Mapas Auto-Organizativos dentro de un FPGA

(Redes neuronales)

Matias Namiot
Facultad de Ingeniería de la UNLP
Centro de Técnicas Analógico-Digitales (CeTAD)
La Plata, Argentina
matiasnamiot@gmail.com

Leonardo Capossio
Facultad de Ingeniería de la UNLP
Centro de Técnicas Analógico-Digitales (CeTAD)
La Plata, Argentina
capossio.leonardo@gmail.com

José A. Rapallini
Facultad de Ingeniería de la UNLP
Centro de Técnicas Analógico Digitales (CeTAD)
La Plata, Argentina
josrap@gmail.com

Antonio A. Quijano
Facultad de Ingeniería de la UNLP
Centro de Técnicas Analógico Digitales (CeTAD)
La Plata, Argentina
adrian.quijano@gmail.com

***Abstracto*—El objetivo de este trabajo es la implementación de un algoritmo de Mapas Auto-Organizativos (SOM, en inglés Self-Organizing Map) aplicable a un dispositivo programable del tipo FPGA (Field Programmable Gate Array) para procesar datos de diferentes problemas.**

Palabras clave: Redes neuronales, FPGA, SOM, VHDL, Self-Organizing Map

I. INTRODUCCION

Con el avance de la tecnología, frente a los dispositivos semiconductores, la computación, las velocidades de funcionamiento y las unidades de almacenamiento, el ser humano ha intentado acortar aún más la brecha que existe con la naturaleza. Las señales no determinísticas que se presentan en ella como el reconocimiento de patrones, son un claro ejemplo de la necesidad de generar sistemas que tengan la posibilidad de resolver problemas complejos bajo entornos imprecisos o con ruidos.

Las redes neuronales generaron un acercamiento hacia la resolución de procesos con dichas características, y en dónde se encuentran los denominados Mapas Auto-Organizativos (SOM), reconocidos por su capacidad para aprender la organización de los datos de entrada de manera no supervisada, es decir, sin información de la salida esperada. El resultado del entrenamiento de una red SOM es una estructura que respeta la topología de los datos reflejando las similitudes existentes entre los patrones de entrada.

Los mapas de Kohonen son un tipo de redes neuronales que poseen la capacidad de aprendizaje no supervisado, y de descubrir la estructura que subyace en los datos, por medio de la modificación de sus pesos.

Estas redes son sencillas de modelar y poseen un gran potencial en la práctica, ya que han demostrado un muy buen desempeño en problemas de clasificación de patrones, extracción de rasgos, cuantificación vectorial, monitoreo de procesos, reducción de dimensiones, minería de datos, análisis exploratorios, etc. Algunos ejemplos claros son aplicaciones en reconocimientos del habla, clasificación de proteínas, reconocimiento de patrones financieros, etc.

En este trabajo se desea implementar para un dispositivo FPGA un algoritmo de Mapas Auto-Organizativos (SOM) de Kohonen con el fin de ser utilizado con diferentes tipos de datos. El algoritmo producirá una caracterización de los datos tal que permita tomar decisiones sobre nuevos datos ingresados.

II. MAPAS AUTO-ORGANIZATIVOS

Los mapas auto-organizativos (SOM) fueron desarrollados a lo largo de la década de los ‘80 por el físico finlandés Teuvo Kohonen [1] [6], como una continuación natural de la línea de desarrollo de las redes competitivas.

Estas redes tuvieron una buena eficacia en problemas reales como la clasificación de patrones, cuantificaciones vectoriales, extracción de rasgos, minería de datos, etc.

La idea básica de un SOM, es la posibilidad de realizar su entrenamiento sin presentar las salidas a las que se debe asociar cada patrón de entrada, ya que la red a través de un proceso de auto-organización, proporciona un resultado que es el reflejo de las relaciones de similitud existentes de dichas entradas.

El proceso consta de una transformación de un espacio multi-dimensional en una serie de neuronas (figura 1), de tal forma que las similitudes relativas entre los puntos del espacio de entrada se conserven. Cada neurona del SOM tiene asociado un vector de pesos con las mismas dimensiones que el espacio de entrada. Cada neurona de la red está relacionada con las neuronas de su entorno mediante una relación de vecindad, dependiendo de la estructura de la red (rectangular o hexagonal).

La topología de la red, definida por la cantidad de neuronas y su forma de conexión, se establece de antemano.

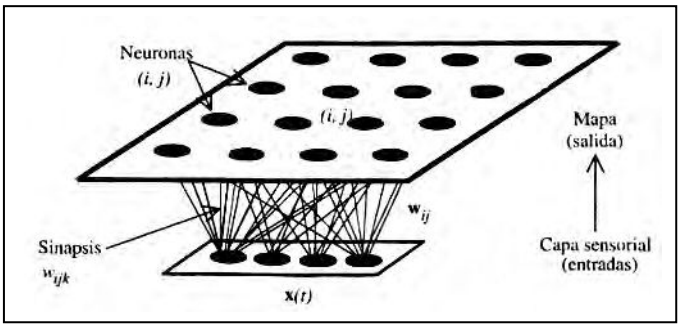

Figura 1. Mapa neuronal

Al excitarse una neurona central, genera la excitación del conjunto de neuronas próximas, y a medida que la distancia entre ellas aumenta, la excitación cae hasta llegar a ser casi nula, ya que el peso que se le da a las conexiones varía inversamente proporcional a la distancia entre neuronas. Esta situación se denomina inhibición lateral [8].

III. ENTRENAMIENTO

Para el entrenamiento del SOM se introducen los vectores de datos correspondientes a cada patrón de entrada, calculando la similitud que existe entre dicho vector y el peso de cada neurona, sintonizando los diferentes rasgos del espacio de entradas y seleccionando a la neurona que más se asemeje a dicho vector. Una vez seleccionada, esta se consolida como ganadora (BMU, Best Matching Unit) y se procede a la actualización de sus pesos sinópticos junto a sus vecinas. De este modo, al reiterarse un patrón similar la vencedora aumentará su intensidad en cada iteración.

Si todo el proceso se repite para numerosos patrones, se obtienen vectores de referencia que sintonizan un dominio específico para las variables de entrada, con una tendencia a representar una función de densidad de probabilidad [Kohonen, 1988]. En el caso de tener un espacio dividido en grupos, se podría realizar un procesamiento dónde las neuronas se especializan en cada uno de los grupos, para luego realizar una clasificación (clustering) de los patrones.

Generalmente, las neuronas de un mismo grupo comparten propiedades comunes que permiten una descripción sintética de un conjunto de datos multidimensionalmente complejos, de allí el uso de las SOM en minería de datos.

Un concepto muy importante que incorpora Kohonen en sus modelos de mapas auto-organizativos es la llamada función de vecindad, que genera relaciones entre la neurona ganadora y sus vecinas.

Hasta ese entonces, se realizaban actualizaciones de los pesos en las neuronas ganadoras lo que respondía a un esquema competitivo clásico y sencillo, pero que no tenía la versatilidad que incorpora la vecindad.

Esta función define un entorno centrado en la vencedora, que produce un efecto durante el aprendizaje sobre sus neuronas vecinas, haciendo que las mismas actualicen sus pesos, aportando respecto al modelo competitivo clásico la ventaja de una mejor convergencia y una mayor robustez ante variaciones en los valores iniciales de sus pesos.

Una vez comenzado el entrenamiento, la medida de la vecindad se va reduciendo a medida que transcurren las iteraciones, hasta que finalmente se modifican solamente los pesos de la ganadora, por lo que el aprendizaje del mapa está comprendido por dos partes importantes, una parte que produce un despliegue del mapa en forma global, y otra que especializa a las neuronas con un ajuste fino.

IV. ALGORITMO DE APRENDIZAJE

Para el diseño de aprendizaje no existe un algoritmo totalmente estándar, aunque los resultados obtenidos son independientes de la realización, pueden llegar a existir algunas variaciones en el mapa final. Esto se debe a que existen parámetros variables en el inicio de un aprendizaje, cómo puede ser el número de vecindad, el valor inicial de los pesos, la velocidad de actualización, etc.

Antes de comenzar con el aprendizaje, se deben establecer las características de la red y sus valores iniciales.

A. Mapa topológico

Es uno de los factores importantes a definir. Este mapa es el encargado de reflejar la estructura de los datos. Existen de diferentes dimensiones y formas, pero por lo general se utilizan mapas bidimensionales rectangulares, de "n*"* neuronas por lado (cuadrado), para facilitar la visualización de los resultados y la actualización de neuronas.

B. Cantidad de neuronas

Idealmente el número de neuronas debe ser de un tamaño considerable para poder extraer más características de los datos. Existe una regla empírica utilizada como referencia [Kohonen, 1996], dada en función del número de patrones con los que se dispone para el entrenamiento como se observa en la ecuación (1), donde "N" es el número de muestras con las que se cuenta.

$$n = 5.N^{0.54321} \qquad (1)$$

Si se distribuye de manera uniforme esta cantidad de neuronas, el mapa cuadrado queda constituido por "n" de ellas por cada lado (ecuación 2).

(2)

C. Pesos iniciales

Estos valores iniciales de la matriz de pesos W juegan un factor importante en el comportamiento del algoritmo, ya que una elección adecuada agiliza la convergencia hacia una buena solución, reduciendo los tiempos de procesamiento. La elección de los mismos en general es empírica.

D. Buscar ganadora

El modelo de Kohonen se basa en el cálculo de la similitud entre el vector de entrada y los pesos correspondientes a cada neurona.

El criterio de distancia que se utiliza en esta implementación para el cálculo de similitud es la distancia de Manhattan (3). Este criterio es utilizado por su sencillez para aplicaciones en las que se requiere optimizar los recursos de hardware. Este método no requiere de multiplicaciones, pero a cambio aumenta el error en la medición. En caso de que no se

pueda llegar a una convergencia de los resultados, se debe utilizar una medida de distancia que posea mayor exactitud y precisión como la Euclídea.

$$d(w_{ij}, x) = \qquad (3)$$

E. Regla de aprendizaje

El efecto de la regla de aprendizaje es acercar de a pequeñas cantidades, el vector de pesos de la neurona ganadora al vector de entrada.

La actualización de los pesos sinápticos se realiza sobre la neurona ganadora y sus vecinas por medio de una función de vecindad h(t) y un parámetro α(t) denominado velocidad de aprendizaje, quedando para la regla Manhattan de la forma (4).

$$\Delta w_{ijk}(t) = \begin{cases} \alpha(t).h(t), & si\ x_k(t) > \\ 0, & si\ x_k(t) = \\ -\alpha(t).h(t), & si\ x_k(t) < \end{cases} \qquad (4)$$

Para este trabajo se han realizado los ensayos con h(t) como una función escalón y α(t) constante.

F. Diagrama en bloques

En la "Figura 2" se puede observar que el algoritmo diseñado inicia los parámetros de funcionamiento previo a la realización de la caracterización. Esto incluye la dimensión del mapa bidimensional, los valores con los que se inicia el vector de pesos, el valor α con que se actualizan los pesos, la vecindad y la cantidad máxima de iteraciones.

Luego se captura un patrón de entrada que busca una neurona ganadora, por medio de la distancia mínima aplicando la regla de Manhattan.

Una vez localizada, se procede a la actualización de la misma y de sus vecinas, dependiendo del valor de vecindad actual, ya que la misma se reduce a medida que el algoritmo avanza con las iteraciones. Este proceso se realiza para cada uno de los patrones una cantidad máxima de veces, seleccionada al inicio de la ejecución.

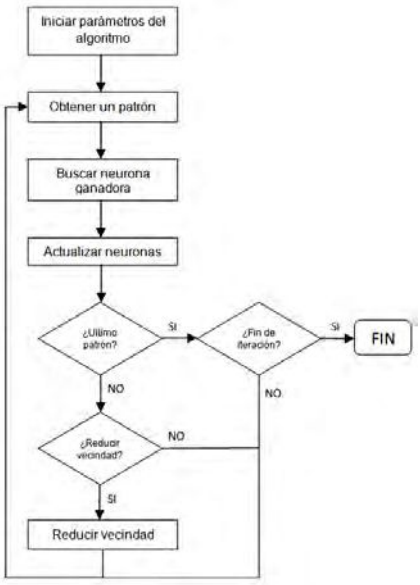

Figura 2. Diagrama en bloques del SOM

V. DISPOSITIVO FPGA

Se desea realizar la implementación en un dispositivo FPGA (Field Programmable Gate Array) debido a la posibilidad de independizar el problema del manejo exclusivo de una computadora. Con el transcurso del tiempo, estos dispositivos han ido adquiriendo importantes características en soporte, estabilidad, tiempos de procesamiento y funcionalidades. Sus usos son cada vez más amplios y las capacidades de almacenamiento lo hacen muy tentador en el momento de realizar un diseño que asemeje funciones de un ordenador.

El término FPGA hace referencia a dispositivos lógicos programables de propósito general, diseñado como un arreglo de bloques y compuertas (ver "Figura 3"), donde su funcionamiento se basa en la posibilidad de configurar la interconexión entre dichos bloques, para que de esta manera se pueda describir el comportamiento de prácticamente cualquier circuito digital.

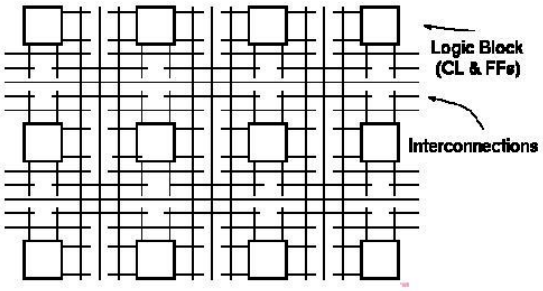

Figura 3. FPGA

Los algoritmos a introducir en el FPGA se implementarán mediante el lenguaje VHDL (Very High Speed Integrated Circuit Hardware Description Language). Este lenguaje fue diseñado en base a los principios de la programación estructurada con la idea de definir la interfaz de un módulo de hardware mientras se dejan invisibles sus detalles internos, y hoy en día se ha convertido en una herramienta imprescindible para diseñadores e ingenieros que se encuentren de alguna manera ligados al desarrollo de sistemas electrónicos digitales.

El dispositivo a utilizar será el FPGA Virtex5 fabricado por Xilinx [2], el cuál posee grandes características como más de 330K de celdas lógicas, y trabaja a velocidades de hasta 550Mhz. La idea de utilizar dicho dispositivo es para poder continuar con el trabajo a futuro incorporando nuevos módulos para conversiones A/D, comunicación con otros dispositivos, etc.

VI. SIMULACIÓN

El diseño, simulación y verificación de todos los bloques se realizó con la herramienta Xilinx System Generator (XSG), generando los bloques que se observan en la "Figura 4".

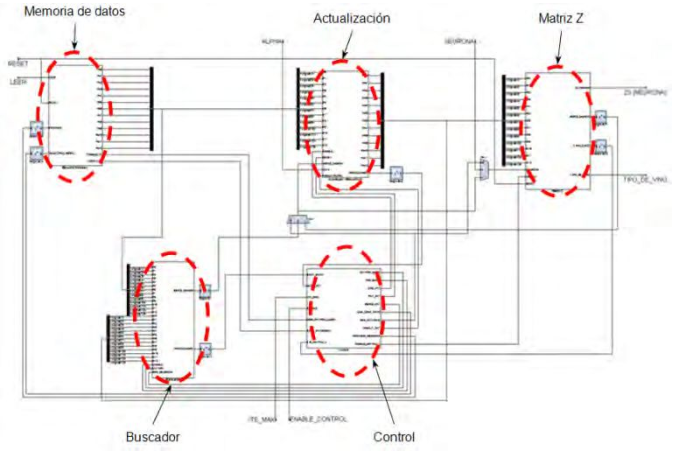

Figura 4. Bloques del XSG

Los resultados obtenidos de la simulación se pueden observar en el mapa neuronal ("Figura 5") que representa a tres clases de tipos de vinos con 13 características similares cada uno. Si se realiza una inspección de los resultados, se puede realizar una clasificación de los mismos.

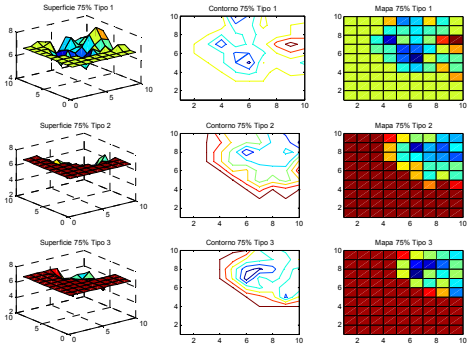

Figura 5. Mapas neuronales de tres clases de vinos diferentes

Los porcentajes aproximados de utilización del dispositivo se pueden observar resumidos en la "Tabla I".

TABLA I. RESUMEN DE UTILIZACIÓN DEL DISPOSITIVO

Slices	**Resumen de utilización para una Virtex5**		
	En uso	***Disponible***	***Utilización***
Slice Registers	106	28800	1%
Used as Flip Flops	106	-	-
Slice LUTs	144	28800	1%
Used as logic	142	28800	1%
Using O6 output only	142	-	-
Used as exclusive route-thru	2	-	-
Route-thrus	2	-	-
Using O6 output only	2	-	-
Occupied Slices	47	7200	1%
LUT Flip Flop pairs	158	-	-
Number with an unused Flip Flop	52	158	32%
Number with an unused LUT	14	158	8%
Number of fully used LUT-FF pairs	92	158	58%
Unique control sets	8	-	-
Slice register sites lost to control set restrictions	10	28800	1%
Bonded IOBs	11	480	2%
BUFG/BUFGCTRLs	1	32	3%
Number of DSP48Es	2	48	4%

VII. CONCLUSIONES

La incorporación de las redes neuronales dentro de un dispositivo mejora notablemente las velocidades de procesamiento, y permite la posibilidad de generar aparatos electrónicos con la habilidad de tomar decisiones en forma independiente de los datos que ingresen al sistema con ruidos o poco precisos. Estas decisiones son aplicables para diferentes problemas; en nuestro caso se utilizó para realizar una clasificación de vinos a través de varias muestras de sus 13 características.

Se pudo observar que la incorporación de técnicas de inteligencia computacional, en este caso SOM, no es imposible dentro de los dispositivos FPGA y se generó el código correspondiente en VHDL para su funcionamiento.

Cómo trabajos a futuro se pretende finalizar con los chequeos sobre el correcto funcionamiento del hardware, ampliar las posibilidades de ingreso de datos mediante algún bloque de comunicación, realizar procesamientos con otras bases de datos y agregar la posibilidad de que el usuario final configure una mayor cantidad de variables.

REFERENCIAS

[1] "Redes Neuronales y Sistemas difusos", Bonifacio Martin del Brio - Alfredo Sanz Molina, 2º edición.

[2] Xilinx – http://www.xilinx.com

[3] http://en.wikipedia.org/wiki/Cepstrum

[4] "Calculadora controlada por voz", Juan Marcelino Aguayo Rodríguez, Prof. Guillermo Kemper Vásquez, Prof. Antonio Moran.

[5] Dan, Z., Zheng, S., Sun, S; Dong, R: Speaker Recognition based on LS-SVM. In: 3rd International Conference on Innovative Computing Information and Control, pp. 25-28 (2008).

[6] Gopalan, K., Anderson, T.R., Cupples, E.J.: A comparison of speaker identification results using features based on cepstrum and Fourier-Bessel expansion. IEEE Transactions on Speech and Audio Processing 7, 289--294 (1999).

[7] Gudnason, J., Brookes, M.: Voice source cepstrum coefficients for speaker identification. IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 4821--4824 (2008).

[8] Han, W., Chan, C.-F., COI, C.-S., Pun, K.-P.: An Efficient MFCC Extraction Method in Speech Recognition. IEEE International Symposium on Circuits and Systems. 4 pp. (2006).

[9] Kohonen, T.: Self-Organizing Maps. 2nd Edition. Springer. ISSN 0720-678X (1997).