número de capas utilizadas en la red neuronal propuesta, así como el número de neuronas en cada una de ellas. Es de importancia porque es la base de la arquitectura lógica a plantear, además de ser la que va a resolver el problema planteado.

- **Entrenamiento de la red neuronal**

Corresponde a la cantidad de datos (base de datos) con la cual se va a entrenar a la red neuronal. Este es un factor importante pues, dado que la red requiere conocimientos, a una mayor cantidad de datos de entrenamiento (mejor entrenamiento) el sistema se comportará de una manera más eficiente y producirá menos errores. Caso contrario, la red no se comportará según lo previsto.

- **Recursos utilizados**

Corresponde a la cantidad de compuertas, flip-flops, multiplicadores, memorias (recursos lógicos) utilizados por el diseño. Es una forma indirecta de medir el área ocupada en el FPGA. Según este parámetro escogeremos al FPGA. En el diseño se procurará que sea el mínimo posible para reducir el área utilizada.

- **Frecuencia de trabajo**

Es la frecuencia a la cual va a trabajar el sistema diseñado. Es una medida de la velocidad del sistema.

- **Tiempo de respuesta del sistema**

Indica el tiempo que se demora el sistema en cumplir con la labor asignada y otorgar un valor válido a la salida.

- **Porcentaje de Error**

Corresponde a la cantidad de errores que genera el sistema respecto a la cantidad de salidas que genera en un periodo de operación. Es un indicador de la calidad del sistema, razón por la cual debe ser el menor posible.

- **Portabilidad del diseño**

Indica si el diseño realizado con VHDL es transparente o no a los diferentes FPGA's. Esto quiere decir que un código será portable si puede ser sintetizado en cualquier FPGA. Significa que es independiente del dispositivo, pues no ha utilizado recursos propios del FPGA, los cuales pueden diferir entre las diferentes familias y fabricantes. De esta forma se logra que el sistema pueda ser implementado en cualquier FPGA que cumpla con los requerimientos mínimos del mismo sistema.

CAPÍTULO 3

PLANTEAMIENTOS PARA EL DISEÑO DE LA RED NEURONAL PERCEPTRON MULTICAPA

3.1. Hipótesis de la investigación

3.1.1. Hipótesis principal

Dado que los procesos y problemas actuales son cada vez más complejos y que buena parte de estos pueden ser resueltos con aplicaciones basadas en redes neuronales tanto en hardware como en software; entonces, el diseño de la arquitectura de una red neuronal Perceptron Multicapa sobre un FPGA permite el desarrollo de un sistema modular en hardware de propósito general que puede ser configurado para dar solución a tareas específicas; y en este caso, aplicarla en un sistema de reconocimiento de caracteres.

3.1.2. Hipótesis secundarias

- Contar con una implementación en hardware de una red neuronal permite liberar de un procesamiento intenso al procesador (CPU) y utilizarlo en otras tareas.
- La red neuronal multicapa (MLP) es una arquitectura de propósito general pues puede ser configurada para cumplir diferentes tareas según sean requeridas.
- Dadas las características propias de las redes neuronales, el desarrollo de una red neuronal sobre un FPGA permite obtener una arquitectura flexible y adaptable a distintas aplicaciones.

3.2. Objetivos de la investigación

3.2.1. Objetivo general

Diseñar una arquitectura para una red neuronal Perceptron Multicapa en un FPGA y verificar su funcionamiento mediante su implementación en un sistema de reconocimiento de caracteres.

3.2.2. Objetivos específicos

- Diseñar la red neuronal artificial para el reconocimiento de caracteres en un software como MATLAB con ayuda de su herramienta para redes neuronales (Neural Networks Toolbox) y entrenarla mediante el algoritmo Backpropagation.
- Diseño de la arquitectura de una neurona artificial, describirla en el FPGA y verificar su funcionamiento.
- Buscar en todo momento la optimización de los recursos del FPGA, así como diseños modulares de cada componente. Además, que el diseño global sea lo más portable posible.

3.3. Metodología de la investigación

El diseño de la arquitectura se desarrollará siguiendo el siguiente esquema, el cual muestra la secuencia de pasos para la obtención del sistema.

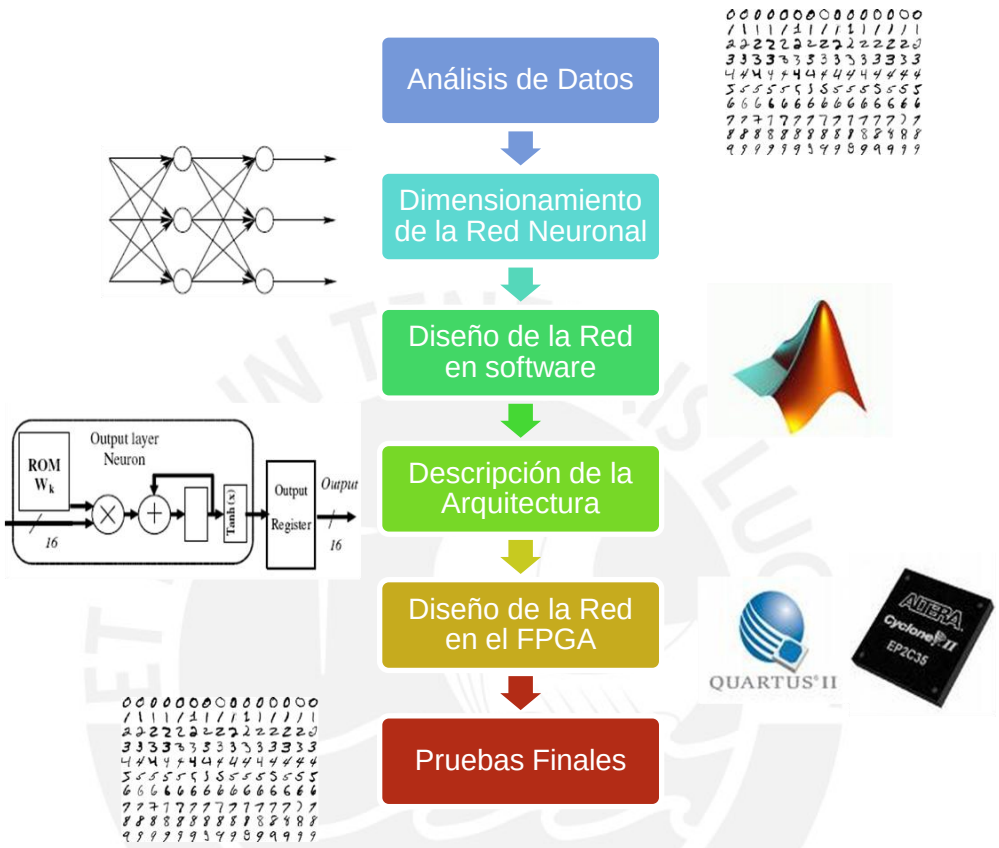


Figura 6. Metodología de diseño.

El primer paso consiste en analizar los datos que son objeto de estudio. Así podremos obtener información necesaria que será utilizada en la siguiente etapa. Aquí, el tipo de imagen, así como su tamaño, dimensiones y otras características son extraídos, definidos y considerados dentro de los parámetros de diseño.

Con la información obtenida y con los parámetros definidos se procede a dimensionar la red neuronal. Las dimensiones que pueden ser definidas son la cantidad de entradas y la cantidad de salidas de la red, puesto que se infieren directamente de los requerimientos del sistema.

A partir de este punto comienza el diseño. Al tratarse de una red MLP, las incógnitas a resolver son la cantidad de capas ocultas y la cantidad de neuronas de cada capa oculta. El Teorema de Aproximación Universal (Cybenko, 1989;

Funahashi, 1989; Hornik et al., 1989) nos otorga el fundamento matemático para la aproximación a realizar:

Sea $\varphi(\cdot)$ *una función continua no constante, acotada y monótonamente creciente. Sea* I_p *el espacio p-dimensional unitario* $[0,1]^p$ *y* $C(I_p)$ *el espacio de funciones continuas sobre* I_p*. Entonces, dada cualquier función* $f \in C(I_p)$ *y* $\varepsilon > 0$ *existe un entero M y un grupo de constantes reales* α_i, θ_i *y* w_{ij} *donde* $i \in [1, M]$ *y* $j \in [1, p]$ *con los que podemos definir*

$$F(x_1,..., x_p) = \sum_{i=1}^{M} \alpha_i \varphi\left(\sum_{j=1}^{p} w_{ij} x_j - \theta_i \right) \quad (1)$$

como una aproximación de la función $f(\cdot)$*, tal que*

$$\left| F(x_1,..., x_p) - f(x_1,..., x_p) \right| < \varepsilon \quad (2)$$

para todo $\{x_1,..., x_p\} \in I_p$ [6].

Esta definición se ajusta perfectamente a una red MLP con una capa oculta, pues cuenta con *p* entradas, *M* neuronas en la capa oculta, pesos sinápticos w_{ij} y umbrales b_i, una función de activación sigmoide que cumple con lo anterior y la salida que es función lineal de las salidas de las neuronas ocultas. Hay que tener en cuenta que el teorema no dice que utilizar una única capa oculta sea lo más óptimo, pero nos garantiza la existencia de una solución [6].

Además, la cantidad de neuronas en la capa oculta debe ser escogida según cumpla los requerimientos establecidos. Esto quiere decir que el diseño puede empezar con dos neuronas en la capa oculta, generar la red neuronal y verificar si cumple lo determinado. Caso contrario, se aumenta la cantidad de neuronas hasta cumplir con los requerimientos. Es importante mencionar que se debe buscar, en lo posible, el menor número de neuronas para esta capa, pues de no ser así estaríamos perdiendo la capacidad de generalización de la red neuronal, pues el sistema estaría aprendiendo también el ruido presente en las imágenes de entrenamiento [6].

De esta forma, se empieza el diseño escogiendo una sola capa oculta con una cantidad de neuronas intermedia entre el número de entradas y el número de salidas.

Tras obtener un diseño satisfactorio, se describe la arquitectura para la red diseñada. En esta etapa hay que tener en cuenta las dimensiones de la red, de sus pesos sinápticos, de la función de activación utilizada y de los formatos utilizados tanto en la entrada como en la salida. Un factor determinante es escoger la representación numérica adecuada y su resolución pues de ello depende el diseño de los posteriores bloques del sistema.

Teniendo la arquitectura completa, se procede a su diseño para su implementación en el FPGA. La arquitectura se describe utilizando VHDL y una herramienta EDA que realice la síntesis del circuito y su posterior implementación en el dispositivo. Para esta labor, se deben considerar los objetivos del diseño en cuanto se quiera un diseño portable u optimizado al dispositivo, o específico a los requerimientos o genérico. Cada uno de los módulos diseñados debe ser simulado para verificar su funcionamiento y buscar garantizar que no va a generar errores al momento de unir el sistema completo. Adicionalmente, es de esperar que en cualquiera de los casos exista un balance adecuado entre los recursos lógicos utilizados y la frecuencia de trabajo del sistema.

Por último, se realizan las simulaciones finales del sistema y se procede a su implementación en el FPGA. En este punto se realizan pruebas adicionales para comprobar que la red neuronal cumple con lo esperado.

Cabe mencionar que se realiza el diseño de la red en software debido a que el aprendizaje no se realizará en hardware, pues el FPGA será destinado solo a la tarea de reconocimiento mas no se implementará la etapa de entrenamiento. Es por este motivo que el entrenamiento se realiza en software y se extraen los resultados obtenidos del proceso, los pesos sinápticos y la cantidad de neuronas y capas ocultas.

3.4. Análisis del sistema y determinación de requerimientos

La red neuronal va a ser entrenada y diseñada para reconocer dígitos manuscritos por lo cual requiere de 10 salidas, cada una de las cuales identificará a un digito en particular.

La etapa de entrenamiento se realizará con la base de datos modificada del Instituto Nacional de Estándares y Tecnología, MNIST (Modified National Institute of Standards and Technology database), la cual consta de 30000 ejemplos de la base

de datos especial 1 del NIST (SD-1) y 30000 de la base de datos especial 3 del NIST (SD-3) para el entrenamiento y de 5000 de cada uno para el test; es decir, de 60000 imágenes de dígitos manuscritos para la etapa de entrenamiento y 10000 para la etapa de test. La base de datos SD-3 es más fácil de reconocer que la SD-1 debido a que la primera fue realizada a partir de empleados (Census Bureau employees) y la segunda a partir de estudiantes de colegio [19].

Dichas imágenes se encuentran en escala de grises y además, están normalizadas en un tamaño de 20x20 pixeles y el centro de masa de cada una está centrado en un campo de 28x28 pixeles [19].

Según esto, el sistema recibe de entrada una imagen de 28x28 pixeles, por lo cual, los datos serán umbralizados y agrupados para disminuir la cantidad de entradas a la red. Así, gracias a la umbralización, disminuimos la cantidad de datos de 784 pixeles a 784 bits.

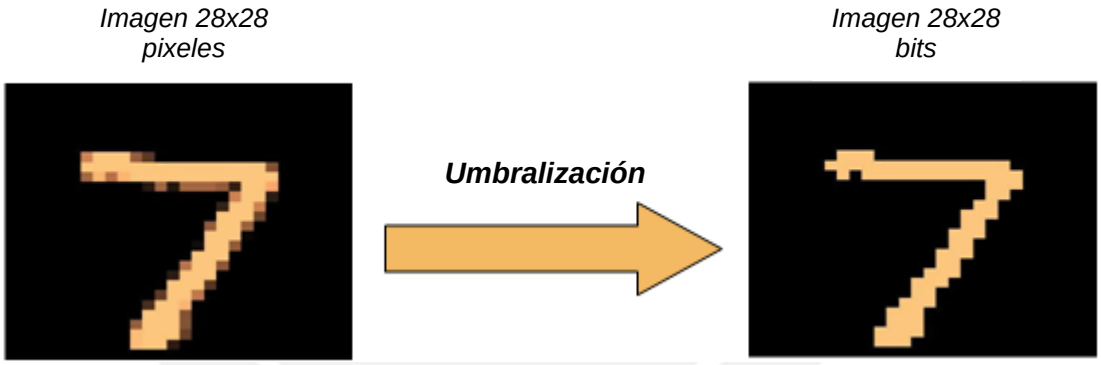


Figura 7. Proceso de umbralización.

La red neuronal a utilizar es una red MLP, y el algoritmo de entrenamiento es el de backpropagation; por lo cual la función de activación requerida por el algoritmo es una función diferenciable, pues se utiliza su gradiente para la actualización de los pesos sinápticos.

La magnitud de los pesos obtenidos en este proceso determina la cantidad de bits así como el formato de la representación numérica. También, la cantidad de neuronas y la cantidad de capas ocultas tienen implicancia en la cantidad de recursos lógicos a utilizar.

Las salidas de la red neuronal serán validadas comparando los resultados con los rangos esperados para un correcto reconocimiento. De esta forma se eleva el porcentaje de eficiencia del sistema al incorporar una etapa de verificación.

Finalmente, el sistema debe tener una confiabilidad aceptable, pues realiza una tarea crítica dentro de un OCR. De este modo, se espera una eficiencia cercana al 90% en la etapa de test.

Por lo tanto, los parámetros definidos para el sistema son los siguientes:

- Señal de entrada del sistema: Imagen digital de un dígito manuscrito de 28x28 pixeles en escala de grises.
- Señal de salida del sistema: número reconocido y una señal de validación, la cual indica si el número otorgado es válido o no.
- Cantidad de entradas de la red neuronal: se determinará con el proceso de diseño de la red en software.
- Cantidad de salidas de la red neuronal: se tendrán 10 salidas.
- Cantidad de capas ocultas: una capa oculta.
- Cantidad de neuronas de la capa oculta: se determinará con el proceso de diseño de la red en software.
- Función de activación de la red: se utilizará en todas las capas la función Sigmoide como función de activación, al tratarse de la más utilizada para este tipo de red neuronal.
- Umbralización: este proceso se realizará con un umbral de 100 para convertir los pixeles mayores o iguales que el umbral a un '1' lógico y los menores que el umbral a un '0' lógico.

3.5. Consideraciones para el diseño

El sistema desarrollado forma parte de un OCR, y específicamente conforma la etapa de reconocimiento del carácter. Así, se tendrá como entrada a esta etapa la imagen extraída del documento mediante la segmentación, así como centrada y normalizada al tamaño de 28x28 pixeles en escala de grises. Cualquier paso previo se considera ya realizado y no forma parte del desarrollo de la presente tesis.

Como ya se mencionó, la etapa de entrenamiento de la red neuronal no se implementará en el dispositivo. De esta forma se tendrá un hardware dedicado con capacidad de reprogramación, con lo cual, basta con re-entrenar la red en software para que cumpla con la nueva tarea, obtener los parámetros requeridos por el sistema e ingresarlos en la descripción del circuito para su posterior síntesis e implementación en el FPGA.

Para el entrenamiento, se utiliza la herramienta de redes neuronales del software Matlab 7.1 de Mathworks (Neural Networks Toolbox). Se desarrollarán los programas necesarios para la umbralización de la imagen, para la extracción de algunas características, para obtener la red neuronal, su entrenamiento con la base de datos de entrenamiento del MNIST y su posterior etapa de test con la base de datos de test del MNIST. Adicionalmente, se generarán programas para elaborar los archivos necesarios (*.vhd) que contengan la información de los pesos sinápticos y umbrales obtenidos tras el proceso de aprendizaje. Así, se tendrá una metodología automatizada para que a partir de los resultados obtenidos se tengan los archivos necesarios a incluir en el diseño en hardware de la red neuronal.

El diseño en software se seguirá hasta obtener una eficiencia cercana al 90% teniendo como entrada la base de datos de test del MNIST. Dado que el objetivo principal de la tesis es el de obtener una arquitectura para una red neuronal, una eficiencia de 90% es aceptable para verificar el buen funcionamiento del sistema diseñado. Además, hay que indicar que la base de datos del MNIST es una muestra muy representativa del universo y que con dicho porcentaje no se pierde la visión global de obtener un sistema confiable.

El diseño en hardware se realizará mediante la descripción del mismo utilizando el lenguaje de descripción VHDL en la herramienta EDA Quartus II 7.1 de Altera, al realizarse el diseño en un FPGA Cyclone II de la misma compañía. La descripción será modular y genérica para facilitar el diseño funcional y para que la red diseñada sea fácilmente adaptable para que cumpla en algún futuro una tarea diferente a la actual. Dado que el sistema se basa en suma de productos y acoplo de etapas, las operaciones de multiplicación y división aparecen con frecuencia; por lo tanto, el diseño utilizará recursos dedicados del dispositivo, como son multiplicadores embebidos, lo cual le quita portabilidad al diseño. Si bien esto puede parecer perjudicial, es necesario para la optimización del sistema, pues las operaciones de multiplicación y división son las más críticas y ocasionan los mayores retardos. Pero, el hecho de utilizar recursos dedicados del FPGA no impide que el sistema se implemente en otra familia de dispositivos, pues las herramientas EDA infieren estos bloques dedicados tras interpretar la descripción realizada. Así, solamente se pierde un grado de portabilidad mas no se vuelve no portable.

Por último, hay que mencionar que existe una dependencia entre el tamaño (dimensiones) y requerimientos de la red neuronal y el FPGA a utilizar. Esto se debe a que el dispositivo tiene limitaciones en cuanto a la cantidad de recursos

lógicos que posee y a la cantidad y variedad de bloques dedicados que optimizan el diseño en cuanto a rapidez y área ocupada. Así, el dispositivo debe ser elegido según la aplicación y en este caso según la red neuronal que va a cobijar. Hay que mencionar que las limitaciones de espacio se deben a que en la presente tesis no se considera la característica propia del FPGA de poder reconfigurarse. Sin embargo, la característica que si se aprovechará será la del paralelismo que posee, al implementar todas las capas de la red neuronal y realizar los cálculos lo más paralelo posible.

CAPÍTULO 4

DISEÑO DE LA ARQUITECTURA PARA LA RED NEURONAL PERCEPTRON MULTICAPA

El presente capítulo muestra el proceso de diseño de la red neuronal, lo cual implica la determinación de sus dimensiones, sus parámetros, y la arquitectura en hardware que permita implementar la red neuronal propuesta en el FPGA.

Siguiendo la metodología planteada, se procede al diseño de la red neuronal en software para obtener y extraer sus dimensiones y parámetros. De esta forma se encuentran el número de entradas, el número de neuronas en la capa oculta, y los valores de los pesos sinápticos y umbrales de la red neuronal.

Posteriormente, considerando estos valores encontrados, se diseña la arquitectura digital en base a bloques y componentes modulares y genéricos que permiten una adecuada interacción entre ellos. Así, el diseño global se complementa con el proceso de aprendizaje anterior incorporando los pesos sinápticos y los umbrales de cada neurona.

4.1. Diseño de la red neuronal mediante software

Para este proceso se utilizó le herramienta de redes neuronales del software Matlab 7.1 (Neural networks toolbox); con la cual se realizó todas las etapas de este proceso: dimensionamiento, entrenamiento y validación de la red neuronal; y la extracción de los pesos sinápticos y umbrales encontrados.

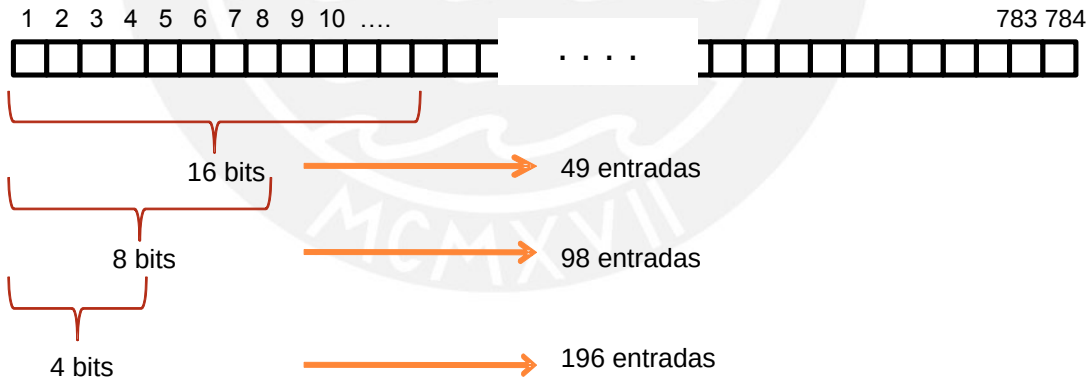
Si bien el objetivo del presente trabajo es obtener una red neuronal en hardware, este paso es de suma importancia pues nos permite escoger la mejor red neuronal para nuestro propósito, así como encontrar sus parámetros y dimensiones para incorporarlos al diseño en hardware y establecer con ellos la representación numérica adecuada y la arquitectura del circuito.

4.1.1. Desarrollo y entrenamiento de la red neuronal.

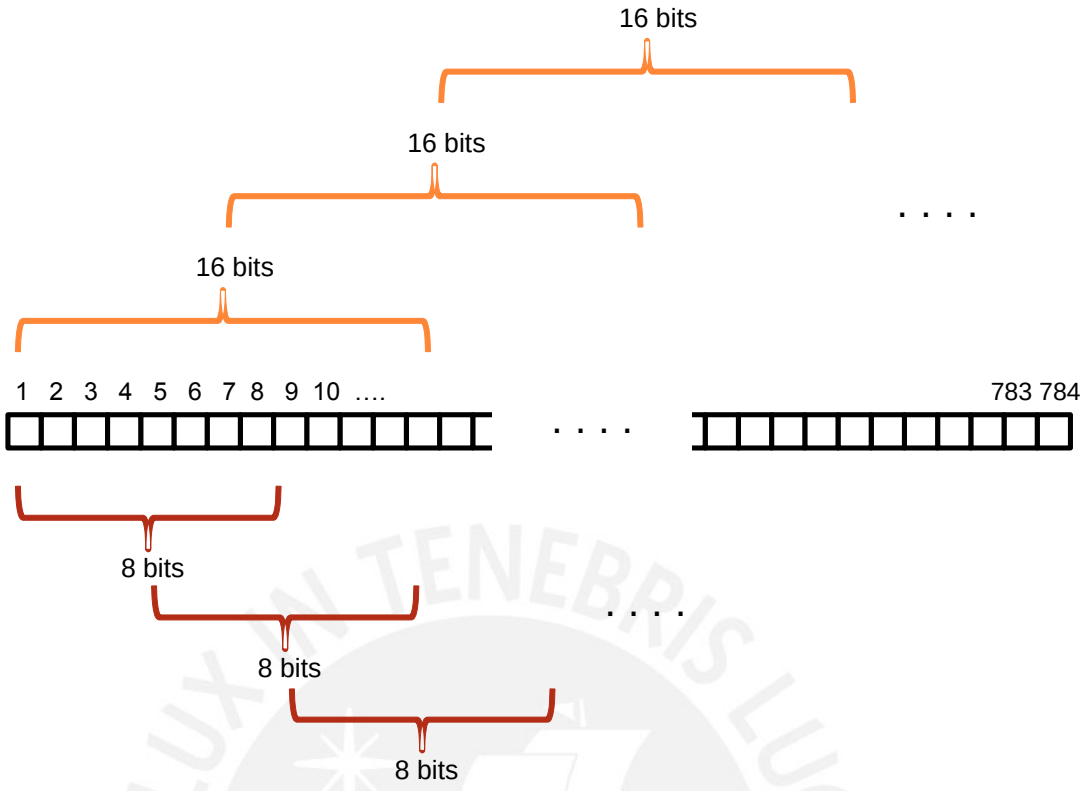
El primer paso consiste en escoger la cantidad de entradas en la capa de entrada de la red neuronal. Para ello, sabemos que la entrada de la red es una imagen de 28x28 bits, es decir, un total de 784 bits.

Con el objetivo de disminuir el número de entradas y la cantidad de datos con la que debe trabajar la red neuronal, los bits de la imagen de entrada son agrupados. Este proceso es denominado “extracción de características”, y existen diversas formas de agrupar dichos bits. Debido a esto, se desarrollaron diferentes redes neuronales para encontrar la forma de agrupación óptima en cuanto a cantidad de entradas, cantidad de neuronas, valor absoluto de los datos y eficiencia.

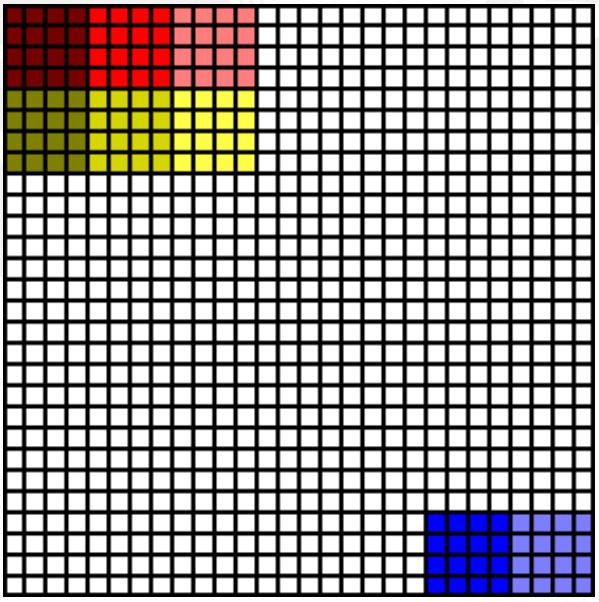
La figura 8 muestra algunas formas de agrupación de bits, donde en (a) los bits de la imagen se juntan para formar un solo vector unidimensional del cual se extraen palabras binarias de la cantidad de bits mostrada; en (b) se procede muy similar a lo anterior, solo que las palabras formadas se sobreponen para contar con una mayor cantidad de datos; mientras que en (c) la forma de agrupación cambia y se divide la imagen en imágenes más pequeñas, sumando los bits de estas sub-imágenes para ingresarlas a la red.



(a)



(b)



(c)

Figura 8. Extracción de características, formas de agrupación de los bits de la imagen de entrada: (a) palabras binarias de un vector unidimensional, (b) palabras binarias sobrepuestas de un vector unidimensional, (c) suma de bits de sub-imágenes de menor tamaño.

Con estas formas mencionadas, se procedió a entrenar diversos arreglos de redes neuronales mediante el algoritmo de entrenamiento backpropagation. Todo esto para encontrar la red adecuada para nuestro propósito.

Antes de iniciar el aprendizaje, la base de datos fue adaptada a los requerimientos de cada forma de agrupación. Así, se crearon diversos scripts que lograran lo antes mencionado. También, el archivo de respuestas deseadas fue adaptado para que cumpla con lo estipulado de las 10 neuronas en la capa de salida.

Luego, se procedió al entrenamiento de las redes neuronales. Los parámetros del proceso de entrenamiento fueron escogidos según las recomendaciones hechas por los desarrolladores de la herramienta en software [20]. De esta manera, los únicos factores importantes a decidir son la cantidad de veces que los datos pasan por la red para que los pesos y umbrales sean modificados (epochs) y la función de entrenamiento, que determina el algoritmo a seguir para el proceso y el tiempo de convergencia de la red neuronal según los parámetros establecidos para el entrenamiento.

Según [20], para una aplicación de reconocimiento, la mejor opción como función de entrenamiento es el algoritmo “Resilent Backpropagation”, cuya función de entrenamiento es llamada “trainrp”. Un factor importante en el proceso, y que se relaciona directamente con la eficiencia del sistema es la medida de la función de desempeño (performance function). En este caso, dicha función es la del error cuadrático medio (mse) y nos indicará que cerca se está de conseguir un error promedio de cero.

La primera opción que se tuvo en cuenta fue la de formar palabras binarias que no se sobrepongan. La tabla 2 y la figura 9 muestran los resultados obtenidos para 5 redes neuronales. En las tablas, el parámetro performance hace referencia al error cuadrático medio (mse).

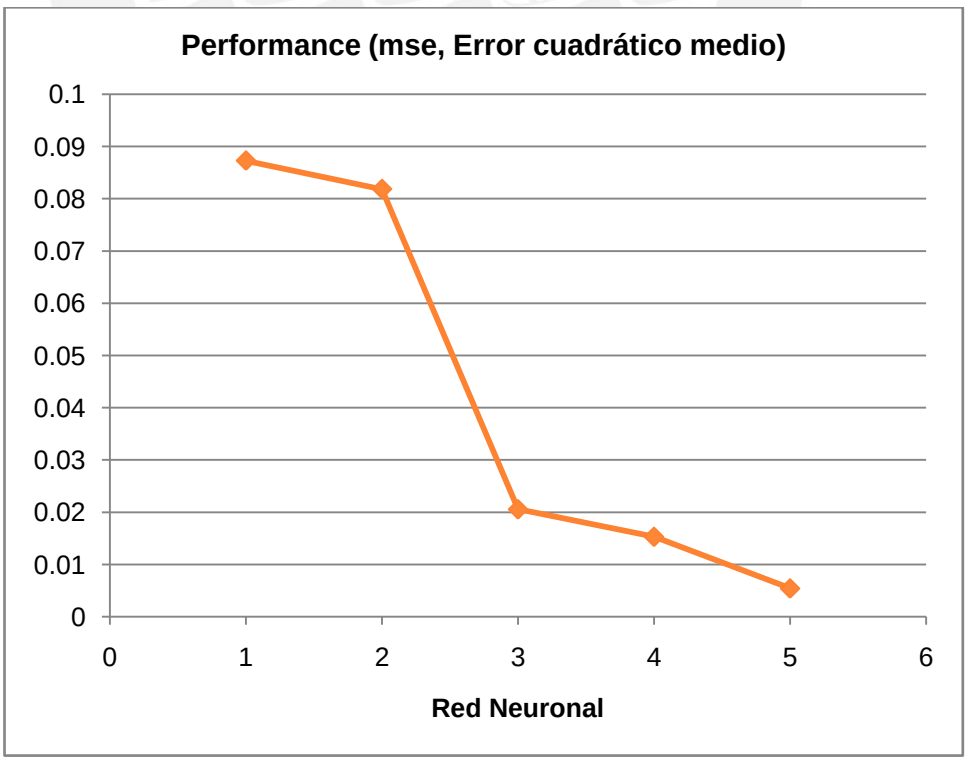
En ellas se observa que si bien alcanzan un error bajo en algunos casos, las magnitudes absolutas de los pesos sinápticos son muy elevadas, lo que dificultaría el diseño de la arquitectura al requerir una mayor cantidad de bits. Además, la cantidad de entradas es aún elevada si escogemos trabajar con el error más bajo.

Tabla 2.a. Características de las redes neuronales [Elaboración propia].

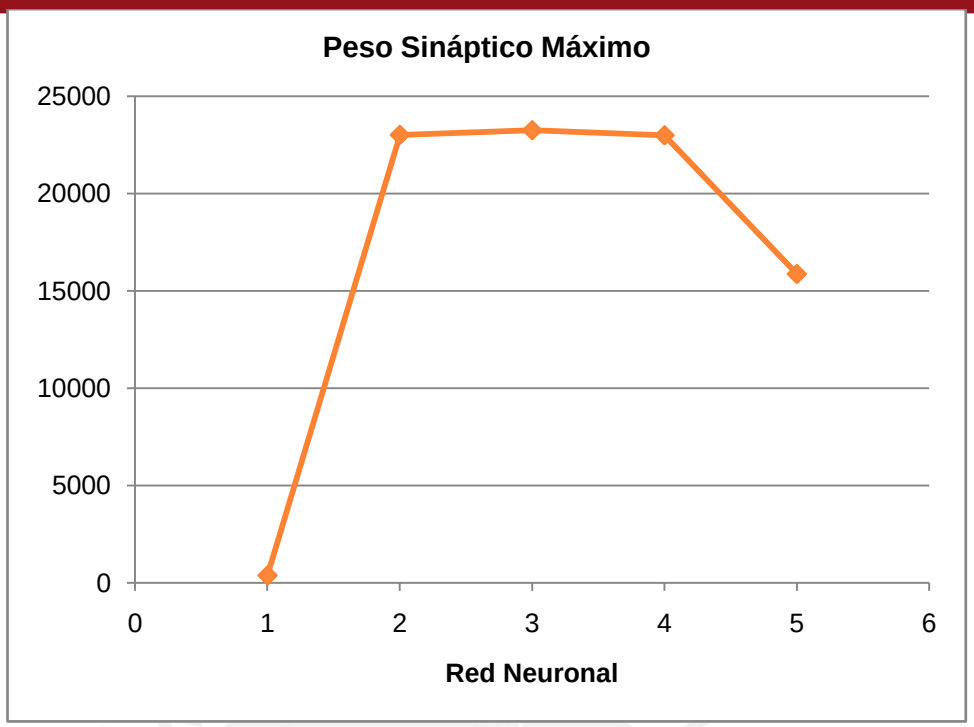
Red Neuronal	#bits / entrada	Neuronas en la capa de entrada	Neuronas en la capa oculta	Neuronas en la capa de salida
1	16	49	25	10
2	16	49	40	10
3	8	98	25	10
4	8	98	50	10
5	4	196	50	10

Tabla 2.b. Entrenamiento [Elaboración propia].

Red Neuronal	Epochs	Performance (mse)	Peso máximo
1	500	0.0872867	380
2	500	0.0818516	23020
3	500	0.0205614	23260
4	500	0.0153069	23000
5	500	0.00543357	15870



(a)



(b)

Figura 9. Error cuadrático medio (a) y peso sináptico máximo (b) por red neuronal.

Se llegó a la conclusión que este tipo de agrupación no es la más adecuada para esta aplicación puesto que los pesos que se obtienen son muy elevados para implementar la red neuronal completa dentro del FPGA.

Como segundo caso, se formaron palabras binarias sobrepuestas hasta la mitad de la anterior, según lo muestra la figura 8. Además, tras entrenar una red neuronal, se observó que la magnitud absoluta máxima de los pesos sinápticos no variaba mucho, por lo cual se pensó en añadir una capa oculta adicional para corregir este inconveniente.

Los resultados de este proceso se muestran en la tabla 3. El número de neuronas desarrolladas en este caso fue mayor puesto que se requería una mayor cantidad de datos al tratarse de dos más variables a considerar dentro del análisis.

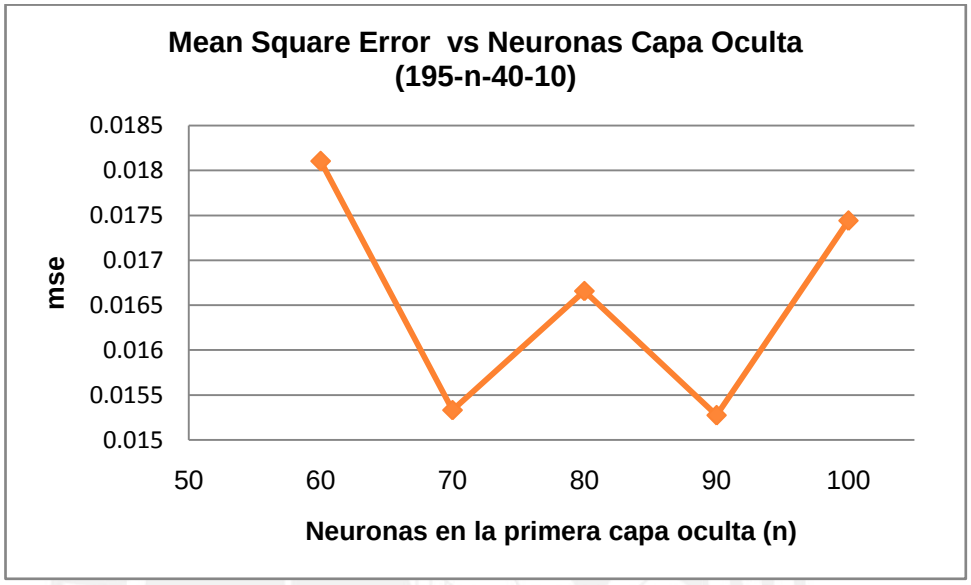
Tabla 3.a. Características de las redes neuronales [Elaboración propia].

Red Neuronal	#bits / entrada	Neuronas en la capa de entrada	Neuronas en la capa oculta		Neuronas en la capa de salida
6	8	195	50		10
7	8	98	50	20	10
8	8	195	60	20	10
9	8	195	50	25	10
10	8	195	100	40	10
11	8	195	80	40	10
12	8	195	60	40	10
13	8	195	80	20	10
14	8	195	70	40	10
15	8	195	90	40	10
16	8	195	70	35	10
17	8	195	70	50	10
18	8	195	70	60	10
19	8	195	70	30	10
20	8	195	80	50	10
21	8	195	80	60	10
22	8	195	80	30	10

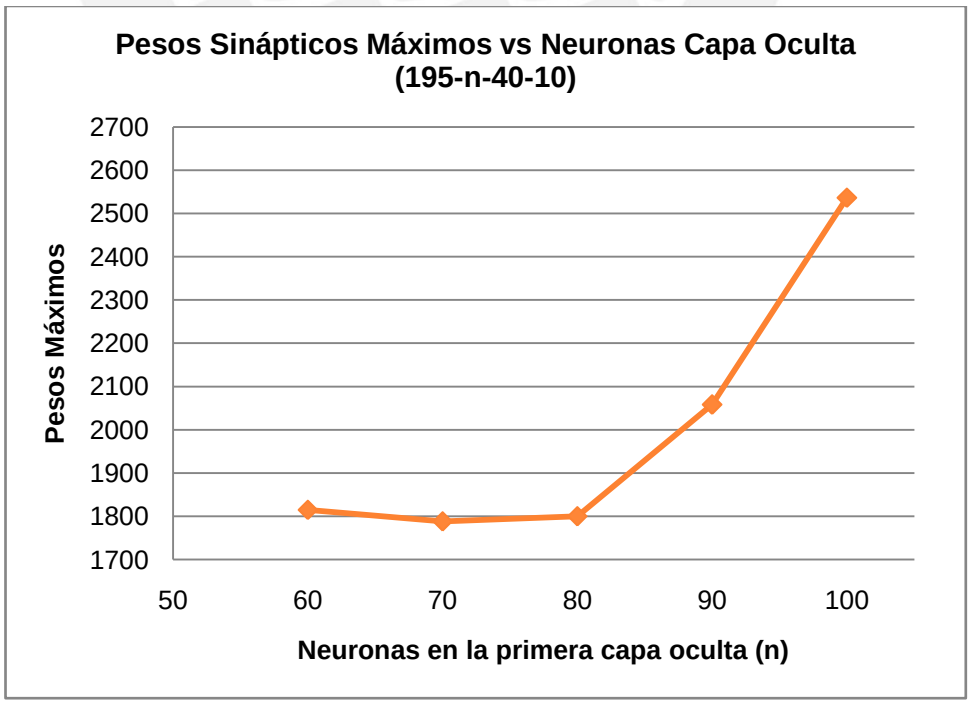
Tabla 3.b. Entrenamiento [Elaboración propia].

Red Neuronal	Epochs	Performance (mse)	Peso máximo
6	500	0.0140425	19489
7	500	0.0130687	23500
8	100	0.0179081	1800
9	100	0.0181599	2140
10	100	0.017442	2536
11	100	0.0166598	1800
12	100	0.0181039	1815
13	100	0.0186173	2056
14	100	0.0153329	1788
15	100	0.0152742	2058
16	100	0.0160615	2264
17	100	0.0188552	3016
18	100	0.0172501	2400
19	100	0.0176996	2311
20	100	0.016802	1982
21	100	0.0184598	2058
22	100	0.0183491	1583

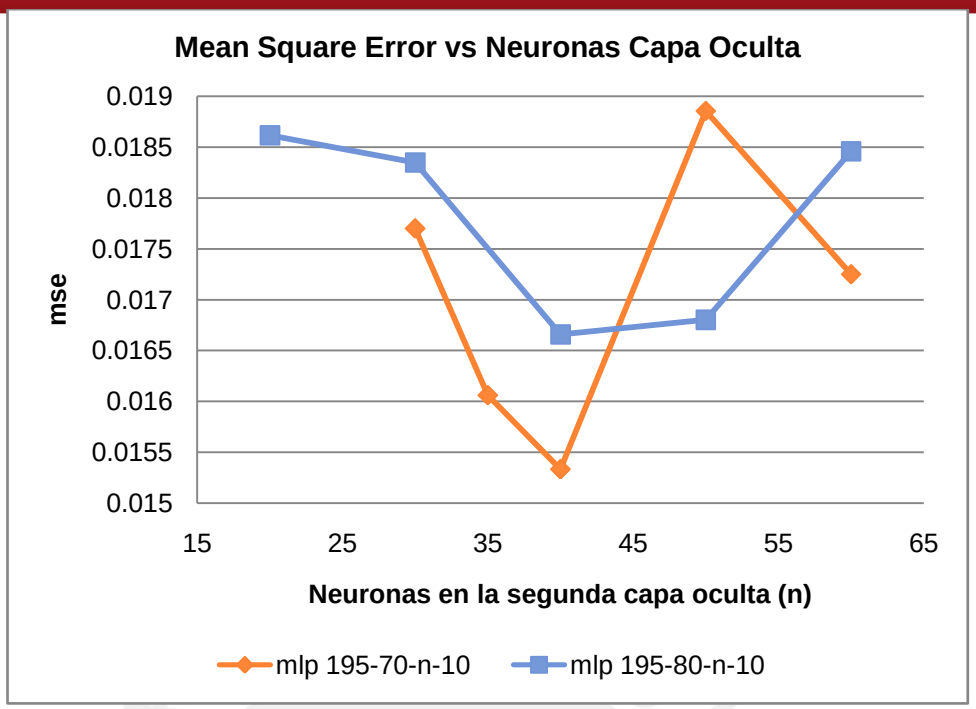
De la tabla presentada se extraen los siguientes gráficos mostrados en la figura 10. En ellos se observa la variación de los parámetros como son la magnitud absoluta de los pesos sinápticos y el error cuadrático medio frente a los cambios en la cantidad de neuronas en las capas ocultas.



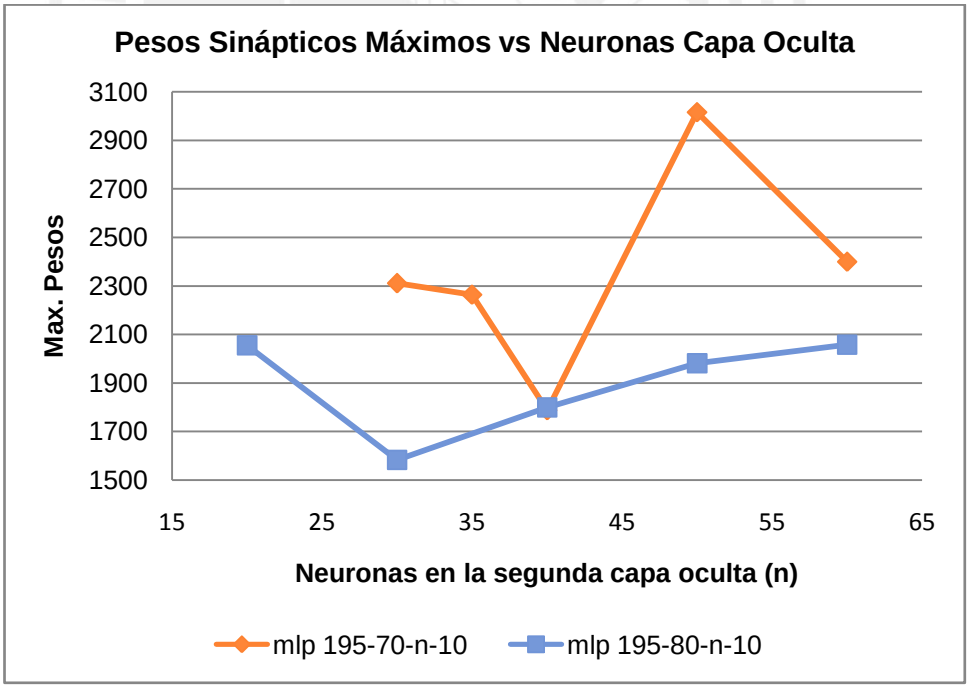
(a)



(b)



(c)



(d)

Figura 10. Variaciones del error cuadrático medio y del peso sináptico máximo frente a cambios en la cantidad de neuronas en la primera capa oculta (a) y (b), y frente a cambios en la segunda capa oculta (c) y (d).

Podemos observar que para valores similares del error cuadrático medio a los de una sola capa oculta se tiene pesos máximos de mucha menor magnitud. Esto significa un avance, sin embargo, el costo esta en el aumento de neuronas a utilizar, así como de la cantidad de datos a procesar por la red neuronal.

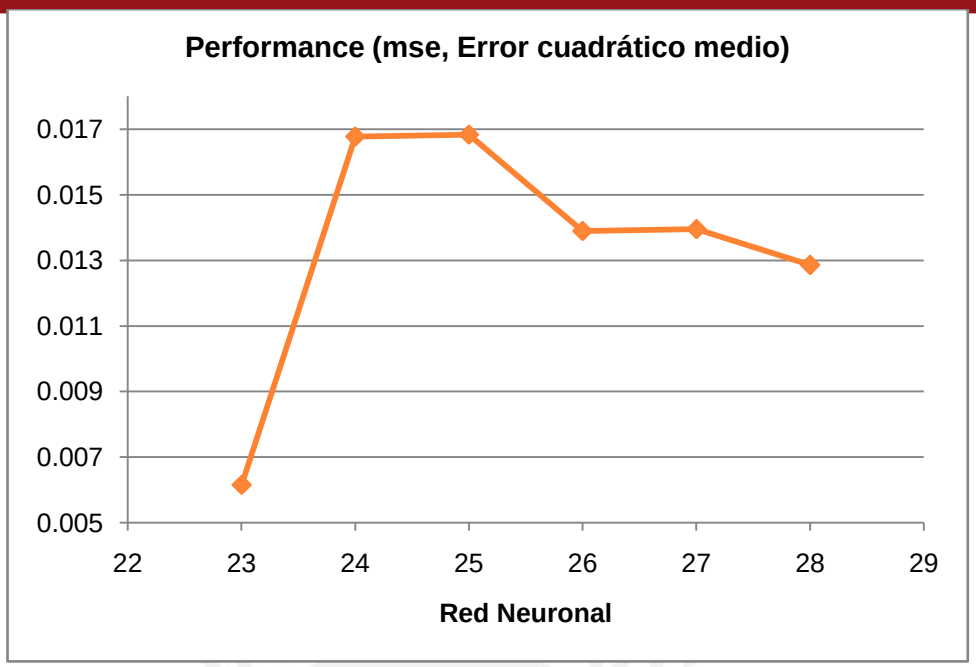
Por último, se procedió a entrenar redes neuronales con la siguiente forma de agrupación de bits. Se obtuvieron los siguientes resultados, los cuales son mostrados en la siguiente tabla 4 y en la figura 11.

Tabla 4.a. Características de las redes neuronales [Elaboración propia].

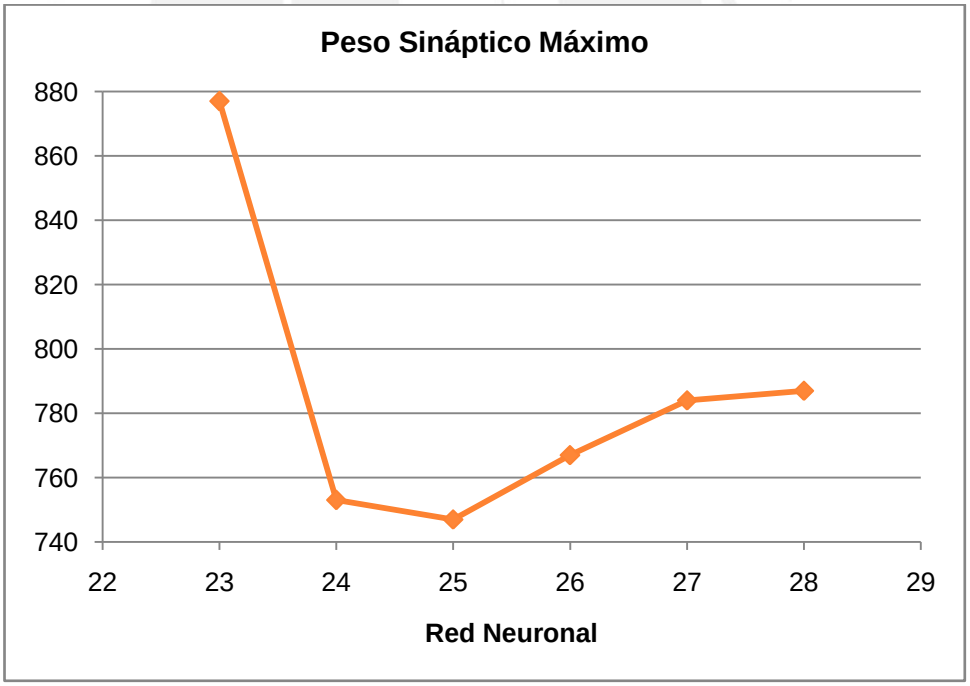
Red Neuronal	#bits / entrada	Neuronas en la capa de entrada	Neuronas en la capa oculta	Neuronas en la capa de salida
23	4	196	50	10
24	16	49	15	10
25	16	49	20	10
26	16	49	25	10
27	16	49	30	10
28	16	49	35	10

Tabla 4.b. Entrenamiento [Elaboración propia].

Red Neuronal	Epochs	Performance (mse)	Peso máximo
23	100	0.00615423	877
24	100	0.0167706	753
25	100	0.01682905	747
26	100	0.013892	767
27	100	0.0139489	784
28	100	0.0128555	787



(a)



(b)

Figura 11. Error cuadrático medio (a) y peso sináptico máximo (b) por red neuronal.

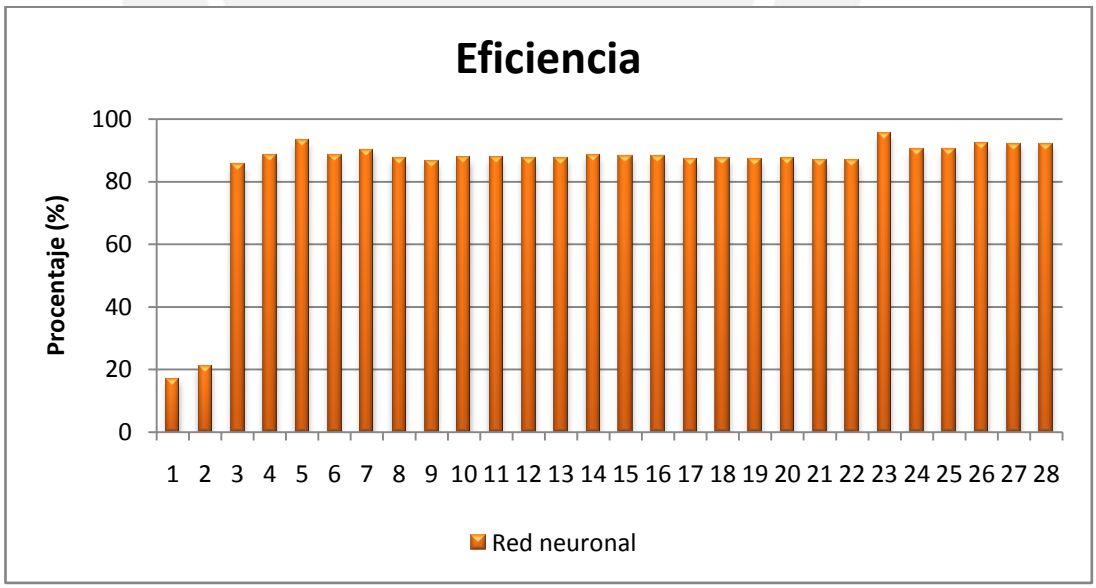
De las graficas se observa que la magnitud absoluta de los pesos sinápticos ha disminuido considerablemente, manteniéndose en un rango aceptable para su implementación en hardware. Además, estos valores no presentan mucha variación si consideramos que pueden ser representados con la misma cantidad de bits.

En cuanto al error cuadrático medio, vemos que la mejor prestación se consigue con un arreglo de 4 bits, pero ello nos conlleva a trabajar con una mayor cantidad de datos y neuronas.

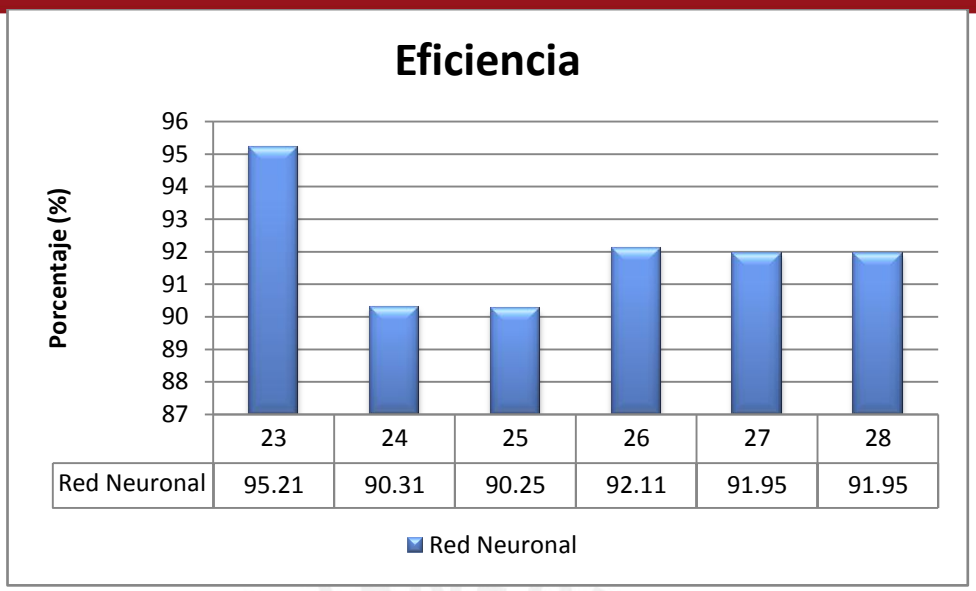
De esta forma, la arquitectura más adecuada para esta aplicación consiste de una agrupación de bits en sub-imágenes de 4x4 y consta del siguiente arreglo: 49-*n*-10, donde *n* debe definirse según la eficiencia que presenten frente a la base de datos de test; puesto que el error cuadrático medio solamente es un parámetro referencial.

4.1.2. Pruebas y validación de la red neuronal.

Para esta etapa se utilizó la base de datos de test del MNIST, con la cual se encontraron las eficiencias respectivas a las redes neuronales estudiadas. La figura 12 muestra estos resultados.



(a)



(b)

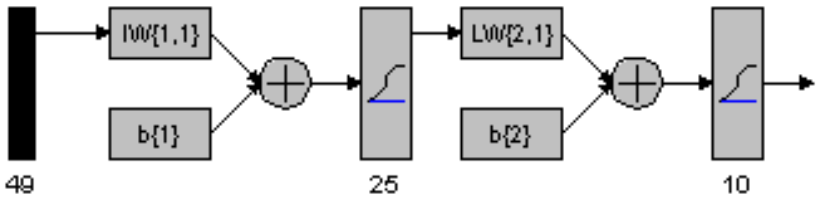
Figura 12. Eficiencia de Redes Neuronales expresadas en porcentaje: (a) todas las redes hechas, (b) redes con agrupación en sub-imágenes.

Estas eficiencias están referidas a las mencionadas en las tablas 2, 3 y 4; en la cual cada número de la leyenda indica una red neuronal específica referida en cada tabla. Asimismo, se hace énfasis en las redes de la tabla 4 pues presentan mejores resultados y un mejor desempeño según el propósito de la aplicación.

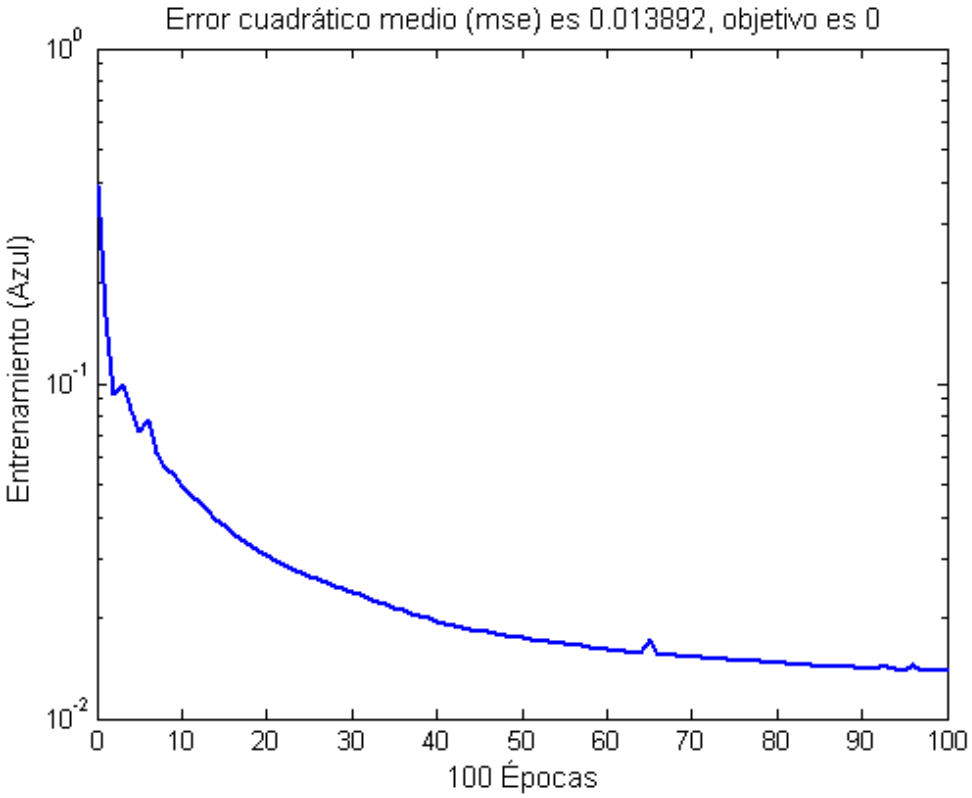
En la figura 12 podemos ver que la mayor eficiencia se obtiene con la red 196-50-10, con agrupación de sub-imágenes de 2x2; pero como mencionamos anteriormente, se tendría una mayor cantidad de datos y neuronas que considerar.

Por este motivo se escoge la siguiente configuración en eficiencia, siendo la red neuronal 49-25-10 la escogida. Esta red nos provee una buena eficiencia (92.11%), pesos sinápticos manejables y una adecuada cantidad de datos y neuronas.

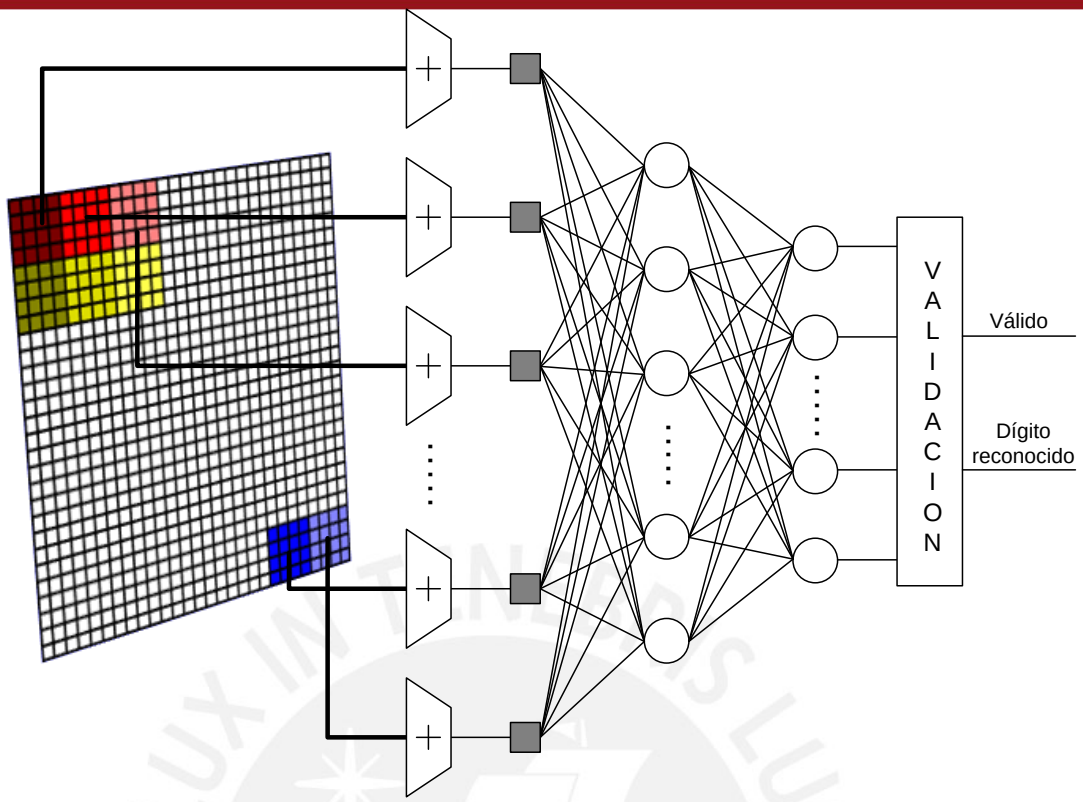
De esta forma, el sistema que permite el reconocimiento de dígitos se compone de una etapa de extracción de características, la red neuronal MLP y una etapa de validación, que se desarrolló en hardware y será explicado posteriormente, y permite verificar que los resultados de la red comparando las posibles neuronas ganadoras con un rango mínimo permisible que garantice que el patrón ha sido reconocido. La figura 13 muestra la red neuronal y el sistema completo, en el cual se distinguen claramente los componentes del mismo.



(a)



(b)



(c)

Figura 13. (a) Red neuronal escogida, diagrama por Matlab, (b) Variación de su Error cuadrático medio (mse) durante el proceso de entrenamiento, (c) Sistema completo: división de la imagen, red neuronal y etapa de validación.

4.1.3. Extracción de los pesos sinápticos y umbrales

Con la red neuronal diseñada mediante software, se cuenta con todos los parámetros necesarios para empezar el diseño de la red en hardware. Como será explicado a mayor detalle posteriormente, para el proceso de extracción de características se considera una precisión de centésimos; con lo cual, los valores hallados son redondeados a su valor más cercano.

Asimismo, estos pesos y umbrales son separados en sus respectivas partes enteras y decimales para disminuir el tamaño de los multiplicadores a utilizar, pues esta operación es una de las más críticas en el sistema.

De esta manera, se realizaron los scripts para generar los archivos necesarios en el formato requerido de constantes en el lenguaje de descripción de hardware VHDL. Con ello se automatiza el esta etapa del proceso, el cual empieza con la elección

del diseño de la red y culmina con los archivos generados para ser incluidos en la descripción del circuito.

Como ejemplo se muestra a continuación una porción de uno de los archivos generados que en este caso se refiere a la parte entera de los umbrales de la capa de salida de la red neuronal.

```
constant b2_ent : bias2_ent:=(

0 =>conv_std_logic_vector(0,b),

1 =>conv_std_logic_vector(-3,b),

2 =>conv_std_logic_vector(0,b),

3 =>conv_std_logic_vector(-2,b),

4 =>conv_std_logic_vector(-3,b),

5 =>conv_std_logic_vector(0,b),

6 =>conv_std_logic_vector(-1,b),

7 =>conv_std_logic_vector(4,b),

8 =>conv_std_logic_vector(-3,b),

9 =>conv_std_logic_vector(2,b));
```

Figura 14. Extracto de un archivo generado en el proceso de extracción de los pesos sinápticos y umbrales.

4.2. Diseño de la red neuronal sobre el FPGA CYLONE II de ALTERA

Con la red neuronal elegida, se procede con el diseño de la arquitectura en hardware de la misma. Esta se compone de estructuras básicas como son las neuronas, quienes componen elementos más complejos como son las capas neuronales. También, se presentan componentes adicionales para el flujo de datos y para el control de los bloques que permiten el correcto funcionamiento del sistema completo.

Sin embargo, el primer paso a considerar es la representación numérica a utilizar. Para ello, se realizó un histograma de los valores pesos sinápticos encontrados, el que se muestra en la figura 15, donde 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 buen funcionamiento de la red.

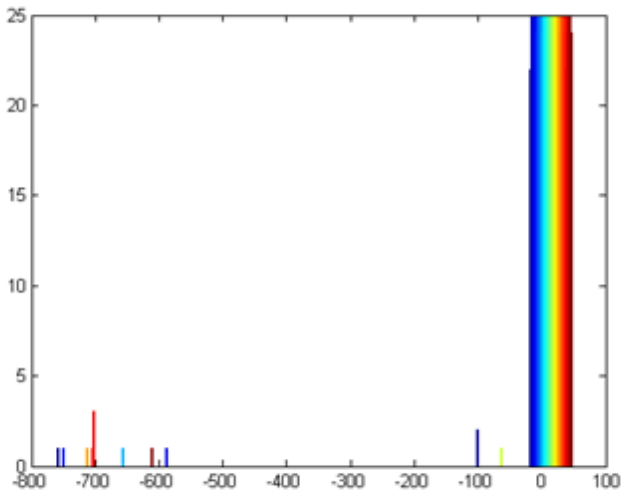


Figura 15. Histograma de los pesos sinápticos de la primera capa.

Observando el análisis 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 (debido al valor máximo del peso sináptico encontrado: 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 en parte decimal y trabajar ambas partes por separado y en paralelo; con lo cual se reduce el número de 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:

$$c = \sum_{i=1}^{n} p_i \cdot w_i = \sum_{i=1}^{n} p_i (w\_ent_i + w\_dec_i), \quad (3)$$

$$c = \sum_{i=1}^{n} p_i \cdot w\_ent_i + \sum_{i=1}^{n} p_i \cdot w\_dec_i, \quad (4)$$

La siguiente tabla muestra los parámetros de los pesos obtenidos tras el proceso de entrenamiento, donde w representa a los pesos sinápticos y b a los umbrales.

Tabla 5: Bits para la representación en complemento a 2 de los pesos sinápticos y umbrales de la red [Elaboración propia].

Parte	Capa Oculta		Capa Salida	
	w	b	w	b
Entera	11	4	9	4
Decimal	8	8	8	8

Con estas consideraciones, y teniendo en cuenta los valores ya calculados, se desarrolla la arquitectura de la red neuronal combinando el planteamiento expuesto por Volnei A. Pedroni en [17] con la idea propia mencionada previamente de separar las partes enteras y decimales de los pesos sinápticos y umbrales. De esta forma, se puede tener una mayor exactitud en los cálculos con los decimales y mayor facilidad para realizar escalamientos si se tuvieran magnitudes muy elevadas en la parte entera.

4.2.1. Descripción y diseño de la arquitectura de una Neurona Artificial

La neurona artificial es la unidad básica fundamental de la red neuronal; sin la cual, no es posible su desarrollo. Es el bloque principal y por lo cual, su descripción debe ser lo más parametrizable posible para que pueda adaptarse a cada una de las capas a la que puede pertenecer.

De esta manera, la arquitectura propuesta es mostrada en la figura 16. Consta de un bloque funcional, figura 17, constituido por dos MAC's (Multiplicador-Acumulador), 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 operación que realiza es la mostrada en la ecuación (5), donde φ representa la función sigmoide, a_j la salida de la neurona j y j la cantidad de neuronas por capa.

$$a_j = \varphi\left(\sum_{i=1}^{n} p_i \cdot w_{ij} + b_j\right) \tag{5}$$

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, esta característica es aprovechada al utilizar conexiones a '1' o '0' lógico envés de recursos lógicos propios del dispositivo como son los bloques de memorias.

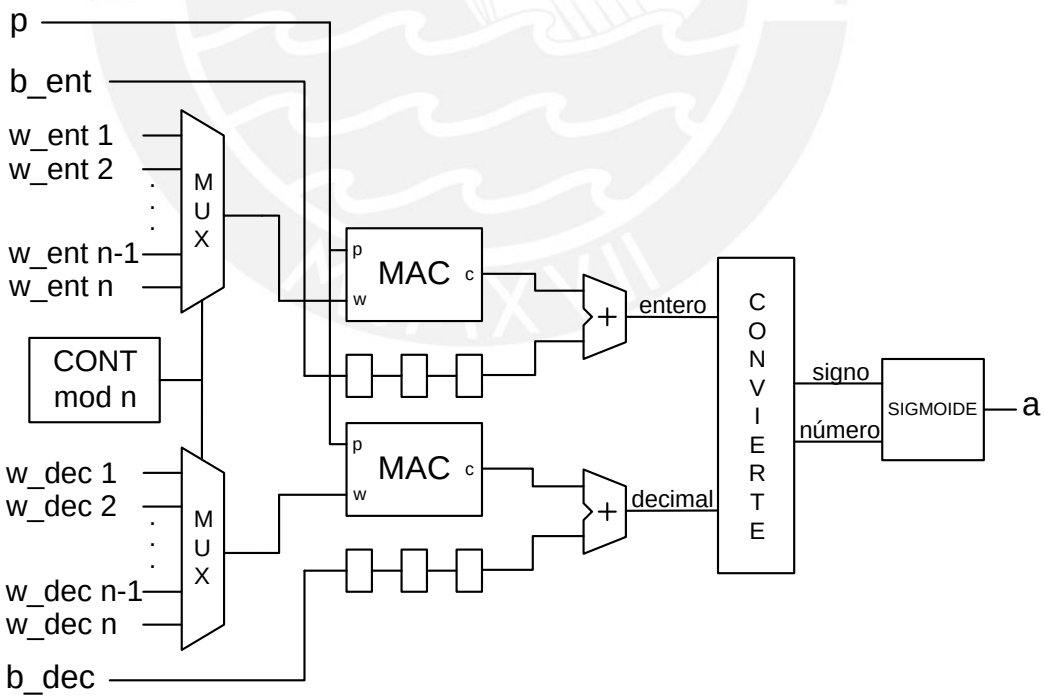


Figura 16. Neurona Artificial.

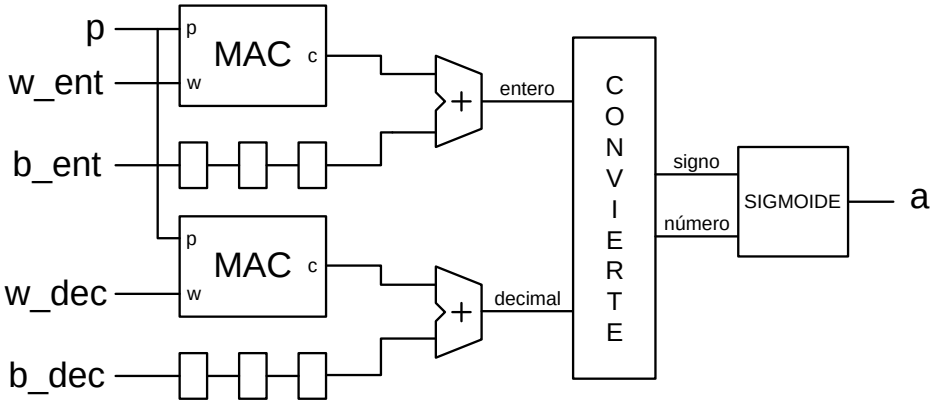


Figura 17. Bloque Funcional de la neurona artificial.

4.2.1.1. Multiplicador-Acumulador

Consiste de un arreglo básico conocido de un multiplicador y un sumador que permite realizar una suma de productos, ya sea de números signados, como en nuestro caso, o no. Su arquitectura se muestra en la figura 18.

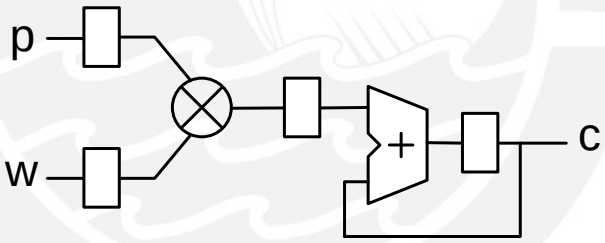


Figura 18. Multiplicador-Acumulador (MAC).

El MAC utilizado contiene las entradas y la salida registradas, pues forma parte de un circuito síncrono y permite la unión de bloque con mayor facilidad. Además, reduce la cantidad de lógica entre 2 registros incrementando la frecuencia máxima de reloj que soportaría el sistema.

Adicionalmente, para el diseño del MAC se consideró el posible desborde que se pudiera generar en el proceso de acumulación, colocando al valor máximo respectivo según el desborde haya sido positivo o negativo. Esto es de importancia en el diseño porque, de no ser tomado en cuenta, se perdería la acumulación parcial y se tendría un resultado erróneo al finalizar la suma de productos.

Para el diseño del multiplicador, se presentan dos casos según la tabla 5. Para los multiplicadores en los cuales se tiene como un multiplicando a la parte entera de un peso sináptico de la capa oculta se realizó el bloque de multiplicación mediante la Megafunción LPM_MULT versión WM1.0 de Altera; mientras que para los demás casos se utilizaron los multiplicadores embebidos presentes en el FPGA Cyclone II. Esto porque dichos multiplicadores embebidos pueden ser configurados como 70 multiplicadores de 9x9 bits o 35 de 18x18 bits. Así, para 11 bits, se utilizarían 2 multiplicadores embebidos de 9x9, desperdiciando la capacidad de estos bloques.

4.2.1.2. Bloque Convierte

Tras obtener por separado los resultados de la parte entera y decimal de la suma de productos, es necesario formar el número completo para evaluar el resultado en la función de activación, todo esto según (5). El resultado que debe generar debe estar escalado por un factor de 100 para que cumpla con los requerimientos del siguiente bloque que evalúa la función sigmoide, el cual será explicado posteriormente. Como los valores de la parte decimal ya están escalados solamente se requiere de una multiplicación por 100 en la rama de la parte entera. Por último, la salida el número formado se compara con un umbral para acoplar la salida a la entrada de la función sigmoide.

Así, la salida de este bloque es el número completo menor o igual que el umbral, descompuesto en su signo y magnitud, permitiendo la adaptación de ambos bloques.

La arquitectura de este bloque se muestra a continuación en la figura 19.

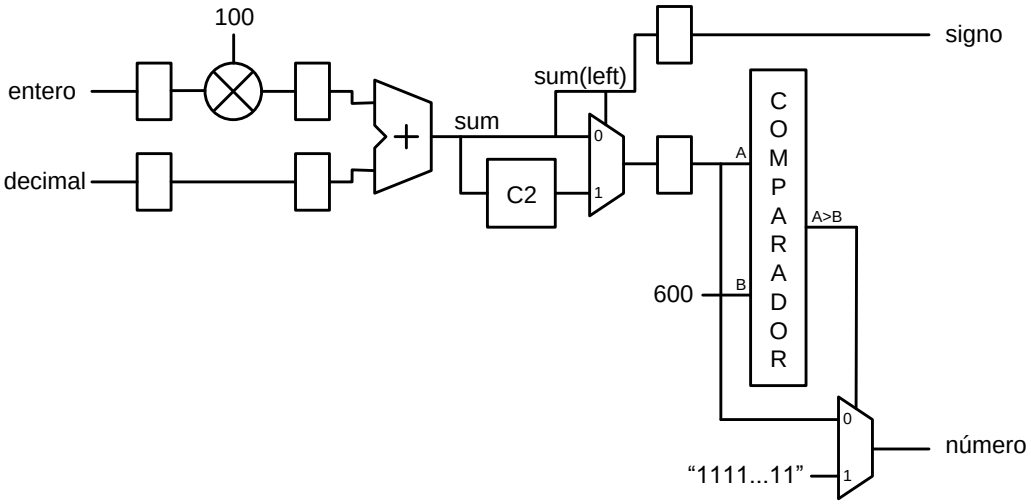


Figura 19. Bloque *Convierte.*

4.2.1.3. Función Sigmoide

La función sigmoide (ecuación 6) fue diseñada según lo desarrollado en [21]. El algoritmo utilizado según el diagrama de flujo de la figura 20, conduce a una implementación mediante tablas de verdad (Look-Up Tables) y una memoria ROM. Este algoritmo aprovecha que esta función esta acotada entre 0 y 1, y también la aproximación de la misma mostrada en la ecuación 7.

$$f(x) = \frac{1}{1+e^{-x}}, \tag{6}$$

$$f(-x) = 1 - f(x), \tag{7}$$

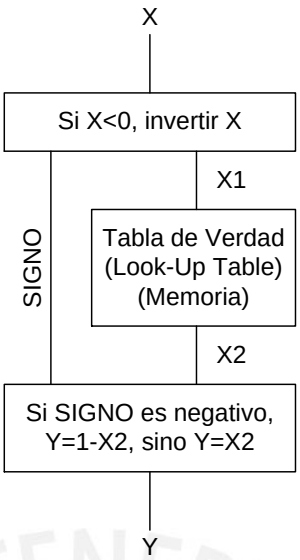


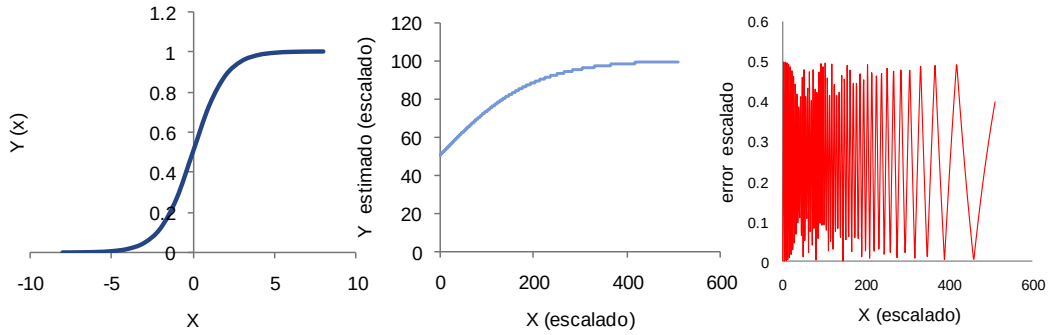
Figura 20. Diagrama de flujo del algoritmo para el cálculo de la función Sigmoide [22].

Para ello, los valores reales de la función fueron escalados por un factor de 100 y redondeados a enteros. Según se muestra en la figura 22, la función empieza a saturarse a partir del valor de 4; con lo cual debemos utilizar una memoria de capacidad superior a 400 posiciones. De esta forma se utiliza una memoria de 512 posiciones y se escogió como valor de saturación al número 5.11, con lo cual utilizamos toda la capacidad de esta memoria. Los respectivos valores son almacenados en orden creciente correspondiendo la última posición (511) a la función evaluada en el punto de saturación. Este proceso de escalamiento se muestra a continuación.

$$\begin{aligned} f(0) &= 0.5 \\ f(0.81) &= 0.6921095 \\ f(5.11) &= 0.99400013 \end{aligned} \quad \Rightarrow \quad \begin{aligned} f(0) &= 50 \\ f(81) &= 69.21095 \\ f(511) &= 99.400013 \end{aligned} \quad \Rightarrow \quad \begin{aligned} f(0) &= 50 \\ f(81) &= 69 \\ f(511) &= 99 \end{aligned}$$

Figura 21. Proceso de escalamiento y redondeo.

Las gráficas obtenidas con este proceso, así como el error en la aproximación se muestran en la figura 22.



(a) **(b)** **(c)**

Figura 22: (a) función sigmoide, (b) aproximación realizada con los valores escalados, (c) error obtenido en la aproximación.

Como podemos apreciar en la figura 20, el primer paso en el algoritmo es determinar el signo de la abscisa y su valor absoluto. Esta parte del algoritmo ya se encuentra implementada al final del bloque *Convierte* (ver figura 19), por lo que en esta etapa se tienen como entrada a X1 y el SIGNO, todo esto según la figura 20. Así, el diseño se muestra a continuación en la figura 23.

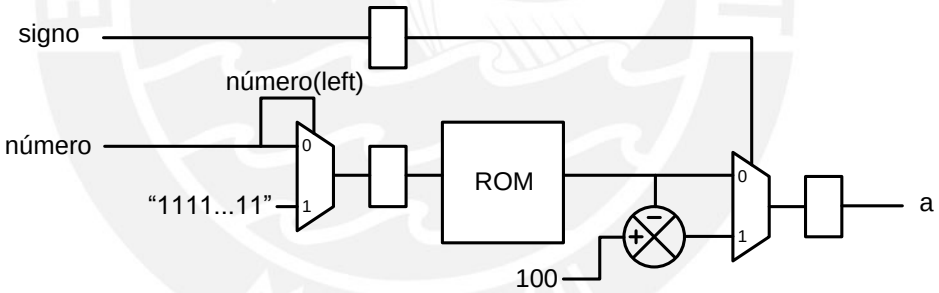


Figura 23. Función Sigmoide.

4.2.2. Descripción de la arquitectura de la red neuronal

Teniendo el bloque funcional básico completo (neurona artificial), el diseño de la arquitectura de la red neuronal consiste básicamente en la unión de neuronas según las dimensiones encontradas. Así, la red está conformada en base a capas neuronales; en este caso, de una capa oculta de 25 neuronas y una capa de salida de 10 neuronas.

La capa oculta presenta un contacto directo con las entradas de la red neuronal, por lo que el bloque funcional descrito anteriormente se utiliza tal cual en esta etapa. Sin embargo, hay que considerar que también debe sincronizar la entrada de los datos con la memoria que contiene las características extraídas de la imagen umbralizada. Este proceso se realiza mediante la correcta interpretación de las banderas de señalización, tarea realizada por el controlador (máquina de estados) de la capa oculta.

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, debido a que son salida de la función sigmode, 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. Este bloque es denominado *Convierte2*. Al igual que la capa anterior, la sincronización de ambas etapas se realiza a través de la interacción de los controladores (máquinas de estados) de ambas capas neuronales. Esto es de importancia pues permite que los cálculos se realicen adecuadamente.

4.2.2.1. Capa Neuronal

Siguiendo la jerarquía en el diseño, la capa neuronal es el siguiente bloque funcional que compone la red neuronal. Su arquitectura se muestra en la figura 24 y 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. Estas banderas son interpretadas por el controlador correspondiente a cada capa neuronal y permiten su interacción.

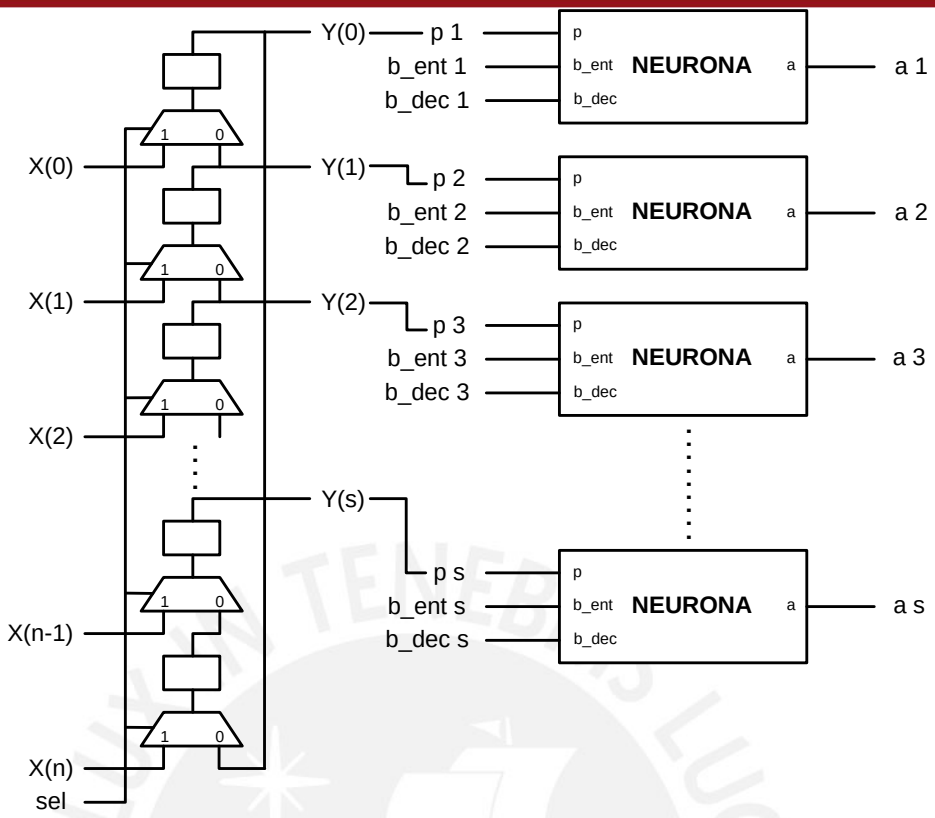


Figura 24. Capa Neuronal: Registro de desplazamiento circular unido con las neuronas que forman la capa.

De esta forma, los datos circulan dentro del registro e interactúan con cada una de las neuronas una sola vez, y con el peso sináptico indicado. Estos datos ingresan a cada neurona y obtienen la suma de productos, su suma con el valor del umbral, y su posterior evaluación en la función de activación, la función sigmoide.

Este bloque funcional espera la activación de una bandera para permitir el ingreso de datos al registro de desplazamiento circular y mantiene la salida de cada neurona hasta que los mismos hayan sido leídos por la etapa posterior. De esta manera, se toman los valores en el momento preciso y no se pierden los resultados obtenidos tras el proceso de la información.

4.2.2.2. Registro de Desplazamiento Circular

Este bloque forma parte de la arquitectura de la capa neuronal y está constituido de un multiplexor y un registro en un arreglo circular que permite que los datos no se pierdan y circulen al interior del registro. Su arquitectura se muestra en la figura 25.

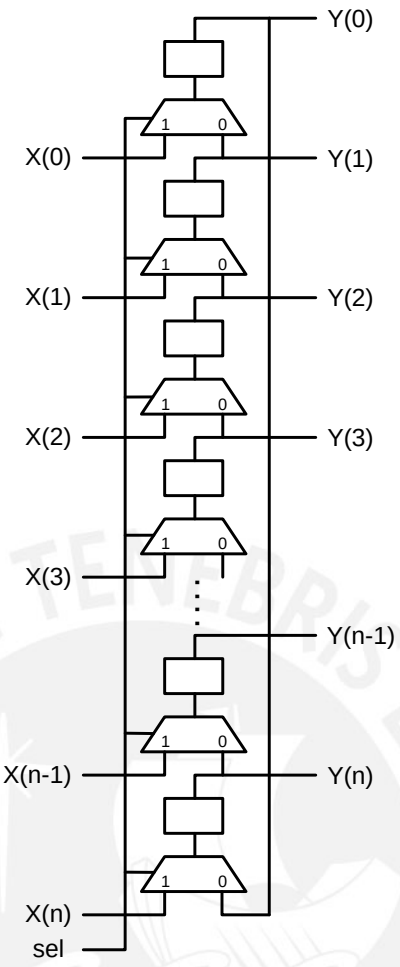


Figura 25. Registro de Desplazamiento Circular.

El multiplexor permite, a través de un selector, que los datos sean ingresados al registro de manera paralela y que se mantengan a la salida de los registros hasta que cambie de valor el selector. Tras esto último, el multiplexor permite el paso de las salidas de los registros y los datos empiezan a circular, siendo en este caso, de abajo hacia arriba.

Una característica de la arquitectura propuesta para este registro es que presenta un número diferente de entradas respecto al número de salidas, siendo esta relación de mayor cantidad de entradas que de salidas. Esto es porque se pasa de una capa de mayor cantidad de neuronas a una de menor cantidad. Sin embargo, el diseño es fácilmente adaptable para el caso contrario, puesto que el diseño principal es el mismo en ambos casos.

4.2.2.3. Bloque Convierte2

Al igual que el bloque *Convierte* anterior, realiza la misma labor; solo que presenta una diferencia respecto al escalamiento que debe realizar. Como se explicó previamente, en este caso las salidas ya se encuentran escaladas por el factor de 100, por lo que se debe quitar la multiplicación de la rama de la parte entera. Respecto a la rama de la parte decimal, se debe añadir un divisor entre 100 puesto que la parte decimal incorpora inherentemente un escalamiento de 100, y al tener uno adicional, debe ser eliminado.

La arquitectura se muestra en la figura 26 y presenta gran similitud con el bloque Convierte. Este es el único cambio que se efectúa en las neuronas de la capa de salida y es este tipo de neurona la que debe ser usada a partir de la segunda capa oculta si la red neuronal presentase más de una de ellas.

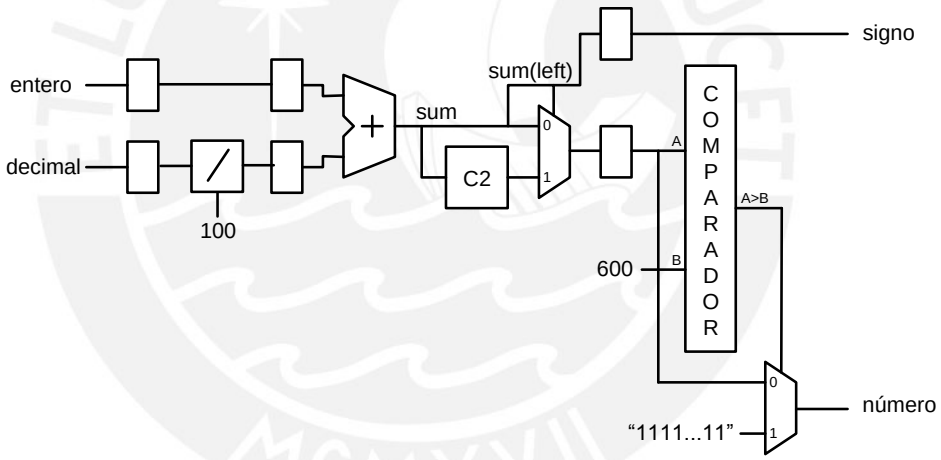


Figura 26. Bloque *Convierte2.*

4.2.2.4. Controlador

Circuito secuencial encargado de realizar la secuencia de estados necesarios para llevar a cabo con éxito los cálculos en la capa neuronal y así otorgar una respuesta confiable a la etapa siguiente. Se compone de una máquina de estados y las banderas necesarias de los bloques adyacentes. La tabla 6 muestra la tabla de estados del controlador.

Tabla 6: Tabla de Estados del Controlador [Elaboración propia].

Estado Presente	Estado Siguiente	Condición
Inicializa	Espera	Directamente
Espera	Espera	Opera = 0
Espera	Carga_datos	Opera = 1
Carga_datos	Nop_carga	Directamente
Nop_carga	Circula_datos	Directamente
Circula_datos	Circula_datos	q_done = 0
Circula_datos	Delay	q_done = 1
Delay	Delay	delay_done = 0
Delay	Nop1	delay_done = 1
Nop1	Nop2	Directamente
Nop2	Inicializa	resultado_leido = 1
Nop2	Nop2	resultado_leido = 0

Las banderas de señalización permiten la interacción del controlador con la capa neuronal y los bloques adyacentes. Así, la señal *Opera* indica que la etapa previa ha concluido y que la presente puede empezar. Este proceso empieza con el ingreso de los datos al registro circular. La señal *q_done* indica que la cuenta del contador ha finalizado, lo que significa que se ha recorrido todo el registro circular; con lo cual empieza un periodo de delay en el cual se termina de procesar el último dato de entrada. Posteriormente, la señal *delay_done* indica que este periodo ha concluido, con lo cual esperamos un par de ciclos para indicar que se tiene la respuesta a la salida de la capa neuronal. Finalmente, el resultado se mantiene a la salida y el circuito permanece en este estado hasta que las salidas hayan sido leídas. Esto se indica con la activación de la señal *resultado_leido*, con lo cual el sistema regresa a su estado de inicialización y a esperar los siguientes datos a procesar.

Debido a que las señales de control tienen acción directamente con el circuito y a que la mayoría de estas son entradas registradas en sus respectivos bloques, las salidas del controlador son no registradas.

4.2.3. Diseño en VHDL de la arquitectura de la red neuronal

Para el diseño de la arquitectura desarrollada mediante el lenguaje de descripción de hardware VHDL se siguió la siguiente metodología sugerida en [17]. El flujo de diseño se muestra en la figura 27 y en ella se observa los pasos a seguir para alcanzar un circuito exitoso.

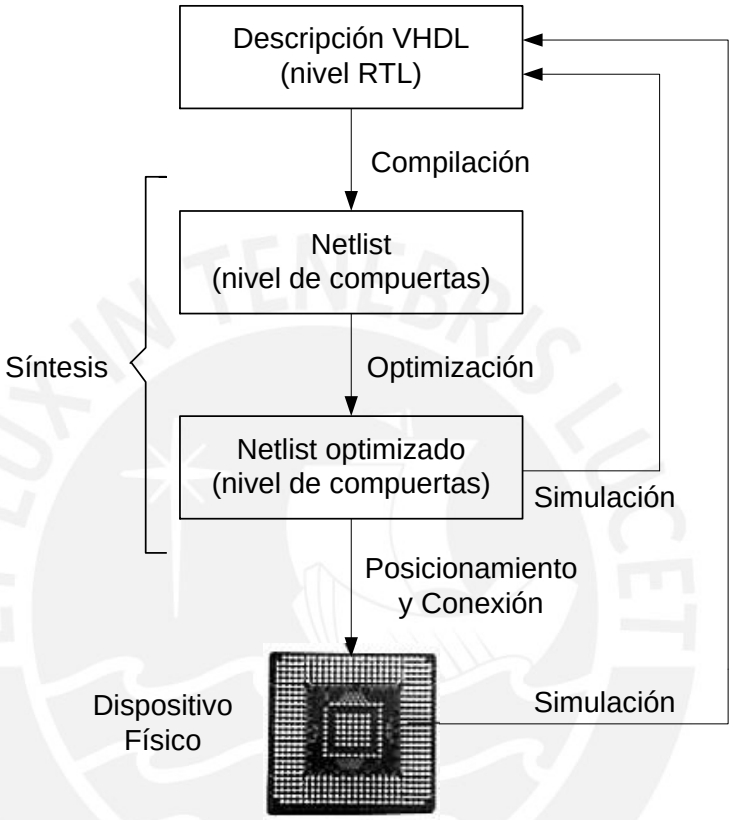


Figura 27. Resumen del flujo de diseño con VHDL [17].

El diseño se realizó en base a una descripción estructural de los componentes de cada una de las partes del circuito, y utilizándolos según donde sean requeridos mediante un *mapeo*, que consiste en instanciar el dispositivo descrito previamente en otro circuito.

También, el diseño de los circuitos fue completamente genérico y modular. Genérico porque todos los parámetros son referidos ya bien desde la sentencia GENERIC dentro de la entidad del código o desde un archivo aparte que sirve de archivo de configuración del circuito descrito o mediante atributos de las señales. Modular porque el diseño del sistema se compone de módulos o bloques funcionales más pequeños.

Se utilizo un único paquete de componentes para evitar cualquier confusión en el momento de instanciarlos. Asimismo, un único paquete para definir los parámetros de la red en forma de constantes y un único paquete para los tipos de datos utilizados. El fin es el mismo, tener en el mismo archivo todos los parámetros comunes para utilizarlos según sean requeridos.

Todos los componentes del sistema fueron descritos a excepción de los multiplicadores, divisores y memorias. En estos casos, se utilizaron Megafunciones de Altera para utilizar los recursos embebidos o bloques optimizados por el fabricante, o símbolos como "*" que instancia un multiplicador embebido en el FPGA. Para el caso de los sumadores, el operador "+" es sintetizado por el Quartus II y utiliza los circuitos para suma y acarreo que se encuentran en cada elemento lógico.

Según esto, el circuito descrito pierde un poco su portabilidad frente al cambio de fabricante o de familia; sin embargo, son bloques que se encuentran, en la actualidad, en la mayoría de los FPGA's; por lo que se puede decir que es fácilmente adaptable a un nuevo dispositivo.

Se ha tenido cuidado durante la descripción para evitar que exista mucha circuitería concurrente entre dos registros pues limita la frecuencia máxima de operación del sistema. Así, entre componentes u operaciones que requieran de grandes recursos son instanciados o colocados registros que cumplan con lo estipulado líneas atrás.

El uso adecuado de las herramientas que nos proporciona el VHDL nos permitió una fácil descripción de porciones de circuito que se repiten a lo largo del circuito. Esto se llevo a cabo utilizando la sentencia FOR GENERATE. Del mismo modo, la sentencia FOR LOOP permite realizar síntesis similares dependiendo la descripción del circuito dentro del PROCESS.

La mayoría de los circuitos presentan registros con entradas de habilitación y reset. Esto permite que sean fácilmente inicializados durante un estado de inicialización en el cual se activen todas las entradas de reset de los registros del circuito. En cuanto a los habilitadores, estos permiten un mejor control en el comportamiento y funcionamiento de los bloques constituyentes del sistema y también del sistema global.

Una buena práctica en la descripción de los diversos componentes fue la de evitar que el software instancie latches indeseados a la salida de los bloques. Esto se

logró realizando una buena descripción al momento de utilizar procesos para describir circuitos combinacionales.

Por último, para no utilizar recursos lógicos con los pesos sinápticos y umbrales hallados, se han utilizado paquetes para declararlos como constantes. De esta forma se utilizan las interconexiones del FPGA hacia la alimentación mas no recursos lógicos o memorias.

CAPITULO 5

SIMULACIONES Y RESULTADOS

En el presente capítulo se presentan los procesos de validación del diseño realizado y los resultados obtenidos. En un primer momento, los distintos bloques conformantes del sistema son simulados para verificar su comportamiento frente a diversos estímulos, procurando cubrir la mayor cantidad de casos que se pudiesen presentar. Posteriormente, los bloques se van uniendo en entidades mayores y son simulados a su vez para verificar el funcionamiento en conjunto. Este procedimiento continúa hasta formar el sistema completo. Finalmente, se procede a la implementación del sistema completo en el FPGA y a la verificación de su comportamiento en la tarjeta de desarrollo y mediante la visualización real del comportamiento dentro del dispositivo a través de la herramienta Signal Tap II Logic Analyzer del Quartus II v.7.1 de Altera.

Para esta parte, se desarrollaron componentes y herramientas adicionales a la red neuronal para permitir su interacción con los componentes de la tarjeta de desarrollo y con la computadora personal (Personal Computer, PC). El diseño de estos componentes se muestra en la primera sección del capítulo.

Por último, los resultados obtenidos a través de las pruebas son mostrados y analizados en la última sección del capítulo.

5.1. Diseño y desarrollo de componentes y herramientas adicionales

Para poder realizar las simulaciones del sistema y verificar su funcionamiento, se necesitaba acceder a la red neuronal desde el exterior, es decir, fuera del FPGA; ya sea mediante la tarjeta de desarrollo o mediante la PC.

Así, las etapas de extracción de características (división de la imagen) y validación, mostradas en la figura 13, fueron desarrolladas. Del mismo modo, una interfaz vía serial con la PC para realizar pruebas con la base de datos de test del MNIST y corroborar lo obtenido en software.

5.1.1. Etapa de pre-procesamiento

Realiza las etapas de umbralización y extracción de características de la imagen digital del número. La arquitectura desarrollada es presentada en la figura 28.

La imagen, almacenada en una memoria RAM tras ser recibida desde la PC vía puerto serial, es umbralizada byte por byte mediante un comparador. El bit obtenido es acumulado según su pertenencia a la sub-imagen correspondiente para que, cuando se culmine con los 16 bits que conforman esta sub-imagen de 4x4, sean almacenados en su respectivo registro.

De esta manera, los primeros 7 registros corresponden a las sub-imágenes pertenecientes a la porción superior de la imagen, empezando por la sub-imagen de la izquierda (registro 0) y culminando con la de la derecha (registro 6). Este proceso prosigue hasta culminar con las 49 sub-imágenes, donde la última de estas corresponde a la del extremo inferior derecho (registro 48).

Este bloque se conecta directamente con las entradas de la red neuronal y permite que las características de la imagen lleguen tal cual se esperaba a la red para su procesamiento.

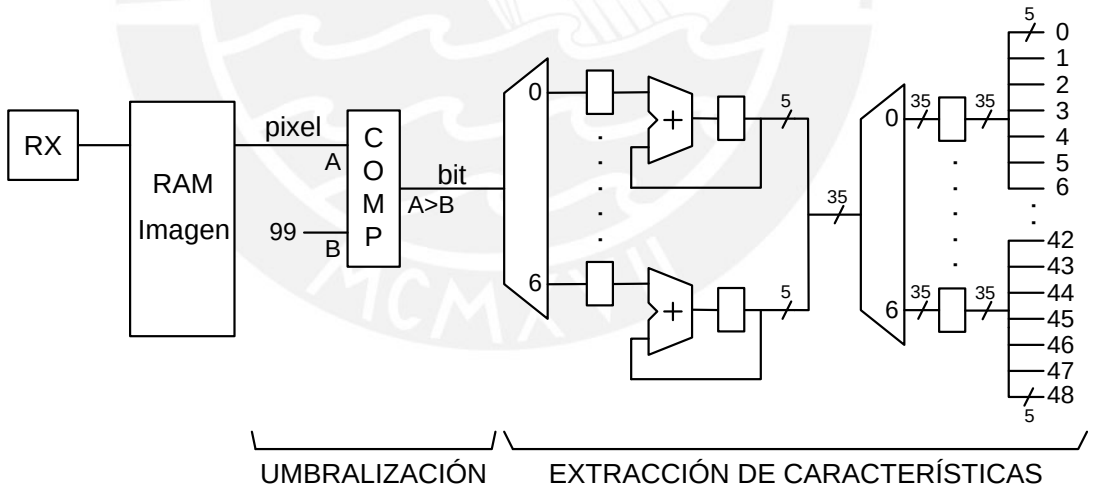


Figura 28. Etapa de pre-procesamiento.

5.1.2. Etapa de post-procesamiento

El resultado de la red neuronal es un vector de 10 elementos, cada uno entre 0 y 100 (salidas de la función de sigmoide escaladas); donde el elemento con la magnitud mayor (más cercana o igual a 100) corresponde a la neurona ganadora. La característica presente en este vector es que solamente uno de sus elementos es cercano a 100, mientras que los otros son valores cercanos a 0 o menores al mayor en una relación aceptable. Esto se debe a que la red neuronal busca reconocer un determinado patrón en la entrada y según sus parámetros de entrenamiento genera una salida en la solamente una neurona se sienta estimulada mientras que las otras inhibidas.

Sin embargo, existen casos en los cuales los patrones (dígitos) presentados en la entrada no son muy distinguibles y presentan ambigüedad para su reconocimiento. Para citar un ejemplo podemos mencionar la escritura del número 1, el cual puede parecerse al 7; o el 5 con el 6; entre otros. Estos patrones generan una salida en la cual existen más de un elemento con valor cercano a 100 e indican que el patrón a la entrada puede pertenecer a más de una clase.

Por estos motivos, para tener un sistema más robusto, se implementó una etapa de validación en la cual se verifica que solamente exista una neurona en cuya salida se tenga un valor cercano a 100, y que este valor este dentro de un rango aceptable para poder asegurar con mayor certeza que el resultado es correcto. Esto se consigue mediante la activación o no de un bit que indica si el resultado es válido o no según lo expuesto anteriormente.

Para resolver esto, cada salida es comparada y posteriormente es codificada mediante un codificador para obtener el número binario reconocido. Para la validación del resultado, las salidas de la comparación previa son sumadas y comparadas con el valor de 1. Así, solamente cuando la suma sea igual a 1 se activará el bit de validación debido a la igualdad en la entrada del comparador. La arquitectura de esta etapa de post-procesamiento se muestra en la figura 29.

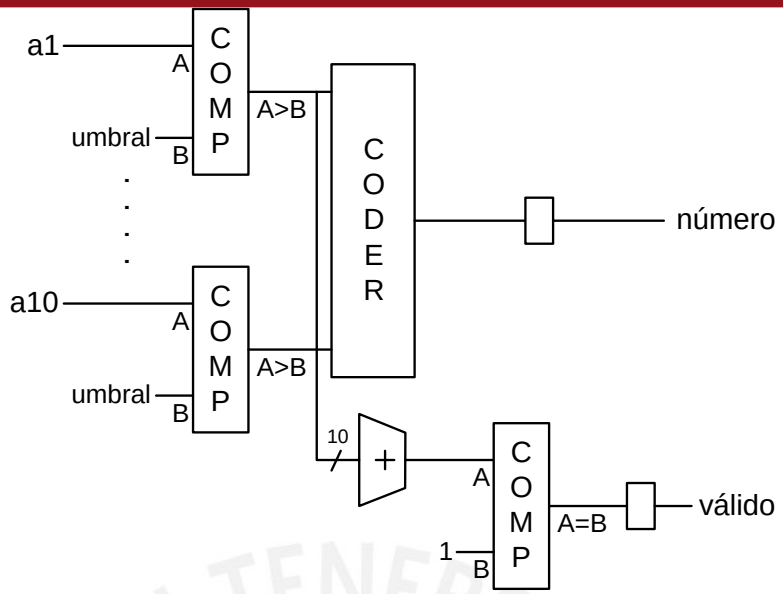


Figura 29. Etapa de post-procesamiento.

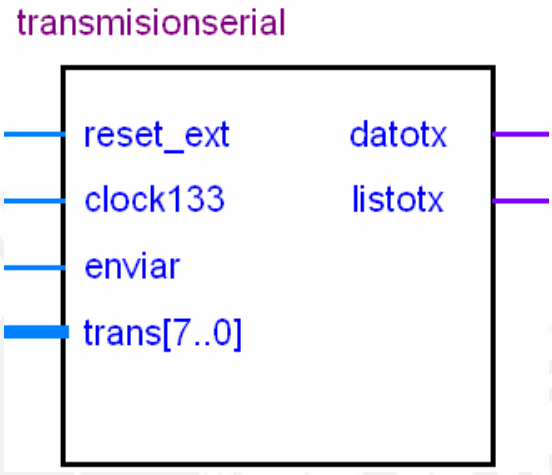
5.1.3. Interfaz serial FPGA-PC

Para realizar las pruebas desde una PC, se tiene que comunicar el diseño en el FPGA con la PC vía algún protocolo de comunicación. Para el presente trabajo se decidió realizar la comunicación mediante el protocolo RS-232 (Puerto serial).

Se utilizó la arquitectura desarrollada en el Grupo de Microelectrónica de la Pontificia Universidad Católica del Perú por el Ing. Mario Andrés Raffo Jara, quién nos permitió su uso y modificación según el criterio de la aplicación a la que iba a ser destinada. Así, los dos bloques, transmisión y recepción fueron adaptados para nuestra tarea a 50 MHz y a 57600 baudios, 8 bits de datos, sin paridad y 1 bit de parada como trama serial [23].

La figura 30 muestra los bloques de transmisión serial y recepción serial originales, donde se muestran las señales de interacción de los bloques; del mismo modo, los cuadros 1.a y 1.b explican las funciones de estas señales y de los parámetros de configuración.

(a)



(b)

Figura 30. Bloque Serial: (a) Recepción Serial y (b) Transmisión Serial.

Cuadro 1.a. Parámetros y señales del bloque de recepción serial [Elaboración propia].

Señal	Tipo	Función
divisor	Parámetro	Regula la velocidad de transmisión de los datos.
reset_ext	Entrada	Señal de reseteo asíncrono del bloque.
clock 133	Entrada	Señal de reloj.
rx	Entrada	Señal de dato de entrada serial.
datorx	Salida	Bus de datos recibidos.
listorx	Salida	Indicador de validez del dato recibido.

Cuadro 1.b. Parámetros y señales del bloque de transmisión serial [Elaboración propia].

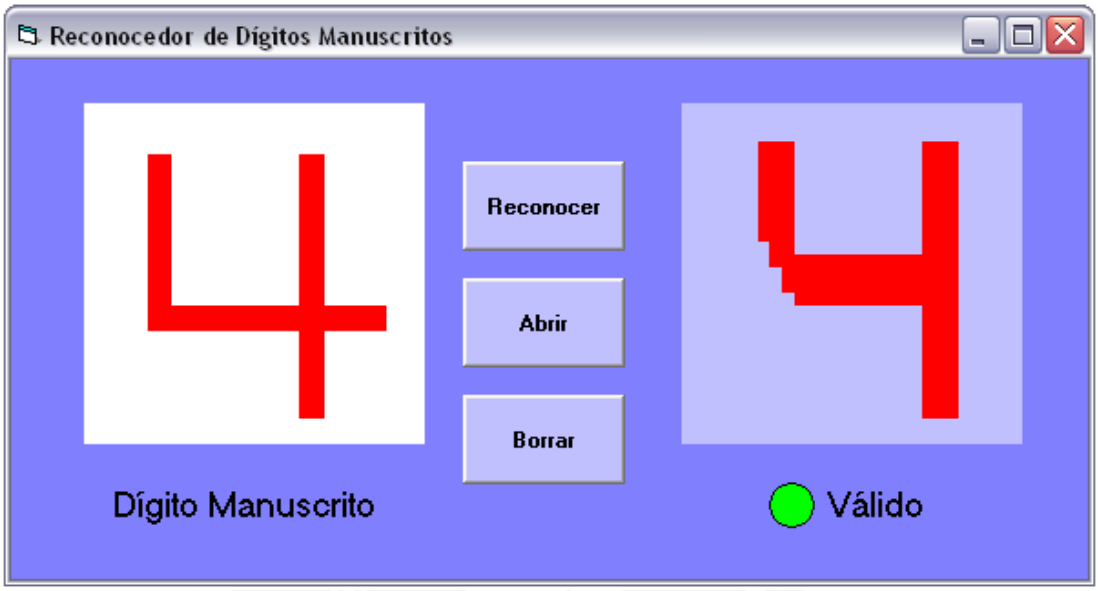
Señal	Tipo	Función
divisor	Parámetro	Regula la velocidad de transmisión de los datos.
reset_ext	Entrada	Señal de reseteo asíncrono del bloque.
clock 133	Entrada	Señal de reloj.
enviar	Entrada	Señal indicadora de inicio de envío del dato.
trans	Entrada	Señal de dato de entrada de serial.
datotx	Salida	Dato enviado.
listotx	Salida	Indicador de término del envío del dato.

Adicionalmente, una máquina de estados permite la interacción de estos bloques con el sistema completo de la red neuronal. Así, la imagen es transmitida desde la PC hacia el FPGA y es almacenada en una memoria; posteriormente, los resultados generados son transmitidos desde el FPGA hacia la PC para ser mostrados en el software desarrollado para esta interfaz.

5.1.4. Software para la interacción con la red neuronal

El presente software, desarrollado en Visual Basic, permite la interacción del usuario desde la PC hacia el FPGA y viceversa a través del puerto serial y protocolo RS-232. El formulario principal, mostrado en la figura 31, consta de 2 recuadros y 2 botones. El primero, de color blanco, se dedica a albergar al dígito manuscrito a ser reconocido mientras que el segundo, de color celeste, muestra la imagen del dígito recibido como respuesta del FPGA. Asimismo, se cuenta con un indicador de validez que indica si la respuesta recibida desde el FPGA es correcta o incorrecta; en el primer caso se encenderá de color verde mientras que en el segundo de color rojo y cambiará su etiqueta a *Incorrecto*. Respecto a los botones, el botón *Reconocer* envía la imagen hacia el FPGA y luego se espera la respuesta de este para mostrarlo en pantalla, mientras que el botón *Abrir* permite abrir una

imagen binaria almacenada en texto para que sea mostrada en pantalla y reconocida posteriormente tras presionar el botón *Reconocer*.



(a)

(b)

(c)

Figura 31. Software para la interfaz serial PC-FPGA. (a) Formulario principal, (b) señalización con reconocimiento válido, y (c) señalización con reconocimiento incorrecto.

Presenta 2 modos de uso al momento de cargar el dígito manuscrito; el primero permite escribir o dibujar el dígito manualmente en el recuadro blanco mientras que el segundo, como se explicó líneas arriba, permite abrir una imagen binaria almacenada en texto y la visualiza en este mismo recuadro.

Al igual que el bloque serial dentro del FPGA, se encuentra configurado a 57600 baudios, 8 bits de datos, sin paridad y 1 bit de parada. El control de puerto serial se realiza por sondeo puesto que, a comparación de la transmisión en la que se envían 784 datos, solamente se reciben dos; por lo que una interrupción no es requerida.

5.2. Simulaciones

Los diversos bloques constitutivos del sistema fueron simulados para corroborar su funcionamiento. En cada simulación se consideró a las entradas del bloque como señales de excitación y a sus salidas como las respuestas producidas. Estas fueron realizadas mediante el simulador del Quartus II v.7.1 en el modo *Timing*, el cual considera los retardos de las señales dentro del dispositivo.

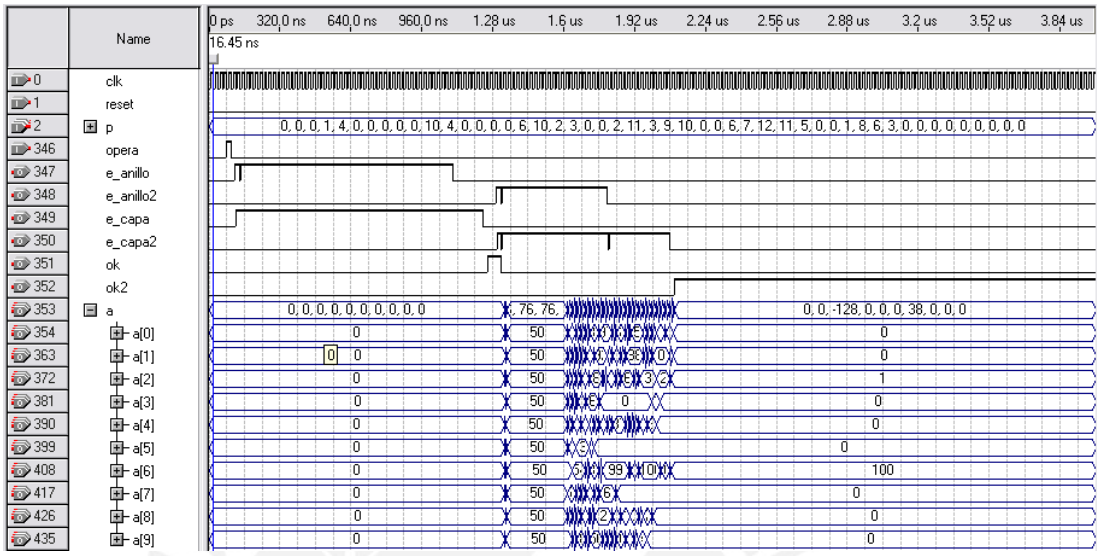
Las simulaciones principales realizadas son las referidas al MAC, al bloque Convierte, Convierte 2, a la Neurona Artificial, al registro de desplazamiento circular, a la capa neuronal, a la red neuronal completa y a las etapas de pre y post procesamiento. De todas estas, las tres últimas muestran mayor relevancia en cuanto a la información que otorgan sobre el comportamiento del sistema; así, cada una de ellas es explicada y mostrada a continuación.

La figura 32 muestra la simulación de la red neuronal completa; donde la señal *p* son las características extraídas de la imagen de un dígito; *opera* es la señal que indica el inicio del procesamiento; las señales *e_anillo*, *e_anillo2*, *e_capa* y *e_capa2* son los habilitadores de los registros de desplazamiento circular y de las capas neuronales de la capa oculta y de la capa de salida; las señales *ok* y *ok2* indican que el procesamiento en una capa ha culminado; y la señal *a* contiene el resultado de todo el procesamiento, en el cual cada elemento representa a un dígito.

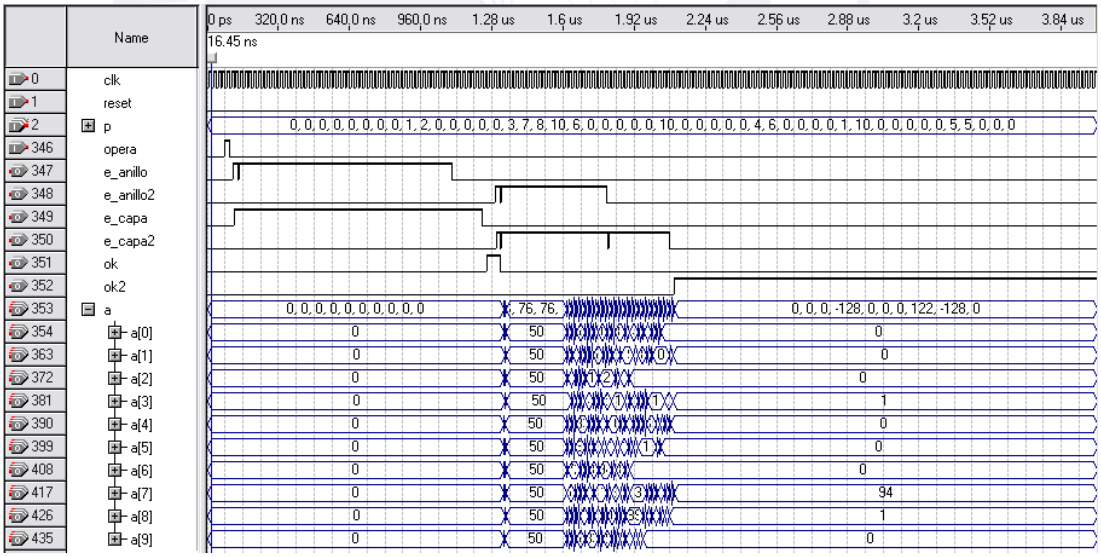
Como se puede apreciar, en la figura 32 (a) la señal *a[6]* es la mayor y presenta un valor de 100, mientras que las demás salidas son cero o muy cercanas a este valor. Esto nos indica que el dígito reconocido solo pertenece a una clase y es el número 6. Para el caso de la figura 32 (b), la señal *a[7]* es la mayor con un valor de 94, mientras que las demás son 0 o cercanos a este valor. Igual que el caso anterior, el dígito reconocido solo pertenece a una clase y es en esta ocasión el número 7.

La figura 33 muestra una simulación realizada a la etapa de pre-procesamiento, para la cual se almacenó en memoria los pixeles correspondientes a la imagen del dígito 7, el mismo utilizado en la simulación mostrada previamente. Para la simulación, la señal *recepcion_completa* indica el inicio del procesamiento; la señal *enable* es el habilitador del bloque; la señal *rst_acum* indica el momento en el cual el acumulador debe ser reseteado para que la acumulación empiece desde cero en la siguiente sub-imagen; la señal *indice* indica el bloque de registros en el cual se va a almacenar los resultados de los acumuladores; la seña *ok_total* indica el

término del proceso y la señal *p* contiene la salida de esta etapa, la cual es las características de la imagen almacena en memoria.



(a)



(b)

Figura 32. Simulación de la red neuronal teniendo como entrada a las características de la imagen del dígito: (a) 6 y (b) 7.

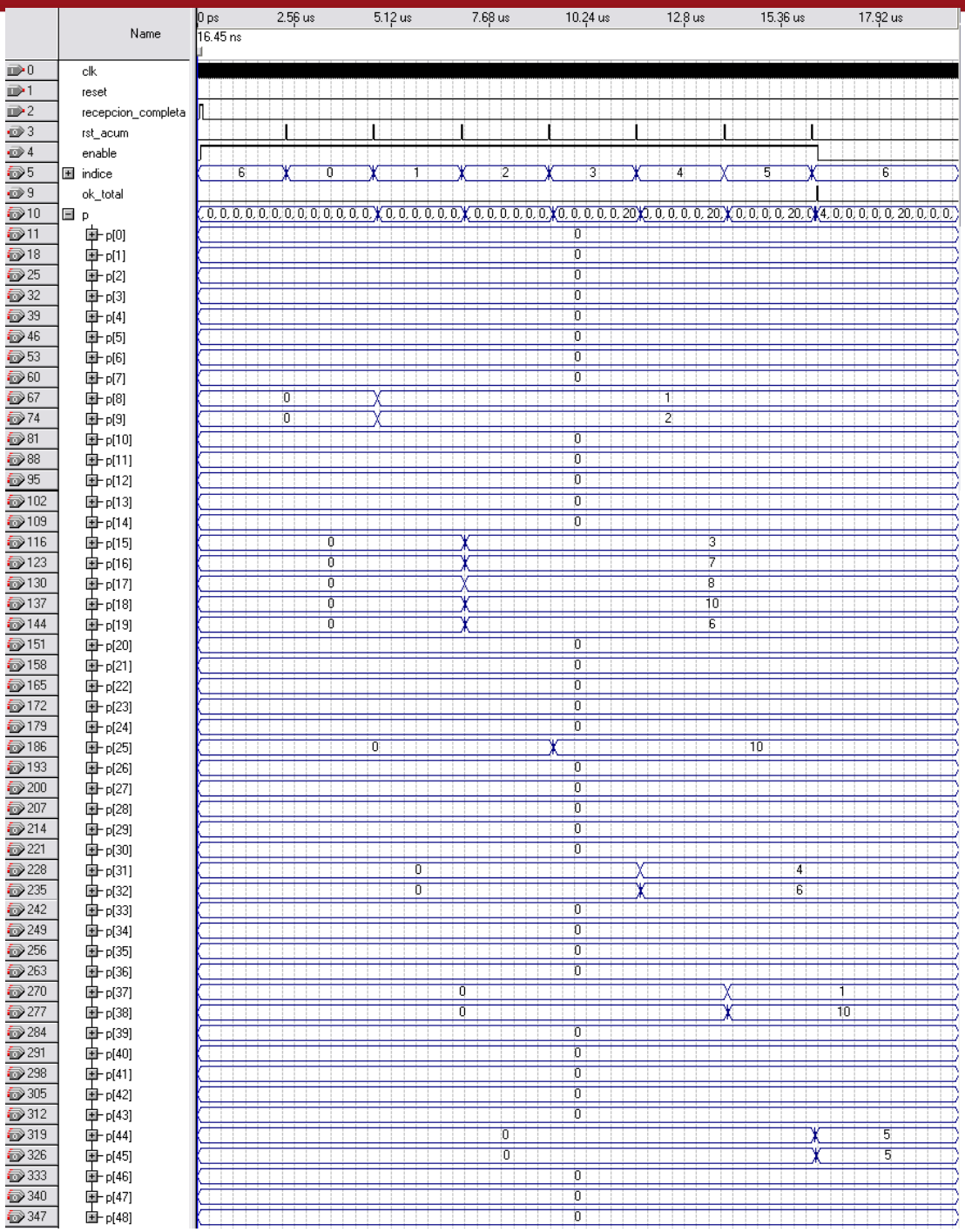


Figura 33. Simulación de la etapa de pre-procesamiento para una imagen almacenada en memoria del dígito 7.

Se observa en la figura anterior que el resultado obtenido coincide con la entrada de la simulación mostrada en la figura 32 (b). Esto nos permite concluir que el comportamiento de la etapa de pre-procesamiento es el esperado y permite extraer las características de la imagen según lo expuesto anteriormente.

La figura 34 muestra la simulación realizada a la etapa de post-procesamiento, en la cual se presentaron diversos casos, tanto reales como exagerados, que pudieran presentarse a la salida de la red neuronal. Así, la señal *a* contiene los valores de salida de la red neuronal y es la entrada de la etapa de post-procesamiento; la señal *entrada_valida* indica el inicio del procesamiento teniendo en cuenta que los valores a la entrada del bloque son los correctos; la señal *lectura_done* indica a la etapa previa que la lectura de sus resultados ya fueron leidos y que pueden continuar con su proceso; la señal *ok* indica el fin del procesamiento y que los resultados a la salida del bloque son correctos; la señal *resultado_leido* indica que los resultados generados fueron leídos por la etapa posterior y puede continuar con el proceso de las siguientes entradas; la señal *numero* contiene el resultado de la codificación del dígito reconocido; y la señal *valido* indica si el resultado contenido en número es válido o no.

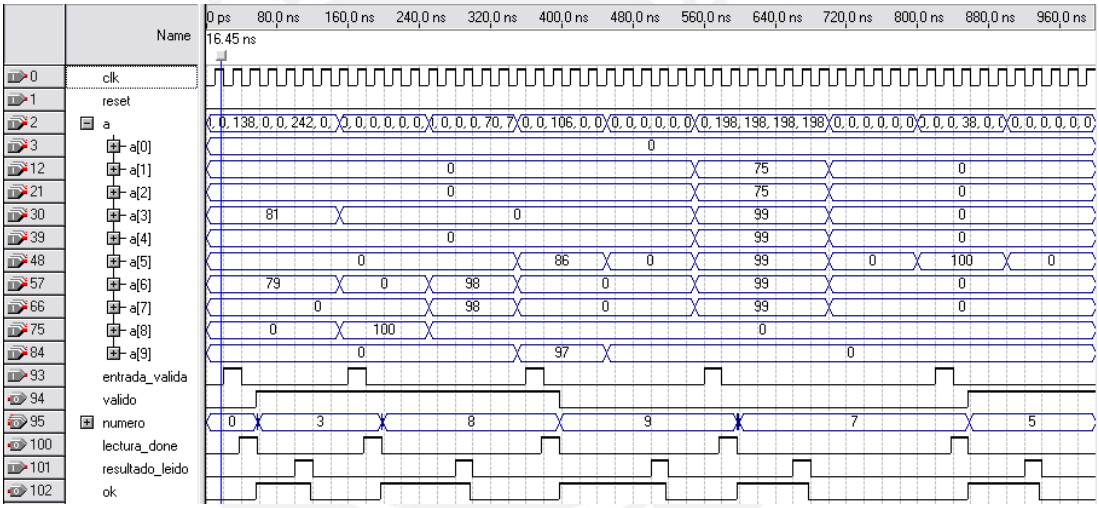


Figura 34. Simulación de la etapa de post-procesamiento.

Finalmente, se juntaron estas tres últimas etapas, pre-procesamiento, red neuronal y post-procesamiento, para realizar una simulación global del sistema. Esta es presentada en la figura 35; en la cual la señal *recepcion_completa* indica que los datos de los pixeles de la imagen del dígito a fueron almacenados en memoria y que el procesamiento puede empezar; la señal *numero* contiene, como se explicó anteriormente, el resultado codificado; y la señal *valido*, al igual que el anterior, indica si este resultado es válido o no. Como podemos apreciar, el dígito llega a ser reconocido y se indica que este resultado es válido con la activación del bit correspondiente.

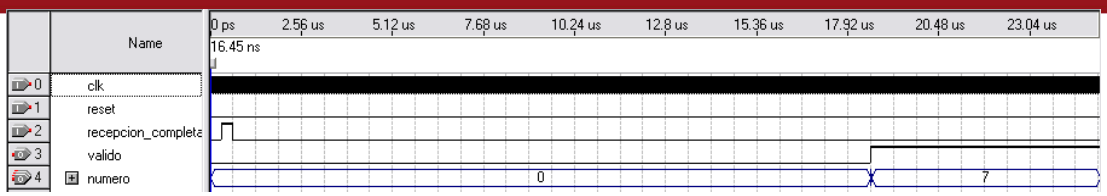


Figura 35. Simulación del sistema global para la imagen del dígito 7 almacenado en memoria.

5.3. Resultados

La descripción del circuito diseñado realizado mediante el lenguaje de descripción de hardware VHDL fue sintetizado utilizando las herramientas del Quartus II v.7.1 para el FPGA Cyclone II EP2C35F672C6 de Altera.

Los resultados del proceso de síntesis se muestran en el cuadro 2, donde se observa la diferencia existente entre la neurona utilizada en la capa oculta y la de la capa de 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 columna correspondiente a *Sistema* se refiere a la red neuronal unida con las etapas de pre-procesamiento y post-procesamiento.

Cuadro 2. Recursos utilizados en el FPGA [Elaboración propia].

Recursos	Neurona 1	Neurona 2	Red Neural	Sistema
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

La red neuronal fue implementada en la tarjeta de desarrollo DE2 elaborada por Terasic (figura 36) con una frecuencia de 50 MHz. La figura 37 muestra los resultados obtenidos tras la implementación utilizando el Signal Tap.II Logic Analyzer; donde la señal *numero* contiene el resultado del sistema; la señal *valido* el bit de validación; la señal *recepcion_completa* indica el inicio del procesamiento en el sistema; la señal *opera_out* indica el inicio del procesamiento para la red neuronal; y la señal *done_out* indica el fin del procesamiento para la red neuronal.

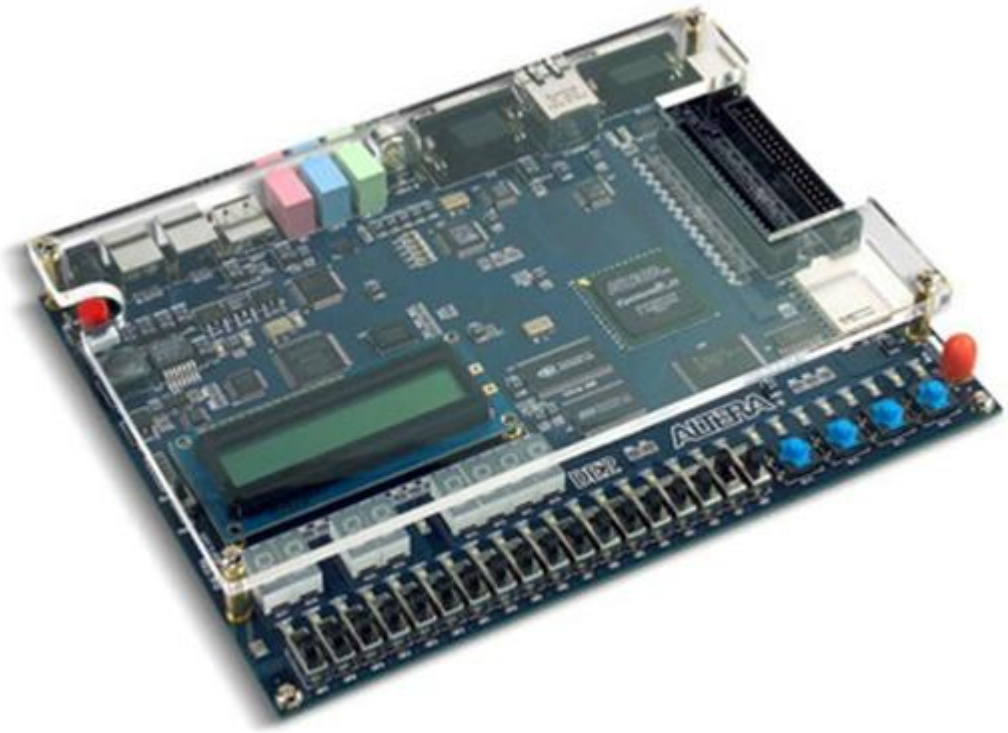
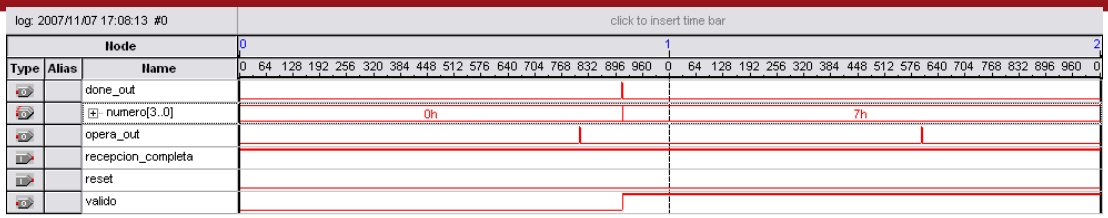
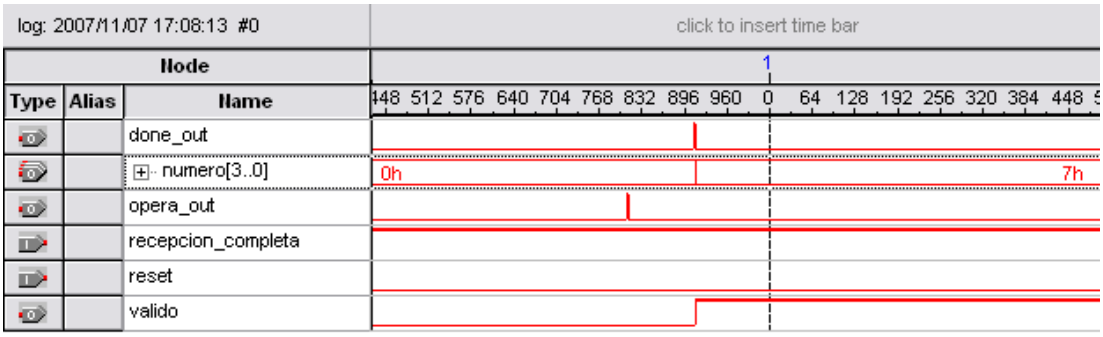


Figura 36. Tarjeta de Evaluación y Desarrollo DE2, Terasic, Altera.



(a)



(b)

Figura 37. Resultados del circuito implementado capturados por el Signal Tap II Logic Analyzer: (a) datos completos, (b) porción para el análisis.

Se observa que el circuito se comporta según lo esperado pues reconoce el dígito 7 cuya imagen estaba almacenada en una memoria. Además, se calculó el tiempo requerido para realizar este procesamiento teniendo como base los resultados capturados por el Signal Tap II Logic Analyzer.

Para esto observamos en la figura 36 que el cambio que se produce en la señal *numero* se realiza en la muestra 913; y al haber utilizado una frecuencia de 50MHz, el tiempo requerido para reconocer un dígito es:

$$\frac{913}{50MHz} = 18.26us$$

Del mismo modo, con ayuda de las señales opera_out y done_out que se activa en las muestras 810 y 911 respectivamente, calculamos el tiempo que utilizado por la red neuronal, el cual esta dentro de los 18.26us del tiempo total. El tiempo que tarda la red neuronal en procesar la información es:

$$\frac{912-810}{50MHz} = 2.04us$$

Comparándolo con los 7.089ms que tarda el Matlab 7.1 en realizar el reconocimiento del digito, calculados con un script, podemos ver la principal ventaja que presenta la implementación en hardware de las redes neuronales frente a su contraparte en software.

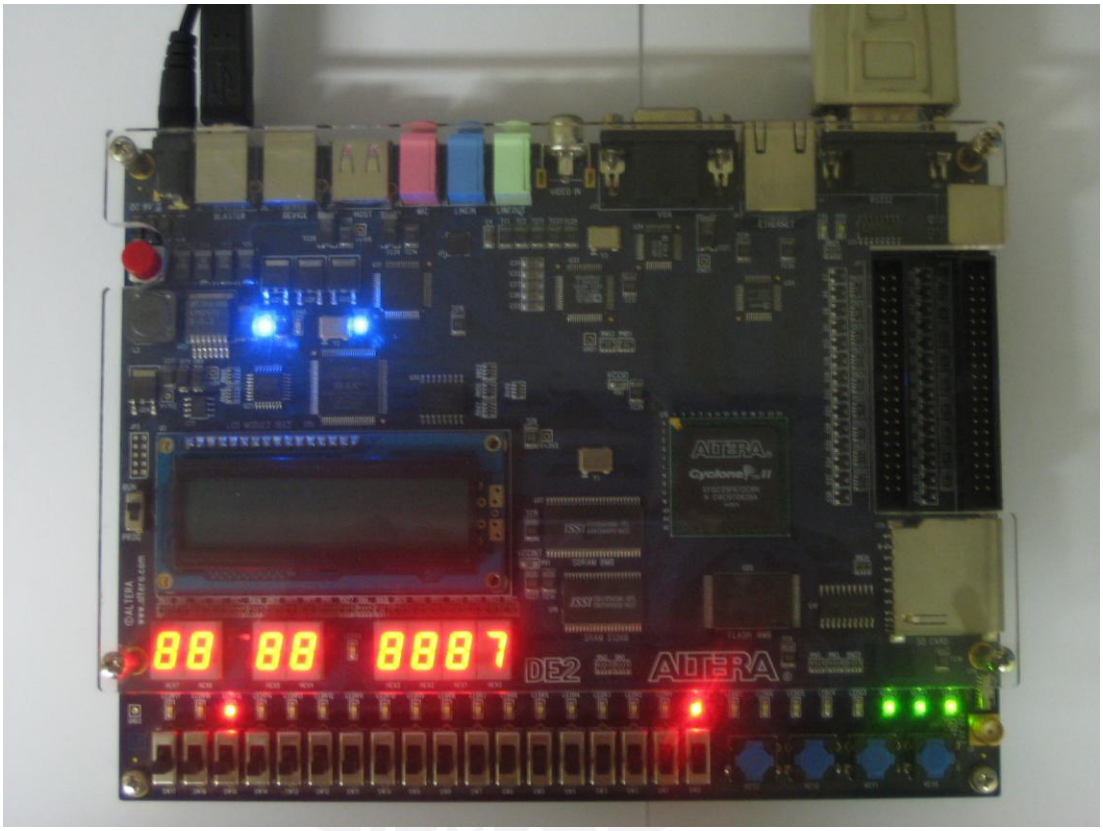


Figura 38. Diseño Implementado en la Tarjeta de Evaluación y Desarrollo DE2, en la cual se presentó como entrada una imagen correspondiente al dígito 7. Se observa tanto en el display como en los leds verdes que el dígito fue reconocido por la red neuronal.

CONCLUSIONES

1. Las redes neuronales pueden ser implementadas tanto en hardware como en software; sin embargo, una implementación en software no aprovecha el paralelismo presente puesto que realiza el procesamiento secuencialmente, es decir, instrucción por instrucción. Esto no ocurre con la implementación realizada en hardware, pues la mayor parte del procesamiento es realizado en paralelo.
2. La implementación de la arquitectura propuesta para la red neuronal artificial perceptron multicapa MLP en el FPGA Cyclone II de Altera y el correcto funcionamiento del sistema en el reconocimiento de dígitos manuscritos indican que el diseño de la red fue realizado con éxito y que el objetivo general de la presente tesis ha sido alcanzado.
3. El método aplicado en el diseño permite una fácil configuración del sistema puesto que todos los bloques conformantes 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 capaces de solucionar el problema para la cual fue entrenada. Por lo tanto, se ha obtenido una arquitectura base con la que es factible realizar futuros trabajos de investigación en los cuales se utilice la arquitectura propuesta.
4. El diseño realizado utiliza los recursos embebidos optimizados dentro del FPGA, lo cual permite aumentar las prestaciones del sistema y no usar recursos lógicos que pueden servir para otro sub-circuito dentro de un sistema más grande; esto hace que el diseño no sea completamente portable. Sin embargo, no lo vuelve no portable, pues estos bloques se encuentran presentes actualmente en la mayoría de FPGA's con lo cual bastará con cambiar estos módulos por los correspondientes en el nuevo FPGA. Adicionalmente, estos mismos bloques pueden ser descritos para que utilicen solamente recursos lógicos; y de este modo el sistema se volvería completamente portable.
5. 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.
6. 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.

7. 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.

RECOMENDACIONES

El objetivo de la presente tesis ha sido el desarrollo de una red neuronal para su implementación en hardware. Según vimos anteriormente, esta red no puede estar aislada, por lo que las etapas de pre-procesamiento y post-procesamiento son necesarias. En este trabajo se desarrollaron estas etapas tan solo para realizar la extracción de características de la imagen y para validar la respuesta obtenida a la salida de la red. Por lo tanto, es recomendable realizar un procesamiento digital a la imagen del dígito previo a la extracción de características así como también una etapa posterior que aumente la eficiencia del sistema.

Para un trabajo posterior en el cual se desee pasar la arquitectura propuesta a un chip integrado de aplicación específica de muy alta escala de integración (Very Large Scale Integrated Circuit, VLSI Circuit), los módulos que hayan sido diseñados para que utilicen bloques embebidos del FPGA deben ser rediseñados para que usen recursos lógicos en vez de estos últimos, puesto que así podrán ser interpretados correctamente en el proceso de síntesis. Del mismo modo, se deben considerar otros aspectos propios de la teoría y técnicas del diseño de circuitos integrados digitales como son la tecnología a utilizar, el tiempo de respuesta de los transistores, consumo de potencia, entre otros.

Para realizar el re-entrenamiento de la red para otra aplicación, se requiere que la red neuronal sea entrenada con un conjunto de datos que sean ejemplos representativos de la nueva tarea asignada. Así, lo ideal será contar con una base de datos adecuada como con la que se trabajó en la presente tesis; sin embargo, también puede generarse a través de la recopilación de estos ejemplos como una etapa previa al diseño y proceso de aprendizaje de la red. Lo que hay que tener en cuenta es que a mayor cantidad de datos, la red podrá responder mejor frente a nuevos patrones pues estará mejor preparada ante esta aplicación.

Cabe señalar que para la síntesis y simulación de la presente tesis fue suficiente el uso de una licencia web, pues no se utilizaron diseños propietarios del fabricante del dispositivo. Sin embargo, para la etapa de implementación del diseño y el uso del Signal Tap II Logic Analyzer fue requerida una licencia del tipo full versión. Por lo tanto, es recomendable contar con este tipo de licencia para la implementación de cualquier diseño, pues este tipo de herramientas son necesarias para que esta etapa se realice exitosamente.

BIBLIOGRAFIA

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

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

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

[4] David Castells i Rufas, Enric Pons, Jordi Carrabina
2005 "Implementación de un sistema OCR en un FPGA", V Jornadas de Computación Reconfigurable y Aplicaciones, JCRA 2005

[5] Laurence Fausett
1994 "Fundamentals of neural networks: architectures, algorithms, and applications", Prentice-Hall

[6] Simon Haykin
1994 "Neural Networks: a comprehensive foundation", Macmillan

[7] Emiliano Aldabas-Rubira
2002 "Introducción al reconocimiento de patrones mediante redes neuronales", Jornadas de Conferencias de Ingeniería Electrónica JCEE'02

[8] Andrés Pérez-Uribe
1998 "Artificial Neural Networks: Algorithms and Hardware Implementation" Extraído de: Bio-Inspired Computing Machines: Toward Novel Computational Architectures. D. Mange and M. Tomassini (Eds.), PPUR Press, pp. 289-316

[9] Anil K. Jain, Jianchang Mao
1996 "Artificial Neural Networks: A Tutorial", IEEE Computer and System, Page(s) 31-44

[10] Robert L. Harvey
1994 "Neural Networks Principles", Prentice-hall

[11] Amos R. Omondi, Jagath C. Rajapakse editores
2006 “FPGA Implementations of Neural Networks”, Springer

[12] José R. Hilera González, Juan P. Romero Villaverde, José A. Gutiérrez de Mesa
“Sistema de reconocimiento óptico de caracteres (OCR) con redes neuronales”, Departamento de Ciencias de la Computación, Universidad de Alcalá de Henares

[13] Richard G. Casey, Erci Lecolinet
1996 “A Survey of Methods and Strategies in Character Segmentation”, IEEE Transactions on pattern analysis and machine intelligence, Vol 18, No, 7, Page(s) 690-705

[14] Pedro Isasi Viñuela, Inés M. Galván León
2004 “Redes de Neuronas Artificiales, Un enfoque practico”, Pearson Educación

[15] Víctor Champac, Leandro Da Silva
2006 “IEEE CAS Tour”, Pontificia Universidad Católica del Perú, 14 y 15 de Diciembre

[16] Stephen Brown, Zvonko Vranesic
2005 “Fundamentals of Digital Logic with VHDL Design”, McGraw-Hill

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

[18] Altera’s Corporation
2005 “Cyclone II Device Handbook”, Volumen 1

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

[20] Howard Demuth, Mark Beale, Martin Hagan
2007 “Neural Network Toolbox 5 User’s Guide”, The MathWorks, Matlab

[21] M. Monge, J. Saldaña, C. Silva,
2007 “Diseño e Implementación de una Arquitectura para el cálculo de la Función Sigmoide en un FPGA Cyclone II de Altera”, XIV Congreso

Internacional de Ingeniería Electrónica, Eléctrica y de Sistemas - INTERCON, IEEE sección Perú, Piura, Perú.

[22] Paul D. Reynolds
2005 “Algorithm Implementation in FPGAs Demonstrated Through Neural Network Inversion on the SRC-6e”, A Thesis Submitted to the Graduate Faculty of Baylor University in Partial Fulfillment of the Requirements for the Degree of Masters of Science.

[23] Mario Andrés Raffo Jara
2008 “Diseño de un Sistema Flexible de Multiproceso orientado al tratamiento de Imágenes”, Tesis (Ing.) Pontificia Universidad Católica del Perú. Facultad de Ciencias e Ingeniería.