IMPLEMENTACIÓN DE REDES NEURONALES ARTIFICIALES EN HARDWARE PARA APLICACIÓN EN DETECCIÓN AUTOMÁTICA DE FULGURACIONES SOLARES.

F. Tanco, C. Verrastro, D. Grinberg, J. Roitman.

Universidad Tecnológica Nacional, Facultad Regional Buenos Aires,
Grupo de Investigación de Inteligencia Artificial (GIA),
Medrano 951 (1179) Buenos Aires, República Argentina.
E-mail: fer_tanco@yahoo.com.ar.
dannygrinberg@yahoo.com.ar

Resumen.

El presente trabajo describe y compara dos modelos de neuronas artificiales para implementar redes neuronales mediante circuitos digitales (hardware) aplicadas a la detección automática de fulguraciones solares. Dicha implementación se materializa sobre dispositivos de lógica programable o FPGA (Field Programmable Gate Arrays) las cuales tienen la capacidad de ser reprogramables permitiéndole flexibilidad al diseño y la posibilidad de crear varios tipos de topología de redes neuronales.
El primer modelo está basado en la multiplicación paralela de números binarios y usa funciones de activación genéricas, mientras que el segundo modelo está basado en la codificación serial de los datos mediante flujos de datos (bit-streams) y restringe las funciones activación a escalones.

Abstract.

The present paper describes and compares two models of artificial neurones used in a hardwired neural networks applied to automatic solar flare detection. Such implementation is done over programmable logic devices or FPGA (Field Programmable Gate Arrays) which have reconfiguration capability adding flexibility to the design. The first model is bases in parallel multiplication of binary numbers and uses generic activation functions, while the second model is based in data serial codification with bit streams and restricts the activation functions to binary thresholds.

Introducción a las RNA.

Basados en la eficiencia de los procesos llevados a cabo por el cerebro e inspirados en su funcionamiento, varios investigadores han desarrollado desde hace más de 30 años la teoría de las Redes Neuronales Artificiales (RNA). Las RNA emulan las redes de neuronas biológicas y se han utilizado para aprender estrategias de solución basadas en ejemplos de comportamiento típico; estos sistemas no requieren que la tarea a ejecutar se programe, sino que generalizan y aprenden de la experiencia. Este proceso de aprendizaje puede ser supervisado o no supervisado usando un conjunto de datos de entrenamiento (patrones), Gonzalez Hilera (1995).

La teoría de las RNA ha brindado una alternativa a la computación clásica, para aquellos problemas, en los cuales los métodos tradicionales no han entregado resultados muy convincentes. Las aplicaciones más exitosas de las RNA son: Looney (1997)

- Procesamiento de imágenes y de voz
- Reconocimiento de patrones

- Planeamiento
- Interfaces adaptativas para sistemas hombre/máquina
- Predicción
- Control y optimización
- Filtrado de señales

Los sistemas de cómputo tradicional procesan la información en forma secuencial. Una computadora consiste por lo general de un solo procesador que puede manipular instrucciones y datos que se localizan en la memoria. El procesador lee, y ejecuta una a una las instrucciones en la memoria; este sistema es secuencial, todo sucede en una sola secuencia determinística de operaciones. Las RNA no ejecutan instrucciones, responden en paralelo a las entradas que se les presenta. El resultado no se almacena en una posición de memoria, es el estado de la red para el cual se logra equilibrio. El conocimiento de una red neuronal no se almacena en instrucciones, el poder de la red está en su topología y en los valores de las conexiones (pesos) entre neuronas.

Las ventajas de las redes neuronales son:

- *Aprendizaje adaptativo*. Capacidad de aprender a realizar tareas basadas en un entrenamiento o una experiencia inicial.
- *Auto organización*. Una red neuronal puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.
- *Generalización*. Facultad de las redes neuronales de responder apropiadamente cuando se les presentan datos o situaciones a los que no habían sido expuestas anteriormente.
- *Tolerancia a fallos*. La destrucción parcial de una red conduce a una degradación de su estructura; sin embargo, algunas capacidades de la red se pueden retener, incluso sufriendo gran daño. Con respecto a los datos, las redes neuronales pueden aprender a reconocer patrones con ruido, distorsionados o incompletos.
- *Operación en tiempo real*. Los cómputos neuronales pueden ser realizados en paralelo, y se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.
- *Fácil inserción dentro de la tecnología existente*. Se pueden realizar chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello facilita la integración modular en los sistemas existentes.

Existen varias formas de nombrar una neurona artificial, es conocida como nodo, neuronodo, celda, unidad o elemento de procesamiento (PE, proviniendo de las siglas en inglés). En la Figura 1 se observa un PE en forma general y su similitud con una neurona biológica.

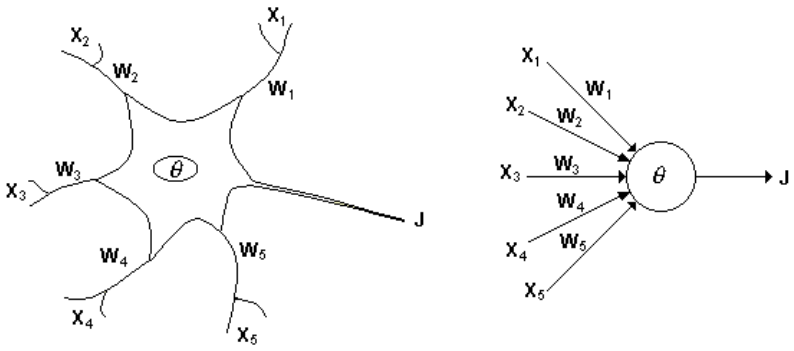

Figura 1. Neurona biológica y neurona artificial.

De la observación detallada del proceso biológico se han concebido los siguientes análogos en el sistema artificial:

- Las entradas X_j representan las señales que provienen de otras neuronas y que son capturadas por las dendritas.
- Los pesos W_j son la intensidad de la sinapsis que conecta dos neuronas, tanto X_i como W_i son valores reales.
- θ es la función umbral que la neurona debe sobrepasar para activarse, este proceso ocurre biológicamente en el cuerpo de la célula.

En este esquema las señales de entrada a una neurona artificial $X_1, X_2,.., X_n$ son variables continuas en lugar de pulsos discretos, como se presentan en una neurona biológica. Cada señal de entrada pasa a través de una ganancia o peso, llamado peso sináptico o fortaleza de la conexión cuya función es análoga a la de la función sináptica de la neurona biológica. Los pesos pueden ser positivos (excitatorios), o negativos (inhibitorios), el nodo sumatorio acumula todas las señales de entradas ponderadas (multiplicadas por los pesos) y las pasa a la salida a través de una función umbral o función de transferencia. La entrada neta a cada unidad puede escribirse de la siguiente manera: Tanco (2003)

$$neta_i = \sum_{j=1}^{n} W_j X_j = \vec{X}\vec{W}$$

Una vez que se ha calculado la activación del nodo, el valor de salida equivale a

$$J_i = \theta_i(neta_i)$$

Donde θ_i representa la función de activación para esa unidad, que corresponde a la función escogida para transformar la entrada $neta_i$ en el valor de salida J_i y que depende de las características específicas de cada red.

La agrupación de varias neuronas conectadas entre sí forman una red neuronal. Existen varias arquitecturas o topologías de interconexión, cada una de ellas con un algoritmo de aprendizaje propio. Las redes FANN (feedforward artificial neural networks) (Figura 2): tienen una capa de entrada de nodos, que no son neuronas artificiales y sólo distribuyen cada entrada en cada neurona de la segunda capa (nodos de bifurcación); una segunda capa de neuronas (capa oculta o intermedia) y una tercera capa de neuronas de salida (capa de salida).

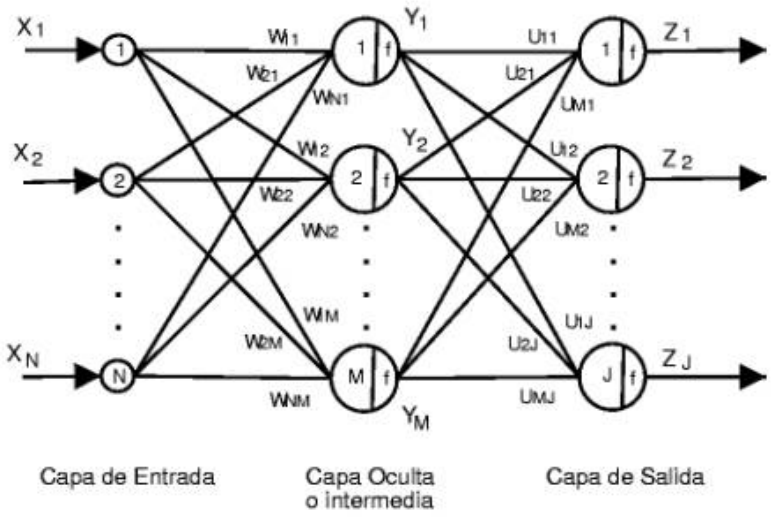

Figura 2. Esquema de la arquitectura de una red neuronal tipo FANN.

Esta arquitectura de dos capas de neuronas es capaz de realizar clasificaciones no lineales tan precisas como se desee. Las redes neuronales han demostrado ser un excelente instrumento para el reconocimiento y clasificación de patrones en imágenes. No obstante, es importante mencionar que gran parte de su habilidad se basa en una buena determinación (a priori) de los parámetros relevantes que identifican el patrón. Un conjunto de parámetros (vector de entrada) mal definido o incompleto imposibilita a la red realizar su tarea en forma correcta. Por otra parte, un conjunto de parámetros sobredimensionado incrementa el número de neuronodos para la clasificación y, consecuentemente, los tiempos de procesamiento, por lo tanto gran parte de la eficiencia de una red está en la correcta definición del problema, en los métodos de entrenamiento y en el conjunto de entrenamiento usado.

Introducción al problema de reconocimiento de imágenes solares.

En Fernández Borda et.al.. (2002) se describe un software, basado en RNA, para la detección automática de las fulguraciones solares observadas con el telescopio HASTA (H_α Solar Telescope for Argentina) localizado en El Leoncito, San Juan. Resolver el problema del reconocimiento de las fulguraciones implica resolver el problema de reconocer ciertos patrones en las imágenes del HASTA, tarea que un observador humano puede realizar sin mucha dificultad. No obstante, los procesos de clasificación realizados por el hombre suelen ser muy complejos y dependen fuertemente de la apreciación personal. La situación se hace más compleja aún, si consideramos que para que el instrumento sea capaz de realizar el reconocimiento, debemos ser capaces de generar un algoritmo que englobe exactamente toda la serie pasos que sigue el cerebro del observador para efectuar la tarea. Los casos, como este, en los que existe una gran cantidad de datos a clasificar y existe una suficiente cantidad de patrones son dónde las RNA poseen una amplia ventaja frente a los algoritmos tradicionales de cómputo secuencial.

Mediante una RNA relativamente sencilla y un conjunto de datos patrones (seleccionados por el experto humano) solo se requiere entrenar la red hasta que la misma reconozca adecuadamente (con un porcentaje de fallos menor a lo requerido). Esto redunda en una optimización del tiempo de desarrollo y de los recursos humanos involucrados y del costo. Todo esto sin detrimento de los resultados que se obtendrían por el método de desarrollo tradicional.

El análisis de imágenes con alta resolución espacial y temporal consume una cantidad de memoria enorme y mucho tiempo de CPU. Por lo tanto, es esencial encontrar un conjunto mínimo de parámetros que contenga toda la información relevante sobre el patrón buscado. Además, otra limitación importante es

que estos parámetros deben ser rápida y fácilmente extraíbles de las imágenes sin procesar, ya a que el objetivo final es que la red identifique fulguraciones en tiempo real.

El modelo de la red neuronal usado para el reconocimiento automático de fulguraciones solares implementado en software se observa en la Figura 3.

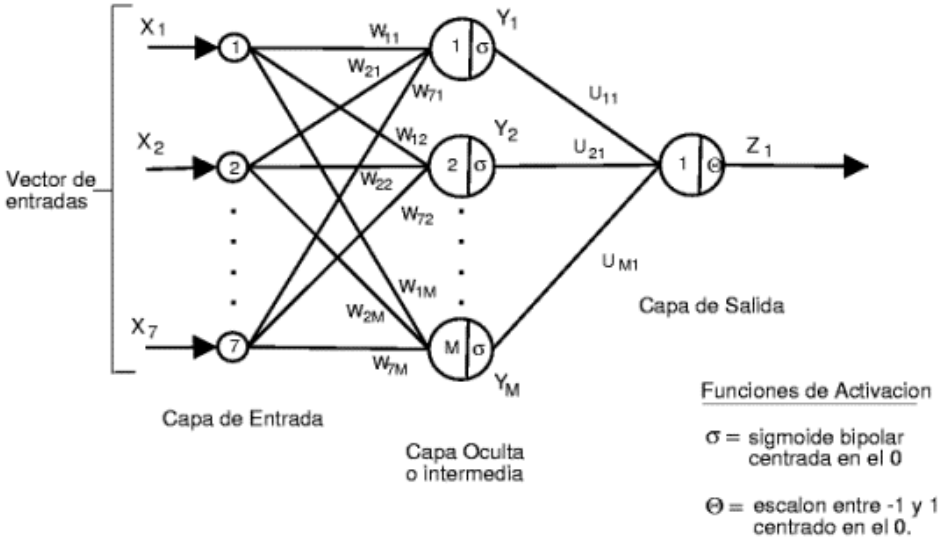

Figura 3. Esquema de la arquitectura usada en la red neuronal implementada.

La red posee una capa de entrada y dos capas de neuronas, la capa oculta y la de salida. La capa de entrada está compuesta por 7 nodos de distribución dados por el número de parámetros que son suficientes para caracterizar al patrón a clasificar. La capa oculta está formada por 11 neuronas, todas ellas con funciones activación sigmoidales y la capa de salida posee una sola neurona con una función activación del tipo escalón para clasificar dos clases (fulguración y no-fulguración).

Los parámetros seleccionados como vector de entradas a la red son: (1) intensidad media de la imagen, (2) desviación estándar de la intensidad, (3) diferencia de intensidad máxima entre dos imágenes consecutivas, (4) valor absoluto de la intensidad en el pixel donde se registró la diferencia de intensidad máxima, (5) posición radial de ese mismo pixel, (6) variación del valor de intensidad media entre dos imágenes consecutivas y (7) contraste entre el pixel donde se registró la diferencia de intensidad máxima y sus primeros vecinos. Los primeros cuatro parámetros están orientados a identificar la imagen correspondiente al inicio del evento. El parámetro (5) tiene el objetivo de enseñarle a la red neuronal a tener en cuenta efectos de proyección y oscurecimiento al limbo. El parámetro (6) tiene en cuenta los efectos de las condiciones climáticas en el sitio de observación. El parámetro (7) tiene como objetivo enseñarle a la red neuronal a distinguir las "plages" (zona de brillo intenso en Hα de las regiones activas) de las fulguraciones, contribuyendo también a filtrar el efecto de la turbulencia atmosférica.

Se seleccionó un conjunto de 361 eventos, de los cuales el 67% (237 eventos) fue usado para el entrenamiento y el 33% restante fue usado para probar el resultado del aprendizaje. La red fue entrenada con el algoritmo de propagación hacia atrás (Backpropagation), el umbral de tolerancia para eventos mal clasificados fue fijado en un 3% durante el entrenamiento de la red. Este umbral se alcanzó sólo después de aproximadamente 10^3 iteraciones del algoritmo (Figura 4).

La red fue puesta a prueba usando un conjunto de 124 eventos. Después de realizar la propagación directa de todos los eventos de este conjunto, se comprobó que sólo fueron mal clasificados una fracción menor al 5%.

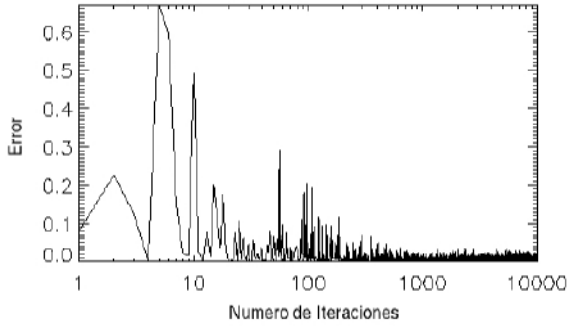

Figura 4. Convergencia del error en función del número de iteraciones.

Implementación de RNA en hardware.

La RNA puede realizarse por software o por hardware. En el primer caso son programas (escritos especificamente para cada aplicación o basados en bibliotecas de programas comerciales) que implementan algoritmos de RNA y en el segundo caso son circuitos electrónicos digitales.

Uno de los mayores inconvenientes para la implementación en hardware de redes neuronales es la cantidad de lógica necesaria para realizar la multiplicación de cada entrada por su correspondiente peso y las subsecuentes adiciones. Como vimos, el modelo matemático de una neurona puede expresarse de la siguiente manera:

$$n_i(x_1,...,x_n) = a_i\left(\sum_{1\le j\le n} w_{ji}\cdot x_j\right)$$

Donde x_j son las señales de entrada, w_{ji} los pesos y a_i la función de activación.

Se optó primero por modelar la neurona a través de una arquitectura de multiplicación en paralelo o tipo "shift and add" Salapura (1994). En la Figura 5 se observa una unidad neurona con 8 entradas, cada una de las cuales acepta valores numéricos de 8 bits.

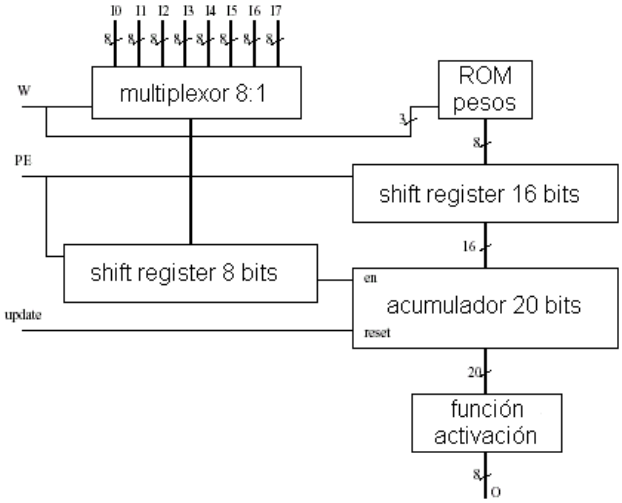

Figura 5. Modelo de neurona con arquitectura paralela.

Cada unidad realiza la multiplicación de cada peso (8 bits signed) por su correspondiente entrada (8 bits unsigned) formando un producto de 16 bits signado. Los ocho productos son acumulados en un resultado de 20 bits (es la suma de productos). El resultado final se obtiene aplicándole una función de activación arbitraria a la salida del acumulador, este proceso escala el resultado intermedio a un valor de salida de 8 bits. El multiplexor selecciona cada una de las entradas de la neurona y la envía a un shift register de ocho bits con salida serie del bit menos significativo (shift right), el cual controla la habilitación del acumulador. Los pesos están almacenados en una look-up table de 8 bytes, uno para cada entrada y pasan a un shift register de 16 bits (salida paralela) que se encarga tanto de completar el byte más significativo con el bit de signo como de desplazar los valores hacia la izquierda. El acumulador se encarga de ir haciendo solo las sumas de los pesos desplazados según sea la señal de habilitación, la cual estará controlada por los '1' provenientes del valor de entrada.

Después de ocho ciclos de reloj se procesa la primera multiplicación, y después de 64 ciclos se procesa la sumatoria de los ocho productos. Luego se aplica a este resultado la función de activación almacenada en una look-up table de 256 bytes. Este último proceso escala el resultado intermedio de 20 bits a un valor de salida de 8 bits unsigned, el cual ya está listo para ser introducido a la entrada de otra neurona.

Este modelo fue sintetizado en un dispositivo de lógica programable o FPGA tipo 4010XL de Xilinx Inc. Esta FPGA posee 10.000 compuertas lógicas, ver XILINX CORP (1992), agrupadas en una matriz de 20x20 CLBs (Configurable Logic Blocks), los cuales proveen los elementos funcionales para la construcción de la lógica del usuario. Los resultados de la síntesis fueron:

Lógica base sin considerar las look-up tables: 51 CLBs.
Look-up tables de función activación y pesos: 15 CLBs (264 bytes).
Consumo total de una neurona: 66 CLBs.

La FPGA posee 400 CLBs, por lo tanto en cada chip solo puden implementarse 6 neuronas y teniendo en cuenta que la red neuronal de la aplicación de detección de fulguraciones solares usa 12 neuronas (11 en la capa oculta y 1 en la de salida) es evidente que en un solo chip no puede sintetizarse la red entera.

Las ventajas de este modelo son:

- Las neuronas podrán ser usadas para cualquier modelo de red, desde arquitecturas feed-forward hasta redes recursivas (Hopfield, Kohonen, etc).

- Capacidad de reprogramar la FPGA para generar el hardware correspondiente a cada modelo elegido.
- Debido a que el entrenamiento se realiza por software, no se requiere la implementación en hardware de dicha fase.
- La red neuronal será escalable, permitiendo la replicación masiva de neuronas, solo limitada por la capacidad lógica de la FPGA.
  La principal desventaja es el consumo de lógica en la FPGA requerido para modelar una neurona.

Es común en redes del tipo backpropagation encontrar modelos que tengan más de seis neuronas distribuidas en la capa oculta y de salida. Considerando que el modelo de la neurona se realiza con la arquitectura de tipo "shift and add", para realizar modelos de redes con más de seis neuronas habría que usar más de una FPGA tipo XC4010XL, e conectarlas entre sí para formar la red neuronal completa.

Considerando que la potencia de una red neuronal es función del numero de neuronas, incrementar el número de neuronas en la misma FPGA aumentaría el rendimiento. Para lo cual se optó por una arquitectura de multiplicación en serie o tipo "bit stream", Gschwind (1994), Murray (1988). Tanto los valores de entrada como los pesos son codificados en flujos de datos serie en el que la fracción de bits en "1" dentro de un período se utiliza para representar el valor. El producto se realiza intersectando bit a bit (operación AND lógica) a ambas señales. Este último proceso se denomina "choppeado digital" y se ejemplifica en la Figura 6 mostrando:

a) las señales de choppeo (chopping signals) para un bit stream de longitud 32. Por ejemplo, el peso 0,5 es codificado mediante una señal bit stream que la mitad de sus bits es "1".
b) la generación del peso 0,8 mediante la combinación de las señales anteriores.
c) la multiplicación de un valor de entrada (10) por un peso (0,8) haciendo la intersección de las dos señales.

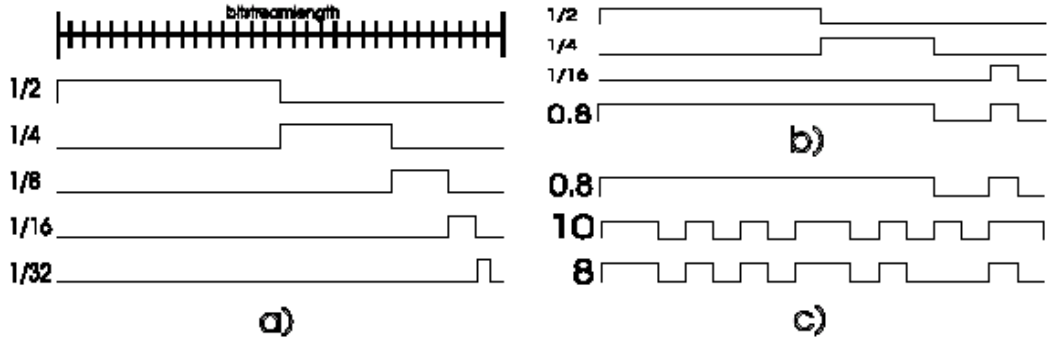

Figura 6. Señales de choppeado y multiplicación.

Para una correcta operación debe haber una gran disparidad de frecuencias entre las señales a multiplicar, siendo la chopping signal (peso) la señal más lenta, y la señal bit stream a ser multiplicada debe tener sus bits en "1" uniformemente distribuidos en el período.

El modelo de la neurona puede verse en la Figura 7. En él las entradas y los pesos se multiplican intersectándolos bit a bit y las señales post sinápticas son acumuladas en el contador de 12 bits, dicha acumulación se realiza por multiplexado de tiempo (multiplexor 8:1) para reducir la complejidad del hardware.

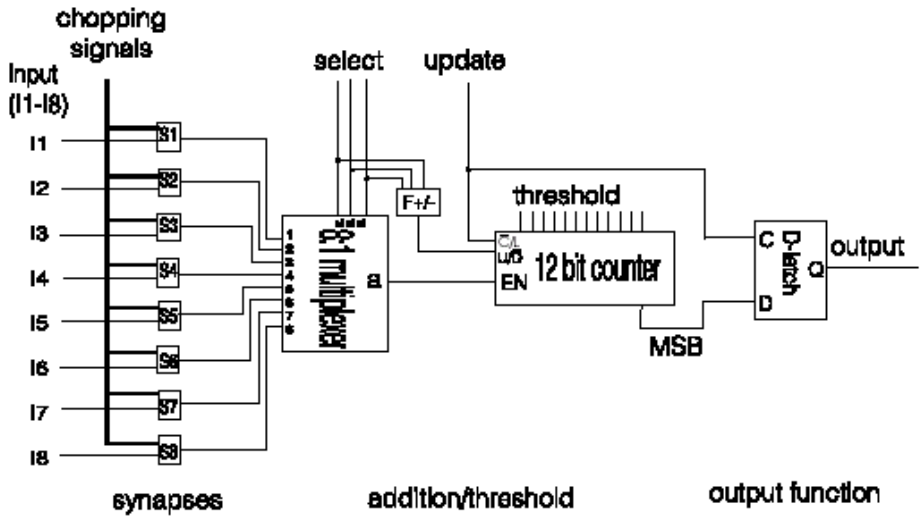

Figura 7. Modelo de neurona "bit stream".

Los valores de entrada son codificados en bit streams el rango [0, 1] al igual que las chopping signals. Una función signo (F+/-) determina si una entrada específica es restada o sumada simulando las sinapsis inhibitorias o excitadoras. El contador es usado también para generar el umbral (threshold) de la función activación, cuando el valor acumulado es mayor al umbral, el bit más significativo del contador se pone a 1 y es almacenado en el latch, listo para ser transferido a las neuronas de la siguiente capa.

Para un bit stream de 256 bits de largo, la acumulación de las señales post-sinápticas requieren 8 ciclos de reloj (un ciclo por cada canal de entrada) y un nuevo ciclo de cómputo empieza cada 2048 ciclos de reloj. Esta condición está controlada por un contador global que distribuye esta señal (update) a todas las neuronas de la red. Cuando la neurona recibe la señal de update, latchea su salida y recarga el contador con el umbral para empezar un nuevo ciclo de cómputo.

Cabe destacar que en este modelo de neurona la función activación es del tipo escalón cuya salida puede ser 1 (neurona activada) o 0 (neurona inhibida). El reemplazo de este tipo de función activación por las funciones sigmoidales que posee la aplicación de reconocimiento de fulguraciones solares hace que el error caracterizado por eventos mal clasificados pase de un 3% a un 6% aproximadamente, lo cual es aceptable.

La precisión de la representación del valor es función lineal del largo del bit stream, la eficiencia de la neurona puede medirse en disparos por segundo o actualizaciones por segundo (Updates per Second UPS), en la Tabla 1 puede verse la eficiencia y la precisión de la neurona para distintas longitudes de bitstreams usando un clock de 33MHz para neuronas de 8 y 4 entradas. La precisión se define como el delta entre dos valores representables consecutivos.

Long. bitstream	Precisión	8 entradas	4 entradas
256	0.0039	16000+	32000+
128	0.0078	32000+	64000+
64	0.0156	64000+	128000+

Tabla 1. Precisión y eficiencia en función de la longitud del bitstream.

El resultado de sintetizar este modelo en una FPGA 4010XL (400 CLBs) arrojó que cada neurona consume 25 CLBs aproximadamente, con lo cual en dicha FPGA podría formarse una red neuronal de hasta 15 neuronas, suficiente como para implementar la aplicación de detección de fulguraciones solares (RNA de 12 neuronas).

Conclusiones.

Usando la codificación de los valores de entrada y de los pesos de las neuronas mediante bit-streams y realizando la multiplicación aplicando un choppeo digital se logra reducir considerablemente el consumo de lógica con respecto al modelo de multiplicación paralela. Considerando que la potencia cálculo de una red neuronal es función del número de neuronas que la componen, esta arquitectura incrementa el rendimiento al permitir un mayor número de neuronas en un mismo circuito integrado FPGA.

La precisión queda determinada por el largo del bit-stream y el hecho de substituir las funciones de activación sigmoidales por escalones no modifica en forma significativa el error general de la red, en nuestro caso particular de detección automática de fulguraciones solares el error de datos mal clasificados pasa de un 3% a un 6%, lo cual es aceptable.

Las FPGA es usada como plataforma para implementar en hardware la red neuronal usando un esquema de entrenamiento off-line (mediante software), esto facilita la implementación debido a la capacidad de reprogramación de las FPGAs.

La simplicidad de los modelos de neuronas en hardware permiten replicarlas masivamente e interconectarlas entre sí para construir redes neuronales complejas generando así una herramienta universal para poder implementar diferentes topologías de redes neuronales, limitadas solamente por la densidad de lógica de las FPGAs.

Al ser las FPGAs circuitos digitales de muy alta velocidad (150MHz típ.), las redes neuronales implementadas en esta tecnologías procesan la información mucho más rápido que las implementadas en software. Por ejemplo, una red neuronal implementada en una FPGA con un clock de 100MHz tardaría en propagar la señal unos 20μs, mientras que un software corrido en una PC tardaría varios ms con un clock de la misma frecuencia.

Referencias.

FERNANDEZ BORDA, R.; MININNI, P.; MANDRINI, C.; (2002) Automatic Solar Flare Detection Using Neural Network Techniques, proc. IAFE, Buenos Aires, Argentina.
GSCHWIND, M.; SALAPURA, V.; MAISCHBERGER, O. (1994) RAN2SOM: A Reconfigurable Neural Network Architecture Based on Bit Stream Arithmetic, Proceedings of the IEEE on Circuits and Systems, London, UK.
GONZALEZ HILERA, J.; HERNANDO MARTINEZ, V.; (1995) Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones, RA-MA.
LOONEY, C.; (1997) Pattern Recognition Using Neural Networks, Oxford University Press.
MURRAY, A.; SMITH, A.; (1988) Asynchronous VLSI Neural Networks Using Pulse-Stream Arithmetic, IEEE Journal of Solid-State Circuits 23(3), 688-697.
SALAPURA, V.; (1994) Neural Networks Using Bit Stream Arithmetic: a Space Efficient Implementation, Proceedings of the IEEE on Circuits and Systems, London, UK.
SALAPURA, V.; MAISCHBERGER, O.; GSCHWIND, M.; (1994) A Fast FPGA Implementation of a General Purpose Neuron, Fourth International Workshop on Field Programmable Logic, Prag, Czech Republic.
TANCO, F.; (2003) Introducción a las Redes Neuronales Artificiales, Grupo de Inteligencia Artificial (GIA), UTN-FRBA, Argentina.

XILINX CORP; (1992) The XC4000 Databook.