REDUCCIÓN DE RUIDO APLICANDO REDES NEURONALES ARTIFICIALES

Yasmany Prieto Hernández[1], Fidel Ernesto Hernández Montero, Alfredo Novales Ojeda
Universidad de Pinar del Río, Martí # 270, Pinar del Río, Cuba

ABSTRACT
This paper is related to the application of Artificial Neural Networks (ANN) for noise cancelation in electric digital signals. The use of ANN is a novel technique in noise reduction, which only needs the noisy version of the signal to obtain the clean one.
In this work is employed a FIR Multilayer Perceptron network. First is applied a software approach in Matlab, where the network is trained, and it's designed a strategy to reach the optimal network for noise cancelation in two cases study: simulated noise in Matlab and noise obtained through a Data Acquisition System connected to a sensor. Good results were achieved in the cancelation of both noises.
Second approach is used to describe the ANN in hardware. The optimal FIR Multilayer Perceptron architecture of Matlab is implemented in VHDL (Very High Speed Integrated Circuit Hardware Description Language) to download in a Xilinx XC3S1200E FPGA (Field Programmable Gate Array), setting as goal in the design highlight the parallelism of ANN operation. After been obtained the VHDL design, a noise cancelation application is simulated, with good results, considering the errors produced by the less accuracy of the numerical format of VHDL ANN. Finally the design is downloaded in the FPGA and is checked that works according to the results of the software approach.

KEYWORDS: Artificial Neural Networks, noise, VHDL, FPGA.

MSC: 68T20

RESUMEN
Este trabajo está relacionado con el uso de las Redes Neuronales Artificiales (RNA) para la cancelación de ruido presente en una línea de señal. La utilización de RNA, constituye una técnica novedosa en la reducción de ruido, que tiene como ventaja solo necesitar de la señal contaminada con el ruido para obtener la señal útil.
En el trabajo se emplea una red Perceptron Multicapa FIR. Inicialmente se utiliza un enfoque software desde Matlab, donde se efectúa el entrenamiento y se diseña una estrategia para obtener una red óptima en la cancelación de ruido para dos casos de estudio: ruido simulado en Matlab y ruido real obtenido a partir de un sistema de adquisición de datos acoplado a un sensor. Se obtuvieron resultados satisfactorios en ambos casos.
Un segundo enfoque es utilizado para describir las RNA en hardware. Se implementa la arquitectura de Perceptron Multicapa FIR óptima de Matlab en VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) para descargarla en una FPGA (*Field Programmable Gate Array*) XC3S1200E de Xilinx, teniendo como meta en el diseño resaltar el paralelismo con que operan las RNA. Luego de obtenida la red en VHDL, se simula una aplicación de cancelación de ruido, obteniéndose buenos resultados teniendo en cuenta los errores que aparecen producto de operar con una lógica numérica de menor precisión. Finalmente se descarga el diseño en la FPGA y se comprueba que opera de acuerdo a lo obtenido en el enfoque software.

1. INTRODUCCIÓN

Las señales durante su transmisión siempre se encuentran bajo la influencia de otras señales no deseadas. Incluso, cualquier procesamiento que se realice a una señal tiende a introducir perturbaciones desagradables en ella misma. A estas perturbaciones que contaminan la señal transmitida o procesada se le llama ruido y constituye una señal molesta que no guarda relación alguna con la útil [1].
Las señales contaminantes pueden traer como resultado la lectura de información falsa, la activación de alarmas, pérdida de comunicación entre transmisor y receptor, pérdida de datos, etc., en los sistemas electrónicos. Con un trabajo cuidadoso de ingeniería, se pueden reducir muchas señales indeseables; sin embargo estas siempre permanecerán con determinada magnitud imponiendo requerimientos a los sistemas.
Existen en general muchos métodos en la actualidad para la cancelación de ruido, básicamente pudieran ser agrupados en tres conjuntos bien diferenciados por las características y requerimientos que imponen a la hora de establecerse: los métodos clásicos, aplicados esencialmente en la rama de las comunicaciones eléctricas, cuando la señal deseada presenta un comportamiento primordialmente periódico, los métodos adaptativos de cancelación de ruido y los métodos clásicos de

[1] yprieto@tele.upr.edu.cu

procesamiento estadístico de señal. Estos métodos presentan condiciones particulares de aplicación, por ejemplo en los dos primeros métodos, es necesario, además de la señal contaminada con el ruido, obtener otra señal, en el primer caso correlacionada con la señal útil, y en el otro, con el ruido actuante. En algunas aplicaciones la obtención de estas señales adicionales es difícil y costosa. En el último grupo de métodos es necesario realizar suposiciones elementales como: linealidad, sistemas estacionarios, y sistemas con estadísticas de segundo orden; sin embargo, la mayoría de los sistemas físicos generan señales que no cumplen estas suposiciones, por lo que en estos casos no se conseguirían resultados óptimos.

El desarrollo y generalización que han venido experimentando los fundamentos de la Inteligencia Artificial en líneas de investigación muy diferentes, también ha permitido llevarlos a tareas de extracción de ruido. Siguiendo esta idea, ya se han establecido métodos basados en Redes Neurales Artificiales (RNA) para la cancelación de ruido en diversas aplicaciones, por ejemplo: en reconstrucción de imágenes holográficas [2], procesamiento de señales de voz [3], reconocimiento automático del habla [4], procesamiento de señales moduladas [5], señales biomédicas [6] y para cancelar el efecto generado por el desorden de las olas del mar en la señal de retorno de un radar [7]. De forma general la reducción de ruido es considerado como un proceso de mapeo de la señal ruidosa a la señal libre de ruido, la RNA se adaptaría para modelar las relaciones no lineales entre estas señales durante el entrenamiento [4]. En cuanto a cuál sería la arquitectura de red óptima para esta tarea existen opiniones diferentes: en [2] y en [5] se alcanzan mejores resultados a partir del Perceptron Multicapa al comparar su desempeño con respecto a una red recurrente de Elman y una ADALINE (*Adaptive Linear Neuron*) respectivamente en cuanto a los valores del MSE (*Mean Square Error*); en [7] se utiliza el Perceptron Multicapa y en [8] el Perceptron Multicapa FIR; en [3] se escoge como mejor arquitectura a la ADALINE de acuerdo a que redujo el ruido considerablemente de acuerdo al análisis de la SNR (*Signal to Noise Ratio*); en [4] se escoge como más efectiva de acuerdo al MSE una red neuronal profunda (*deep neural network*) recurrente, que consiste en redes multicapas de varias capas ocultas y gran cantidad de neuronas o unidades sinápticas en cada capa oculta, con conexiones recurrentes temporales en una de las capas ocultas. Una tendencia encontrada en estos sistemas de reducción de ruido basados en redes neuronales es la inclusión de la variable tiempo en el análisis, lo que permite reconocer patrones temporales en las señales, en [3], [4] y [6] las entradas a la red son la muestra actual de la señal y sus valores desplazados temporalmente, en [2] y [4] se utiliza una red de Elman, que posee una entrada a cada neurona que es la salida de esta realimentada a partir del estímulo anterior, pero estos arreglos solo se reflejan en cuanto a la operación de la red, en [8] el enfoque temporal se extiende a su entrenamiento, este último enfoque fue el utilizado en este trabajo.

El objetivo de esta investigación ha sido aplicar una arquitectura de RNA al proceso de cancelación de ruido en señales reales; y en el camino de obtener un dispositivo que realice esta tarea, se propone el diseño e implementación de esta red en *hardware*.

2. PRINCIPIOS FUNDAMENTALES DE LAS RNA Y SU IMPLEMENTACIÓN HARDWARE

2.1 Introducción a las RNA

Una red neuronal es un procesador distribuido de forma masivamente paralela, propensa, de manera natural, a almacenar el conocimiento debido a la experiencia, y hacerlo disponible para su uso. Esta red se relaciona con el cerebro en dos cuestiones: el conocimiento se adquiere a través de un proceso de aprendizaje y la fuerza de la conexión interneurona, conocida como peso sináptico, la cual se emplea para almacenar el conocimiento [9].

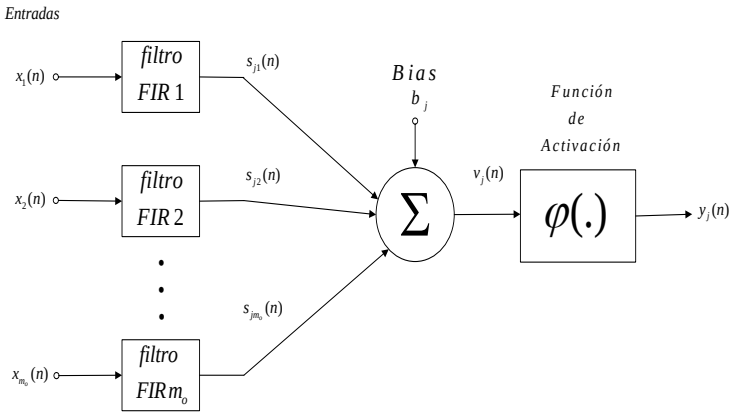

Figura 1. Neurona FIR

En este trabajo se utiliza la arquitectura de RNA Perceptron Multicapa FIR, esta consiste en un conjunto de unidades sensoriales que constituyen la capa de entrada, una o más capas ocultas y una capa de salida. La señal de entrada se propaga a través de la red en dirección hacia adelante, capa por capa, hasta obtenerse la salida. Además la red es capaz de asumir el tiempo como variable de forma distribuida por toda su extensión ya que todas sus sinapsis son unidades de procesamiento temporal.
La arquitectura Perceptron Multicapa FIR tiene como elemento básico la neurona FIR (modelo espacio temporal de la neurona) que puede estudiarse en [9]. Este modelo de neurona que se observa en la Figura 1 puede ser interpretada como un filtro neuronal o un filtro FIR no lineal. En esta representación pueden observarse tres elementos primarios que son: los filtros FIR sinápticos (los cuales permiten el procesamiento temporal de la neurona), el sumador o combinador lineal y la función de activación que en este caso es una sigmoide bipolar.
El algoritmo de entrenamiento del Perceptron Multicapa FIR, denominado Backpropagation Temporal que puede verse en [9], es supervisado y está basado en una regla de aprendizaje por corrección del error, generalización del algoritmo LMS (*Least Mean Square*). Este algoritmo tiene en cuenta el carácter temporal de las señales de trabajo.

2.2. Elementos básicos para la aplicación de redes neuronales artificiales en la cancelación de ruido

La idea del método de aplicación escogido es bien sencilla y consiste, en su forma general, en un enfoque muy similar al empleado por las arquitecturas neurales implicadas en el reconocimiento de patrones.
Esencialmente el proceso estribaría en entrenar la arquitectura de RNA elegida, de forma que aprenda la estadística del proceso aleatorio, inmerso en el cual se encuentra la señal a predecir. Desde un enfoque de reconocimiento de patrones, a la red se le suministra, como patrones de entrenamiento, muestras de determinado ruido, asumiendo obtener a la salida el valor esperado de ese proceso aleatorio. Con estos datos, se entrena la red varias veces [8].
Inicialmente se obtienen varios patrones de determinado tipo de ruido, blanco Gaussiano o ruido de un sensor, en dependencia del que se desee cancelar, para posteriormente emplearlos durante el entrenamiento de una arquitectura neural (ver Figura 2). Durante el entrenamiento, estos grupos específicos de patrones estarían dados a la entrada y se tendría como salida deseada el valor esperado para cada patrón de ruido y así conformar el correspondiente "mapeo" de entrada/salida inherente al aprendizaje supervisado.
Una vez entrenada la arquitectura, esta es operada presentándole a la entrada la señal útil contaminada por el ruido (del mismo tipo con que fue entrenada) y, si la red entrenó correctamente, se obtendrá a la salida un dato que se adecuaría con el valor medio o esperado (señal deseada).

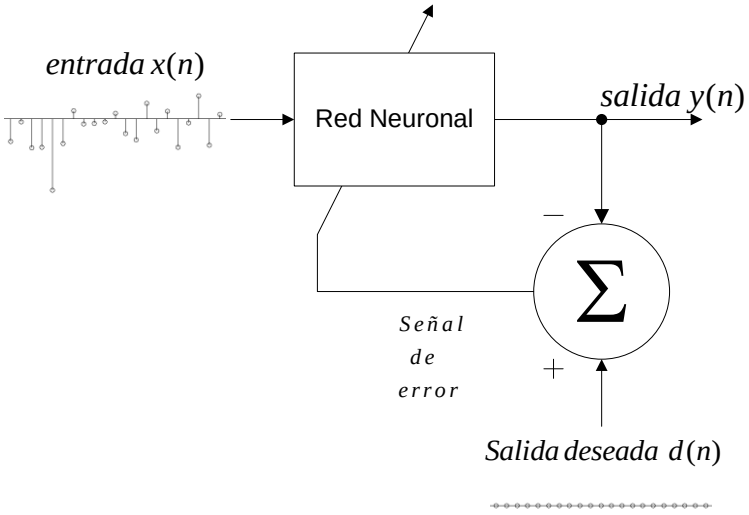

Figura 2. RNA y señales de entrenamiento

2.3. Implementación de Redes Neuronales Artificiales en hardware

Las simulaciones de software son útiles para investigar la capacidad de modelos de redes neurales y crear nuevos algoritmos, pero la implementación hardware sigue siendo esencial para aprovechar al máximo el paralelismo inherente de las redes neurales [10], en este trabajo se utiliza una FPGA para esta implementación.
Las FPGA introducidas por Xilinx en 1985 [11], son el dispositivo programable por el usuario de más general espectro. Consisten en una matriz bidireccional de bloques configurables que se pueden conectar mediante recursos generales de interconexión. En la FPGA se programan los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques [12].
Actualmente ha crecido el interés de implementar RNA con FPGA. Aunque las FPGA no consiguen la potencia o la frecuencia de reloj de chips específicos, sí proveen un incremento en la rapidez del tiempo de ejecución con respecto a la simulación software. Ya que las redes neurales son intrínsecamente estructuras paralelas las arquitecturas paralelas resultan más óptimas que las arquitecturas serie para su implementación. Hasta ahora una restricción principal en el enfoque FPGA ha sido el de la densidad de lógica limitada que posee [10] y la cantidad de multiplicadores [13]. Sin embargo las tecnologías de FPGA avanzan rápidamente en cuanto a aumentar la densidad de compuertas por área y la velocidad, por eso tienden a ser las mejores candidatas en la implementación de Redes Neurales entre otras alternativas.

3. MATERIALES Y MÉTODOS

En la primera parte de la investigación, desde el enfoque software, para el entrenamiento y operación de la red, la programación y obtención de resultados se realizó en Matlab 7.0.1 de MathWorks. Primeramente se revisó el *toolbox* que este posee, y se estudiaron los modelos de redes y procedimientos de aprendizaje. En el *toolbox* de RNA no se encontró el entrenamiento mediante Backpropagation Temporal para Perceptron Multicapa FIR, por lo que se pasó a programar la operación y entrenamiento de la red, además de concebir la red dentro de Matlab.

El índice que se toma para mostrar el comportamiento efectivo de la red es el coeficiente de correlación entre la señal a la salida de la red neural (esto es durante la operación) y la señal útil sin contaminar que se emplea como señal de prueba. Este parámetro da una medida de cuán bien se explican las variaciones en la salida de un sistema debido a las entradas. (En este caso como la señal a la entrada de la red está constituida fundamentalmente por la señal útil y el ruido, este parámetro mediría cuánto de la señal útil está presente a la salida de la red).
Se entrenaron varias combinaciones de la arquitectura *Perceptron* Multicapa FIR de dos capas ocultas, con neuronas en la capa oculta uno de una sola entrada (entrada a la RNA) y una neurona en la capa de salida, variando tanto la cantidad de neuronas en cada capa oculta, como la cantidad de elementos de los filtros FIR sinápticos.
Para obtener la RNA óptima, durante el entrenamiento se presenta a su entrada una muestra de la señal ruidosa de valor esperado cero. Esta muestra se desplaza en el próximo instante de tiempo, y entra a la red la muestra siguiente, mientras como salida esperada se utiliza el valor cero. Así, se pasa 300 veces a través de la red toda la señal que contiene el ruido, reajustando los filtros sinápticos. Después de cada entrenamiento, se pasa a una etapa de operación, la cual consiste en presentar a la entrada de la red neural una señal sinusoidal contaminada con ruido del mismo tipo con que se realizó el entrenamiento. Esta señal entra, muestra a muestra, de la misma forma que durante el entrenamiento, por lo que en el cálculo de la salida no solo intervendría la muestra actual sino también las retrasadas.
Una vez obtenida la señal a la salida de la red, se calcula el coeficiente de correlación entre esta y la señal sinusoidal. Cuando se encuentra un coeficiente mayor que el último encontrado se guarda la topología de la red y el valor de cada peso sináptico de esta. La red óptima es aquella que presentó un valor de coeficiente de correlación mayor.Un paso adelante en esta investigación sería la implementación hardware de la RNA. Este paso permitiría aprovechar al máximo el paralelismo inherente en las redes e iniciar el camino en el desarrollo de un dispositivo hardware capaz de cancelar ruido en señales reales.El desarrollo de modelos de redes neuronales en hardware no es directamente implementable en el Silicio como en el plano software. El diseñador debe concentrarse en otros problemas, como son: tiempo de procesamiento, precisión para representar los datos y realizar las operaciones, espacios de memoria, nivel de paralelismo, flexibilidad de diseño [14]. Además debe cumplir con restricciones provenientes del circuito como: consumo de área y recursos del dispositivo, y el hecho de hacia dónde mueve el mercado a cada tecnología [15]. Una de las principales metas del diseño fue resaltar el paralelismo con que trabajan las RNA según su modelo matemático, basado en los recursos establecidos del hardware que se posee para su desarrollo.
En la segunda parte de la investigación, se decidió diseñar la RNA en VHDL para su implementación en FPGA, dadas las ventajas expuestas anteriormente. Para esto se usó el ISE (*Integrated Software Environment*) 12.4, el cual es un *software* de Xilinx que permite la creación de diseños y su simulación, síntesis e implementación en FPGA. Para validar la RNA se realizaron dos experimentos, uno mediante simulación en ISIM, que es el simulador de ISE y el otro, descargando el diseño en la FPGA y capturando las salidas de la red neuronal a través de ChipScope Pro 12.4, un analizador lógico embebido.*ChipScope Pro* fue desarrollado por *Xilinx*, y permite verificaciones en tiempo real para las FPGAs insertando

unos núcleos para depuración (*debug*) software de bajo perfil (ILA, IBA, ATC2 y VIO) en el diseño. Una vez esos núcleos (o *cores*) han sido colocados y conectados permiten al usuario capturar señales de datos en el chip, en tiempo real. Los datos capturados son enviados fuera del chip por algunas de las conexiones habilitadas para ello y analizados. Los recursos lógicos de la FPGA se utilizan para implementar el circuito de disparo y los bloques de memoria de la FPGA son utilizados para implementar la capacidad de almacenamiento. La herramienta *ChipScope Pro* consta de tres componentes: una herramienta para incluir los núcleos (*CoreInserter*), un generador de núcleos (*CoreGenerator*), y el analizador de señales *ChipScope Pro* (*ChipScope Pro Analyzer*) [16].

4. RESULTADOS Y DISCUSIÓN

Para el entrenamiento y obtención de la RNA se dividió el trabajo en dos; en la primera parte del trabajo se entrenó la red con Ruido Blanco obtenido a partir de la función "*randn*" de Matlab; en la segunda parte, se utilizó para el entrenamiento muestras reales de ruido de un sensor de aceleración.

4.1. Entrenamiento y operación en un medio con ruido blanco simulado

La red Perceptron Multicapa FIR óptima en este caso se encontró al entrenar las combinaciones posibles de la arquitectura que se obtienen de variar de 1 a 20 la cantidad de neuronas de las capas ocultas y de 1 a 10 los elementos de los filtros sinápticos.
La Tabla I muestra los mejores resultados obtenidos con diferentes configuraciones. Como puede verse, la mejor configuración resultó aquella con la que se obtuvo un coeficiente de correlación de 0.83. En operación, a la entrada de esta RNA se le pasa la señal mostrada en la Figura 3, la cual consiste en una señal sinusoidal de frecuencia 1 Hz, contaminada por el ruido.

Tabla 1. Mejores configuraciones obtenidas en la cancelación de ruido

Neuronas capa oculta 1	Neuronas capa oculta 2	Elementos filtro sináptico	Coeficiente de correlación
2	7	4	0.671
16	2	6	0.753
19	3	6	0.821
15	3	6	0.83

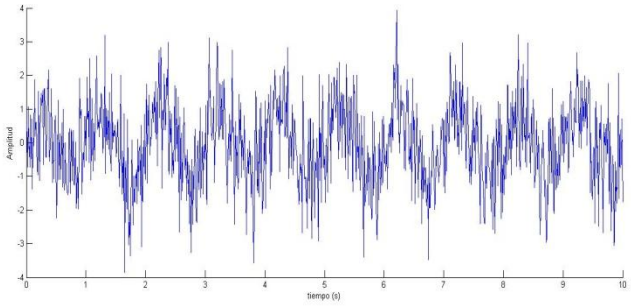

Figura 3. Señal contaminada con ruido blanco simulado

A la salida de la red se obtuvo la señal mostrada en la Figura 4, donde se puede apreciar la similitud de esta con la señal útil y la disminución de la amplitud del ruido. También, como característica de la salida de la RNA, hay que tener en cuenta la aparición de una componente de directa.

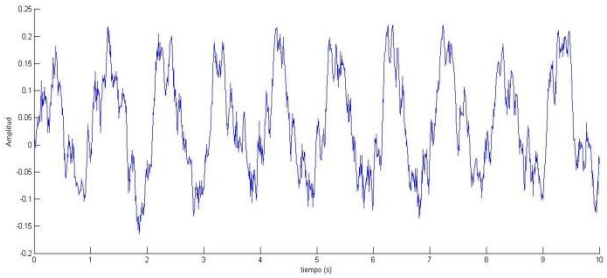

Figura 4. Señal a la salida de la red

4.2. Entrenamiento y operación a partir de ruido real

El ruido real manejado en la tarea de cancelación de ruido por parte de la RNA para entrenamiento y operación se obtuvo a partir de un Sistema de Adquisición de Datos acoplado a un sensor de aceleración. Las muestras se corresponden con el sensor en reposo y son cargadas por Matlab desde un archivo guardado en la computadora.

El entrenamiento y la operación de la red esta vez se realizaron tomando como valores iniciales los pesos sinápticos obtenidos anteriormente como mejores configuraciones para Ruido Blanco (estos valores iniciales permiten una convergencia más rápida del algoritmo de entrenamiento hacia la red más óptima), teniendo en cuenta las características de las RNA de adaptarse a nuevos ambientes de trabajo. La RNA óptima de acuerdo al criterio experimental basado en el cálculo de la correlación fue la misma que mejores resultados ofreció al operar con el ruido blanco simulado. Con esta se obtuvo un valor del coeficiente de correlación de 0,971. Así, la RNA obtenida tiene igual topología a la que se utilizó para operar con Ruido Blanco simulado, sin embargo, los valores de los filtros se reajustaron para trabajar ahora con el nuevo tipo de ruido.

En la Figura 5 se muestra la señal útil que es un seno de 5.4 Hz, contaminada con el ruido real, y en la Figura 6, la señal a la salida de la red, pudiéndose observar la reducción del ruido.

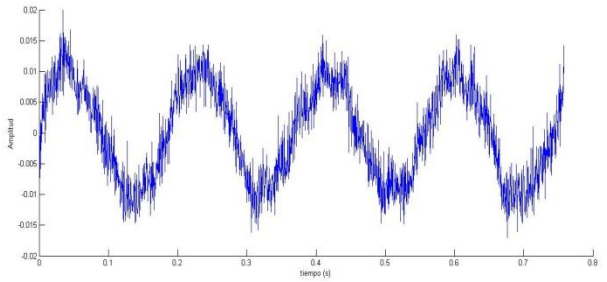

Figura 5. Señal contaminada con ruido real

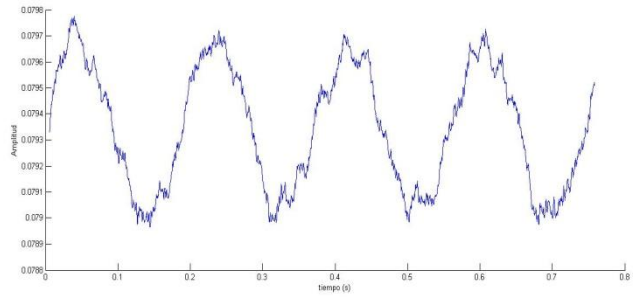

Figura 6. Señal a la salida de la RNA

Como pudo apreciarse el comportamiento de ambas RNA ante las dos clases de ruido fue la acción de reducirlos, por lo que el entrenamiento fue fructífero.

4.3. Implementación en FPGA de la RNA

Para la implementación del diseño se contó con una tarjeta Nexys 2 basada en una FPGA XC3S1200E de la familia Spartan 3E de Xilinx. Dentro de los recursos más importantes empleados en el trabajo que posee esta FPGA están: 28 bloques multiplicadores que aceptan dos números binarios de 18 bits, con o sin signo y 28 bloques RAM de una profundidad de 1024 posiciones de memoria. Estos recursos tienen un peso significativo en el camino que se tomó para el diseño, ya que ambos están implicados en el nivel de paralelismo alcanzable. El poder utilizar al menos un bloque RAM para representar la función de activación y un multiplicador por neurona, permite que cada una opere por separado.

a. Neurona FIR en FPGA

El modelo *Perceptron* Multicapa FIR tiene como elemento básico la neurona FIR. Estas neuronas se encuentran formando capas consecutivas, totalmente interconectadas las de una capa con las de la otra, siendo así un modelo repetitivo, donde solo cambian los valores de los pesos sinápticos y la cantidad de entradas que poseen las neuronas. Este hecho permite que el diseño de la red neuronal pueda ser simplificado al diseño de una neurona o varios tipos de neuronas parametrizables, para luego multiplicarlas y sincronizarlas hasta desarrollar toda la red.
La concepción de la neurona FIR hardware parte de la utilización de módulos que asemejen las operaciones del modelo espacio temporal de la neurona representado en la Figura 1. De esta forma se llegó al diseño mostrado en la Figura 7.
En el diseño desarrollado pueden observarse diferentes módulos. Primero se encuentra el módulo *Desplazador*, el cual se encarga de tomar las señales de entrada a la neurona, que pueden ser la salida de las m neuronas de la capa precedente o la entrada a la red neuronal. Luego desplaza temporalmente las señales que han entrado anteriormente por esta misma vía, de acuerdo a los filtros FIR en el modelo temporal de la neurona. Más tarde, las señales desplazadas y los pesos correspondientes para su multiplicación que se encuentran almacenados en este módulo son entregadas al módulo *Multiplicador* de forma serie. La combinación de los módulos *Multiplicador* y *Acumulador* asemeja la operación de una unidad MAC (*Multiply-Accumulate*).
El módulo *Campo Inducido* recibe el valor final de la suma de las salidas de los filtros FIR sinápticos de parte del Acumulador y le adiciona el *bias.*
Con este valor final, se accede al módulo *Función de Activación*, el cual modela a través de una memoria ROM la función de activación de la neurona. Algunas señales importantes en este diseño son: *clk*, que es la señal de reloj; *Habilitación neuronas*, le indica a la neurona que puede operar y *Dato listo* que informa cuando la neurona tiene lista la salida.
Se utilizó el formato numérico punto fijo para la representación de los pesos sinápticos y las señales dentro de la neurona, para las operaciones algebraicas y para la modelación de la función de activación. Escogiéndose la cantidad de bits para la parte entera y fraccionaria de acuerdo a las características de amplitud máxima, mínima y la precisión de las señales dentro de la red.

b. Modelo VHDL de la RNA

Una vez diseñadas las neuronas que formarán parte de la Red neuronal, solo faltaría conectarlas y sincronizar su funcionamiento. Como primer paso, las neuronas que se encuentran en una misma capa y operan de forma paralela entre sí, son manejadas por un único módulo *Control*. Este módulo *Control*, anteriormente se encontraba interno en el *Desplazador* para el caso de una neurona, pero al manejar señales comunes a todas las neuronas de una capa (*Habilitación neuronas, Dato listo*), se utiliza uno solo.
La actividad entre las capas está dirigida por los módulos Activador Capa 1, Activador Capa 2 y Activador Capa 3, que habilitan el funcionamiento de las neuronas pertenecientes a cada capa teniendo en cuenta que cada capa necesita como entradas las salidas de la capa precedente. De forma general, los módulos Activador Capa son síncronos, tienen una o dos entradas, conectadas a las salidas Dato listo de la Capa anterior y de la misma Capa, para habilitar o inhabilitar respectivamente a las neuronas de esta capa.

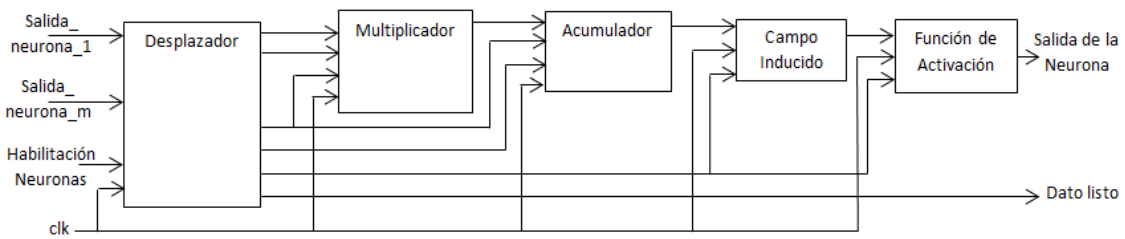

Figura 7. Representación en bloques de la Neurona FIR

En la Figura 8 se muestran los elementos conformadores de la red neuronal VHDL: las flechas gruesas simbolizan las entradas y salidas de las capas de la RNA, las líneas continuas representan las conexiones asociadas a la habilitación de las capas, y las líneas discontinuas las conexiones del reloj.
Se analizaron dos variantes para la implementación de la RNA. El primer caso, en que los datos proviniesen de otro sistema en tiempo real, con un período de muestreo, aquí la red operaria cada vez que una entrada estuviese lista para ser procesada, de forma que se habilitase cada vez que hubiese un dato nuevo a la entrada. Este tipo de procesamiento exige que cada capa de la red opere de forma serie con respecto a las demás. Una segunda opción era aumentar el paralelismo entre las capas y disminuir el tiempo de procesamiento total, aquí los datos estarían almacenados en memoria y el tiempo entre la lectura de una muestra y otra se correspondería con el tiempo de operación de la red. Se utilizó un método de *pipeline*, de tres etapas, cada una correspondiendo con la actividad de cada capa, en este caso la etapa crítica es la capa 2, que impone la frecuencia máxima de trabajo. Esta segunda opción fue la escogida en el trabajo.

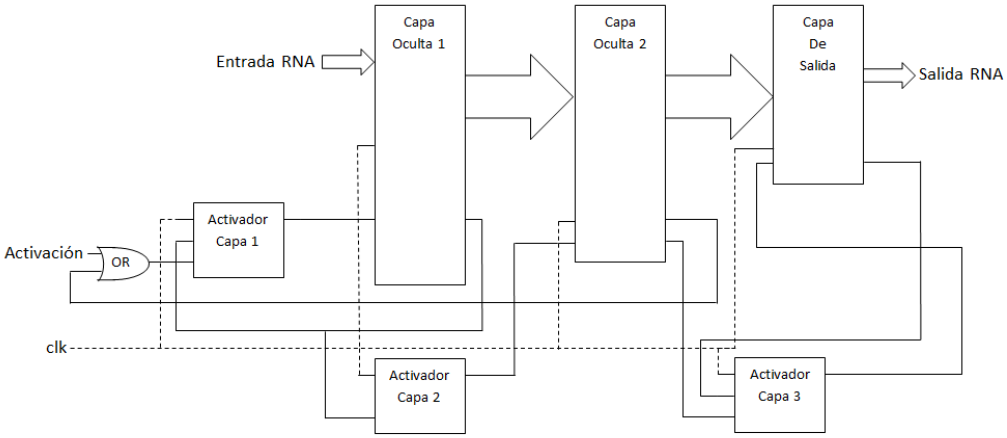

Figura 8. Módulos internos de la RNA VHDL

En la Figura 9 se muestra la operación de la RNA, se puede observar de forma sombreada los tiempos en que cada capa está habilitada y el dato que se está procesando. Este procedimiento permite que al menos dos capas estén operando a la misma vez, aumentando el nivel de paralelismo.
Terminado el diseño, se pasa a sintetizarlo. A través del ISE se obtiene un reporte de la utilización de recursos de la FPGA, el cual se puede ver en la Tabla 2.

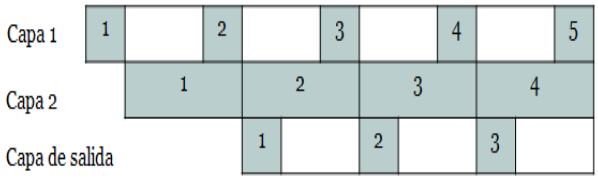

Figura 9. Método de Pipeline

Puede observarse en esta Tabla, como recursos críticos, los bloques RAM y los multiplicadores empotrados, cuya disposición y cantidad fue planificada al principio del diseño.
Después de obtener la RNA en la FPGA, se pasa a una etapa de experimentación. Inicialmente se simuló una aplicación para cancelación de ruido. Se creó un *test bench* (fichero VHDL mediante el cual se indica a la herramienta el diseño a simular y las entradas para llevar a cabo la simulación) asociado al diseño de la RNA, como entrada se colocó la señal ruidosa con que se operó la red, representada en la Figura 5. Al correr la simulación se obtuvo la salida de la RNA hardware, estos valores son extraídos del ISIM (simulador de ISE 12.4) e importados a Matlab manualmente, con la característica de que la salida del ISIM es binaria y hay que transformarla a punto flotante para las operaciones en Matlab. Luego de cargada la señal en Matlab esta es graficada, obteniéndose el resultado mostrado en la Figura 10.

Tabla 2. Resumen de recursos de la RNA VHDL tras síntesis

Recursos Lógicos	Utilizados	Disponible	%
Number of Slices	3064	8672	35
Number of Slice Flip Flops	3671	17344	21
Number of 4 input LUTs	3093	17344	17
Number of bonded IOBs	36	250	14
Number of BRAMs	26	28	92
Number of MULT18X18IOs	19	28	67
Number of GCLKs	4	24	16

Si se compara este resultado con el obtenido como salida de la RNA en la cancelación del mismo ruido en Matlab en la Figura 6, puede observarse que ambas señales tienen características muy similares. Ambas se encuentran oscilando aproximadamente entre los mismos valores, con una ligera diferencia en el valor de la componente de directa. Sin embargo, es fácil constatar que la señal resultante de la simulación en ISIM tiene niveles de ruido tan altos con respecto a la señal útil, como el existente en la señal a la entrada de la RNA (ver Figura 5).

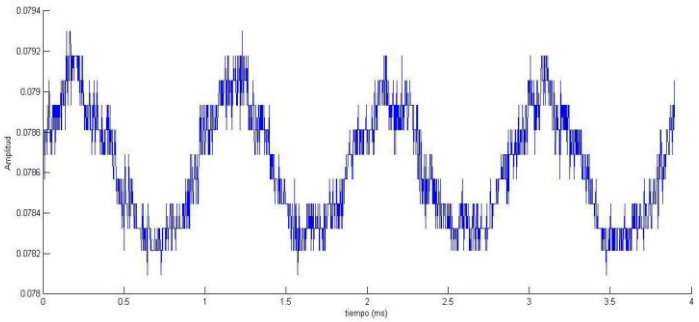

Figura 10. Resultados de la simulación ISIM de la RNA

Este nuevo ruido se debe al hecho de operar con lógica discreta limitada, dado por los errores que se cometen al cuantificar las señales de entrada, los parámetros del sistema (como son los coeficientes de los filtros sinápticos FIR), además de la propagación de estos errores en las operaciones aritméticas y la evaluación de las funciones de activación que intervienen en cada neurona. Una forma de reducir este ruido sería implementando las funciones de Activación con más bloques de memoria RAM; esto pudiese cumplirse usando una tarjeta con mayor cantidad de este recurso.

Un segundo experimento se realizó para comprobar que el funcionamiento de la Red neuronal implementada en la FPGA estuviese de acuerdo con la simulación anterior. Se adicionaron al módulo RNA otros módulos para simular las entradas y se utilizó el *software* ChipScope Pro 12.4 que forma parte del programa universitario de ISE 12.4 para la lectura de las salidas de la RNA en la FPGA.

Fue necesario configurar el núcleo ILA (Análisis de Lógica integrada) de ChipScope, que permite el acceso a señales internas de la FPGA mediante el componente *ChipScope CoreInserter* que inserta el núcleo anterior dentro del diseño, y el componente *ChipScope Pro Analyser*, que permite observar las señales obtenidas a partir de ILA.

Al analizar las salidas obtenidas a través de ChipScope en la computadora y la simulación se comprobó que ambas coincidían, por lo que se pudo validar el diseño directamente en la FPGA.

5 CONCLUSIONES

Resultó claro constatar que las RNA son una herramienta útil en la cancelación de ruido. Capaces de operar y entrenarse con señales aleatorias, tanto simuladas como del mundo real. Las RNA cancelaron parte del ruido que contaminaba la señal, cumpliendo con su objetivo fundamental, sin embargo hay que precisar que estas no pueden asumir solas toda la tarea de obtención de la señal pura a partir de la señal contaminada, ya que a la salida de la red se presentó en ambos casos de la investigación la aparición de una componente de directa y una reducción en la amplitud de la señal pura.

Logró implementarse en una FPGA el Perceptron Multicapa FIR que resultó óptimo en la tarea de reducción de ruido. Para ello hubo que tener en cuenta las características particulares de este dispositivo, que llevaron a una estrategia de diseño que potenciara el paralelismo presente en las RNA. Al comparar los resultados en la cancelación de ruido entre la red obtenida en VHDL y la de Matlab, puede concluirse que las señales obtenidas a la salida son claramente cercanas, de acuerdo al rango de valores y la forma de onda que presentan. La salida de la RNA en VHDL contiene un nuevo tipo de ruido: ruido de cuantificación, debido a que esta Red trabaja con una precisión menor que la de Matlab. Una vía de mejorar estos resultados sería aumentar la precisión en la implementación *hardware* de las funciones de activación, lo que puede lograrse mediante la utilización de otra tarjeta FPGA, que posea mayor cantidad de BRAM.

RECEIVED JULY 2013
REVISED OCTOBER, 2013

REFERENCIAS

[1] LATHI, B.P. (1965): **Signals, Systems and Communication**. John Wiley & Sons, Inc., Minnesota.
[2] BADRI L., (2010): Development of Neural Networks for Noise Reduction. **The International Arab Journal of Information Technology**, 7, 289-294.
[3] FAH L. B., HUSSAIN A. & SAMAD S. A., (2000): Speech Enhancement by Noise Cancellation Using Neural Network. **TENCON 2000. Proceedings**, 1, 39-42
[4] MAAS A. L., LE Q. V., O'NEIL T. M., VINYALS O., NGUYEN P., NG A. Y. (2012): Recurrent Neural Networks for Noise Reduction in Robust ASR, Stanford University, CA. Disponible en Web: **http://cs.stanford.edu/people/ang/?portfolio=recurrent-neural-networks-for-noise-reduction-in-robust-asr** Consultado 20-10, 2013.
[5] VYAS P. K., RAWAT P., KHATRI S. (2011): Back Propagation Based Adaptive Noise Chancellor. **International Journal of Computer Technology and Electronics Engineering *(*IJCTEE**), 1, 14-16.
[6] MATEO SOTOS J., SÁNCHEZ MELÉNDEZ C., VAYÁ SALORT C., CERVIGON ABAD R., RIETA IBÁÑEZ J. J. (2007): A Learning Based Widrow-Hoff Delta Algorithm for Noise Reduction in Biomedical Signals. **IWINAC 2007**, Springer-Verlag Berlin Heidelberg, Part I, LNCS 4527, 377–386.
[7] HAYKIN, S. (1966): Neural Networks Expand SP's Horizons. **IEE Signal Processing Magazine**, 13, 24-29.
[8] HERNÁNDEZ MONTERO, F. E. (2000): **Aplicación de Redes Neuronales Artificiales en la Cancelación de Ruido.** Tesis de Maestría, Facultad de Automática, Instituto Superior Politécnico José Antonio Echeverría, La Habana.
[9] HAYKIN, S. (2001): **Neural Networks – A Comprehensive Foundation**. Prentice-Hall, Delhi.
[10] GADEA, R., J. CERDA, F. BALLESTER and A. MOCHOLI. (2000): Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation. **Proceedings of the 13th International Symposium on System Synthesis (ISSS'00)**, Spain, 225-230.
[11] FUNDING UNIVERSE, "XILINX INC". (2010): Disponible en Web: **http://www.fundinguniverse.com/company-histories/Xilinx-Inc-Company-History.html**. Consultado 23-6, 2010.
[12] LÓPEZ VALLEJO, M. L. y J. L. AYALA RODRIGO. (2004): FPGA: Nociones básicas e implementación. Universidad Politécnica de Madrid, Laboratorio de Microelectrónica. Disponible en Web: **http://www.lsi.die.upm.es/~marisa/docencia/fpga_a2_2004.pdf.** Consultado 10-6, 2010.
[13] ZHU, J. H. y P. SUTTON (2003): FPGA implementations of neural networks – a survey of a decade of progress. **13th International Conference on Field Programmable Logic and Applications**, Lisbon.
[14] IZEBOUDJEN N., A. FARAH, S. TITRI and H. BOUMERIDJA (1999): Digital Implementation of Artificial Neural Networks. From VHDL Description to FPGA Implementation. En: Mira J. and Sánchez-Andrés J. V.,

Engineering Applications of Bio-Inspired Artificial Neural Networks . 139–148, Springer Berlin Heidelberg, Alicante.
[15] MOUSSA M., S. AREIBI and K. NICHOLS (2006): On the arithmetic precision for implementing back-propagation networks on FPGA: A case study. En: Omondi A. R., Rajapakse J.C., FPGA **Implementations of Neural Networks**, 37-61, Springer-Verlag New York, Inc. Secaucus, NJ.
[16] CHIPSCOPE PRO 12.3 (2010): **Software and Cores User Guide**. Disponible en Web: www.xilinx.com. Consultado 15-10, 2010