Implementación de una Red Neuronal Pulsante parametrizable en FPGA

Ivan Peralta[1] José T. Molas[1] César E. Martínez[1,2] Hugo L. Rufiner[1,2,3*]

[1] *Laboratorio de Cibernética, Facultad de Ingeniería, Universidad Nacional de Entre Ríos*
Ruta 11, Km. 10, Oro Verde, Entre Ríos, Argentina
[2] *Centro de Investigación en Señales, Sistemas e Inteligencia Computacional (SINC(i))*
Dpto. Informática, Facultad de Ingeniería - Universidad Nacional del Litoral
CC217, Ciudad Universitaria, Paraje El Pozo, S3000, Santa Fe, Argentina
[3] *CONICET, Argentina*

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
I. R. Peralta, J. T. Molas Giménez, C. E. Martínez & H. L. Rufiner; "Implementación de una Red Neuronal Pulsante parametrizable en FPGA"
Anales de la XIV Reunión de Procesamiento de la Información y Control, nov, 2011.

***Abstract*— En este trabajo se presenta el diseño e implementación de una red de neuronas pulsantes, con parámetros configurables para utilizar en dispositivos tipo FPGA. Los parámetros configurables incluyen: cantidad de neuronas en cada capa (entrada, oculta y salida), umbral de disparo de cada neurona, valor máximo de la función de respuesta neuronal, periodo refractario, retardos en la transmisión de los pulsos entre cada neurona de cada capa y pesos de las sinapsis. Se detallan el funcionamiento, la estructura interna de cada neurona y sus conexiones con el resto. El funcionamiento de la red de neuronas pulsantes fue simulado e implementado en una placa que contiene una FPGA Spartan3E XC3S250E. Se presentan los resultados obtenidos en ambas etapas como validación del procedimiento.**

***Keywords*— Redes neuronales pulsantes, FPGA, VHDL.**

1. Introducción

Las redes neuronales artificiales (ANN, del inglés *Artificial Neural Networks*) están compuestas por unidades básicas denominadas neuronas. Estas se encuentran interconectadas mediante distintos pesos que determinan la intensidad con que interactuan dichas neuronas. Las ANNs actuales trabajan en base a neuronas que transmiten sólo señales discretas de intensidad variable, todo está temporizado en la misma forma en que está temporizada una computadora digital, en "clicks" de duración invariable. Pero en las células vivas, la dimensión temporal, tanto en la señal de excitación como en la respuesta, es tan importante como la intensidad. Esta forma de trabajar de las ANNs se ha producido por una sobresimplificación de los primeros modelos biológicos que aparentemente omitieron esta dimensión, que ahora aparece como crucial. Como suele pasar, la simplificación es válida sólo en algunos contextos. La búsqueda de una analogía más cercana a la realidad biológica ha dado lugar recientemente a la aparición de las redes neuronales pulsantes (PNN, del inglés *Pulsed Neural Networks*) [7].

Las neuronas biológicas procesan información estructurada en el tiempo, por lo cual el fenómeno de temporización es de crucial importancia en los cálculos realizados en los sistemas neuronales biológicos. Es importante tener en cuenta la frecuencia promedio de los potenciales de acción (PA), pero también se debe considerar la diferencia temporal entre estos potenciales. Las neuronas se comunican entre sí en un lenguaje en el cual el significado impartido a la neurona receptora es codificado en la sincronización de los potenciales de acción. Estos conjuntos de potenciales de acción puede ser tratados como una señal. Una neurona puede disparar(generar un PA) si dos PA en la entrada ocurren muy cercanas (en el tiempo), pero no disparará (o se inhibirá) si las mismas dos PA cambian un poco su "distancia" (diferencia temporal entre los dos PA), sin importar si el intervalo que las separa es más largo o más corto [4]. De esta manera la comunicación, y también el cálculo en las neuronas biológicas difiere completamente de la forma en la cual trabajan hasta ahora las computadoras y las redes neuronales artificiales clásicas.

Si se puede codificar la información en la fase de los potenciales de acción (o pulsos para las neuronas artificiales), también se podría incorporar en los modelos de neuronas la habilidad de cambiar su comportamiento en función de la sincronía de las señales de entrada. Esto llevaría a una regla de aprendizaje que permita cambiar las propiedades temporales de las conexiones de la red, es decir a la idea de sinapsis dinámicas [8, 9]. También ha sido demostrado teóricamente que la codificación temporal es muy eficiente cuando se necesita un rápido procesado de la información [6]. Un aspecto muy importante al utilizar este tipo de redes neuronales, es la factibilidad de implementarlas directamente en un dispositivo FPGA (del inglés *Field Programmable Gate Array*) [2, 10]. Las FPGA son circuitos inte-

* Autor correspondiente: lrufiner@bioingenieria.edu.ar

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
I. R. Peralta, J. T. Molas Giménez, C. E. Martínez & H. L. Rufiner; "Implementación de una Red Neuronal Pulsante parametrizable en FPGA"
Anales de la XIV Reunión de Procesamiento de la Información y Control, nov, 2011.

grados digitales cuyo comportamiento puede ser programado por el usuario. Permiten reemplazar grandes diseños digitales que antes se implementaban con componentes discretos (como compuertas y flip-flops) por un solo integrado. Las FPGAs tienen una capacidad lógica de hasta millones de compuertas, incluyen interfaces programables para varios estándares de interface eléctrica y tienen bloques de funciones especiales embebidos entre la lógica programable, tales como memoria, multiplicadores o CPUs completas. Otra característica importante es su capacidad de procesamiento paralelo. Esto es debido a que el funcionamiento de todas las estructuras lógicas implementadas pueden ser sincronizadas mediante una única señal de reloj. Con esta característica pueden ser utilizadas para cómputo en tiempo real en muchas aplicaciones, incluyendo la que aquí se presenta.

En la Figura 1 se muestra un esquema completo del proceso para entrenar y probar una PNN. Este trabajo no está focalizado en los algoritmos de entrenamiento de la PNN y el entrenamiento se podría realizar antes de programar la FPGA. Para entrenar la red existen distintos algoritmos tradicionales como evaluación del gradiente del error, métodos estadísticos, métodos evolutivos y aprendizaje Hebbiano [5]. Otro aspecto que no incluye este trabajo son los métodos de conversión de señales de entrada en trenes de pulsos.

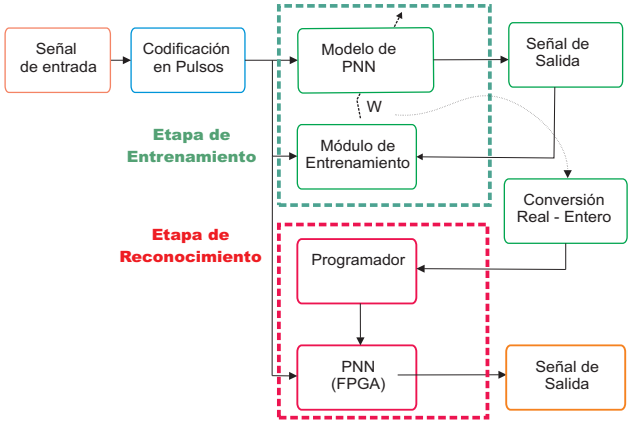

Figura 1: Estructura funcional con otros módulos que se proyectan implementar.

Este trabajo presenta una manera de utilizar las caráterísticas del lenguaje VHDL (VHDL representa la combinación de VHSIC y HDL, donde VHSIC es Very High Speed Integrated Circuit y HDL es Hardware Description Language) para definir distintas características estructurales y variables numéricas de la red neuronal pulsante. Estos parámetros son definidos, previamente a la programación, en un archivo especial de configuración (packaging) y pueden ser fácilmente modificados para reconfigurar la red neuronal pulsante.

La organización del trabajo es la siguiente: en la sección uno se presenta una introducción al trabajo, en la sección dos se explica el modelo propuesto de la red neuronal pulsante, en la tercera parte se detalla el diseño del a estructura y el programa de la red neuronal, detallando cada componente y capa, en la cuarta sección se explica la implementación, simulación y resultados, y en las secciones cinco y seis se detallan las conclusiones, trabajos futuros y agradecimientos.

2. Modelo propuesto de red neuronal pulsante

La estructura de red neuronal pulsante consta de tres capas: capa de entrada, oculta y salida. Cada una de las neuronas de la capa de entrada se conectan con todas las neuronas de la capa oculta y todas las neuronas de la capa oculta se conectan con todas las neuronas de la capa de salida.

En la modelización de una red pulsante existe un compromiso entre la plausibilidad biológica y el costo en tiempo de ejecución del modelo. En la práctica, uno de los modelos más utilizados es el conocido como "Integra-y-dispara" (IFM), cuya descripción se presenta en la Figura 2 [3]. El funcionamiento puede describirse de la siguiente forma. Un pulso generado por la neurona j llega a la sinapsis con la neurona i, produciendo una corriente en el circuito RC, que representa las propiedades eléctricas del soma de i. El circuito RC actúa como un integrador con pérdida. El resultado de la integración actual es el voltaje de respuesta $u(t - t_j^{(f)})$. Si muchos pulsos pre-sinápticos llegan en un intervalo corto, entonces el voltaje total puede sobrepasar el umbral θ, en algún tiempo $t_i^{(f)}$. En ese instante se genera un pulso de salida. Al mismo tiempo, el circuito se pone a tierra para llevar el voltaje a cero nuevamente. Para este modelo, el parámetro para entrenar en cada neurona es la constante RC. Otra posibilidad es reemplazar el comportamiento dinámico de la sinapsis con un filtro IIR o FIR. Los algoritmos de entrenamiento que se podrían utilizar son los tradicionales como el de retropropagación, o alguno de los no supervisados como los basados en la teoría del aprendizaje Hebbiano, pero estos no están implementados en este trabajo.

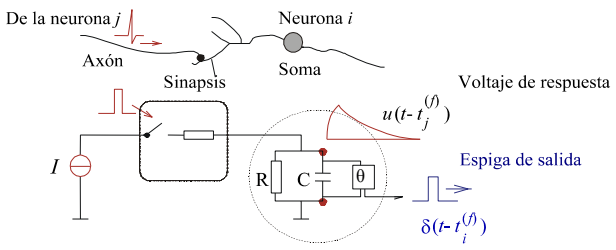

Figura 2: Esquema de funcionamiento del modelo "Integra-y-dispara". Adaptado de [3].

Nuestro modelo está inspirado en el funcionamiento del IFM pero utilizando señales de entrada y salida binarias, además de aritmética entera. A manera de ejemplo, se explica a continuación el funcionamien-

to de una neurona hipotética perteneciente a la capa oculta de una red neuronal de dos neuronas de entrada y dos neuronas de salida. Esta neurona posee dos registros internos que se conectan a una salida de cada una de las neuronas de la capa de entrada, estos registros están inicialmente con valor cero. En el momento en que un pulso llega por una de las entradas, dichos registros internos comienzan a incrementar su valor describiendo una recta hasta llegar a un cierto umbral, la pendiente de dicha recta depende del peso de la conexión entre las dos neuronas involucradas. Una vez llegado a ese umbral comienza a decaer restándole un valor de uno (-1) en cada ciclo de reloj. En otro registro llamado "suma" se va almacenando la suma de los registros internos. En el instante en que la suma sobrepasa un "UMBRAL" determinado se dispara un pulso que entra al primer bit de los FIFOs (del inglés *First Input First Ouput*) de salida, el pulso se dirigirá hacia las neuronas de salida con retardos determinados por la longitud de dichos FIFOs.

3. Diseño de la estructura y programa de la red

La estructura de la red es similar a la utilizada en [1] para demostrar el funcionamiento del algoritgmo "SpikeProp". La cantidad de neuronas de la red se detallan en la Tabla 1. Los parámetros como umbral de suma, periodo refráctario, pesos (parte inferior de la Tabla 1) fueron obtenidos por prueba y error. Este proceso consistió en ajustar los valores para que la PNN genere al menos un pulso en la capa de salida, al ser estimulada con pulsos en la capa de entrada.

La capacidad de configurar la PNN fue implementada a través de un archivo que contiene un "packaging" en lenguaje VHDL. Para evitar problemas de sincronismo, cada componente recibe la misma señal de reloj. La PNN implementada trabaja esclusivamente con valores enteros. Cada tipo de neurona según la capa, fue programada en un archivo VHDL como componentes distintos. Se programaron tres componentes para cada capa: de entrada, oculta y salida, y cada uno contiene una cantidad variable de FIFOs según su cantidad de salidas. También se programó un componente para el FIFO y un componente PNN que contiene a todas las neuronas.

Se aprovechó la ventaja de trabajar con un lenguaje jerárquico para la modelización de los distintos componentes, desde uno sencillo a uno más complejo. Para modelizar los distintos retardos sinápticos entre las neuronas se implementó un componente FIFO a nivel de bits, con una longitud configurable que determina el retardo de la conexión. Se diseñó un componente base que modela una neurona y luego por medio de los parámetros y un comando que permite la replicación de componentes, se configuran las neuronas de cada capa en particular.

3. 1. Componente FIFO

Una manera de simular los retardos de manera eficiente con una FPGA es utilizar estructuras de tipo colas ("FIFO") de registros de bits. El principio se basa en utilizar un bit para modelar un pulso que es introducido al inicio del FIFO y luego de transcurridos cierta cantidad de pulsos de reloj, este bit se ve reflejado en la salida del FIFO. La cantidad de registros del FIFO (largo de la estructura) determina el retardo de la conexión sináptica entre dos neuronas de distintas capas.

3. 2. Neurona de la capa de entrada

Las neuronas de la capa de entrada poseen: una única señal de entrada, una señal de reset, una señal de reloj y un vector de señales de salida de dimensión igual a la cantidad de neuronas de la capa oculta. Cada una de las señales del vector de salida se conectan a una entrada de cada una de las neuronas de la capa oculta.

3. 3. Neurona de la capa oculta

Las neuronas de la capa oculta poseen: un vector de señales de entrada de dimensión igual a la cantidad de neuronas de la capa anterior (entrada), un vector de valores de pesos de igual dimensión que el vector de las señales de entrada, la señal de reset, la señal del reloj y un vector de señales de salida de igual dimensión que la cantidad de neuronas de la capa de salida. Cada una de las señales del vector de salida se conecta a una entrada de las neuronas de la capa de salida.

3. 4. Neurona de la capa de salida

Las neuronas de la capa de salida poseen: un vector de señales de entrada de dimensión igual a la cantidad de neuronas de la capa oculta, un vector de valores de pesos de igual dimensión que el vector de las señales de entrada, una única señal de salida y las señales de reset y reloj.

La Figura 3 muestra esquemáticamente la arquitectura de cada tipo de neurona.

4. Implementación, simulación y resultados

El diseño fue implementado en un kit de desarrollo "Basys Board" de la empresa Digilent que contiene una FPGA Spartan3E XC3S250E con una capacidad equivalente de 250 mil compuertas lógicas. Se asignaron dos pulsadores a las neuronas de la capa de entrada, dos LEDs a las neuronas de la capa de salida, la señal de reset a un pulsador y la señal de reloj de la placa a un divisor de frecuencia para lograr observar los resultados por medio de los LEDs. La Figura 4 muestra una fotografía del kit de desarrollo experimentado. Durante la simulación software se lograron varios pulsos de las neuronas de salida a partir de un generador de pulsos que fue programado como entrada. El software fue impactado en la FPGA y se configuró un pulsador para poder introducir los pulsos de

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
I. R. Peralta, J. T. Molas Giménez, C. E. Martínez & H. L. Rufiner; "Implementación de una Red Neuronal Pulsante parametrizable en FPGA"
Anales de la XIV Reunión de Procesamiento de la Información y Control, nov, 2011.

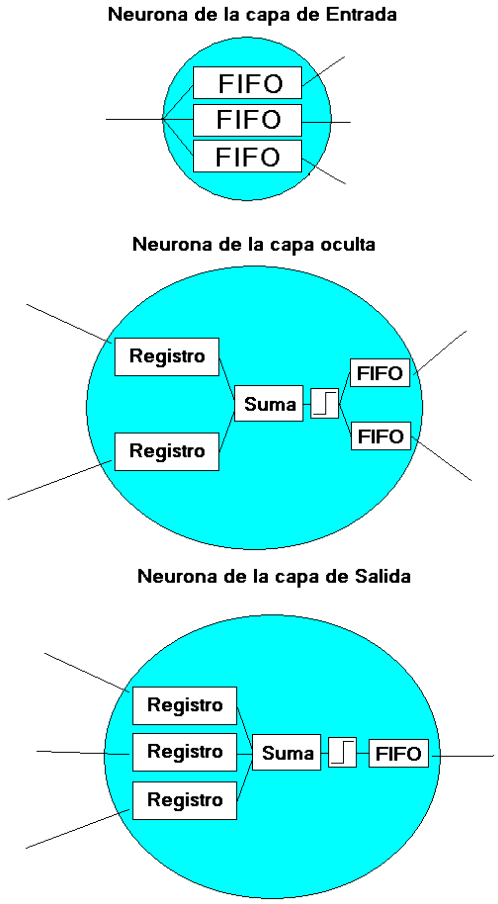

Figura 3: Estructura interna de una neurona de la capa de entrada (arriba), capa oculta (medio) y capa de salida (abajo).

entrada, y se verificaron los mismos resultados que con la simulación.

Para la simulación se utilizó el software ISIM (versión 12.2 (nt)) de la empresa Xilinx Inc. Se configuró en el archivo de estructuras una red neuronal pulsante de dos neuronas de entrada, tres ocultas y dos de salida. La Tabla 1 muestra los valores de los pesos y retardos utilizados en la simulación e implementación. La Figura 5 muestra el comportamiento interno de una neurona de la capa oculta para un umbral de registros de 60 unidades y un umbral de suma de 100 unidades.

Con la configuración utilizada se lograron obtener tres pulsos de salida retardados en el tiempo luego de un estímulo inicial (pulso de entrada) que fue el mismo resultado obtenido por medio de la simulación (Figura 6).

El almacenamiento de datos se realizó con registros de 32 bits. Sin embargo, dado que los valores guardados no necesitan una resolución tan elevada, se podrían utilizar registros de menor tamaño, por ej. 8 bits, a fin de reducir el consumo de recursos del dispositivo.

Un aspecto importante a mencionar es que la velocidad de entrada de datos es la velocidad del reloj, es

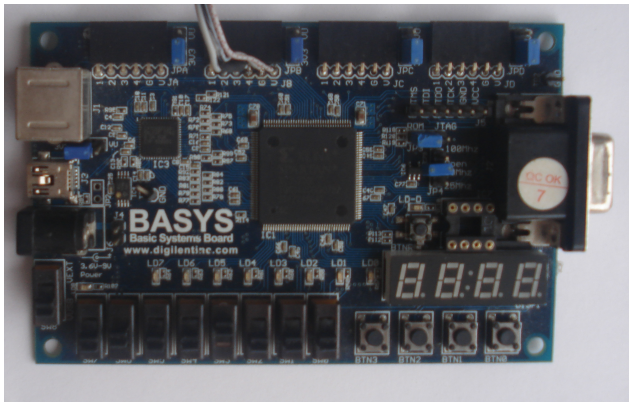

Figura 4: Imagen del kit de desarrollo empleado para implementar la PNN en la FPGA.

Tabla 1: Configuración de la red pulsante.

Parámetro	Valor
Neuronas en la capa de entrada	2
Neuronas en la capa oculta	3
Neuronas en la capa de salida	2
Umbral de Suma	100
Valor Máximo de Registros	60
Período Refractario	10
Longitud FIFOs Capa de Entrada	{(3,6,8);(2,8,4)}
Longitud FIFOs Capa Oculta	{(10,3);(5,6);(7,8)}
Longitud FIFOs Capa de Salida	{(3,5)}
Pesos de Capas Entrada-Oculta	{(2,5,1);(3,8,4)}
Pesos de Capas Oculta-Salida	{(2,2);(1,2);(5,14)}

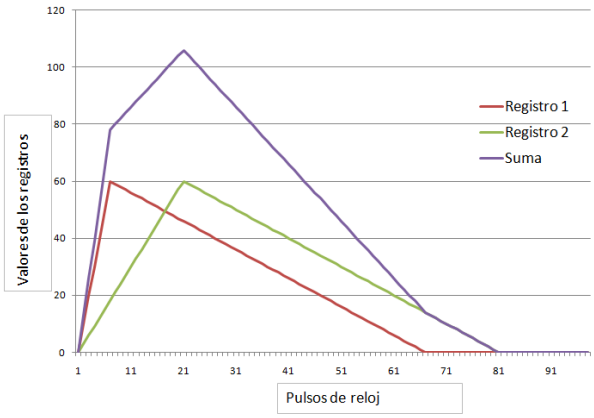

Figura 5: Valores de los registros internos y la suma de ellos de una neurona de la capa oculta, en función del tiempo (pulsos de reloj).

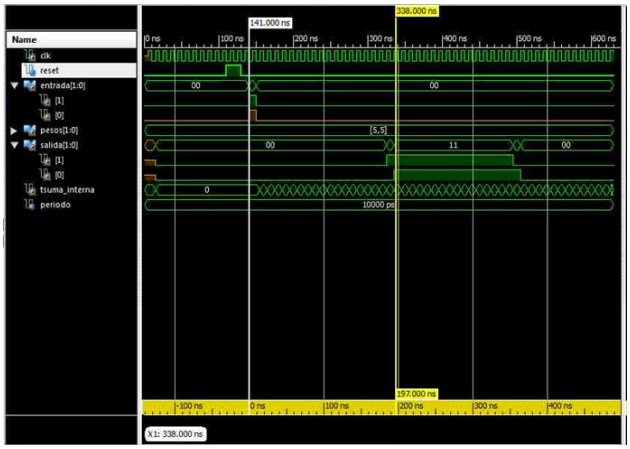

Figura 6: Pantalla del simulador donde se pueden ver distintas señales y valores de registros como: señal de reloj ("clk"), señales de entrada ("entrada[0:1]") , valores de los pesos de las sinapsis ("pesos[0:1]") y señales de salida ("salida[0:1]").

decir por cada señal de reloj puede ingresar un pulso a cualquiera de las neuronas de entrada. El periódo de retardo en la aparición de la salida ante una entrada determinada, depende de los pesos y los retardos (longitudes de los FIFOs) entre las neuronas de cada capa. A medida que los pesos aumentan y las longitudes de los FIFOs son más pequeños, el retardo en la aparición de la salida de la red se reduce.

Finalmente, para estimar el tamaño de la red que se podría implementar con la FPGA Spartan3E y el diseño actual, se aumentó el número de neuronas de la capa media hasta colmar el dispositivo, resultando un máximo de 5 neuronas, ocupando el 96 % de sus slices. De manera similar se procedió con una Virtex5 XC5VFX30T y se concluyó que el mismo diseño ocuparía sólo el 4 % de sus slices.

5. Conclusiones y trabajos futuros

En este trabajo se presentó el diseño e implementación de una red neuronal pulsante mediante una FPGA. En futuros trabajos se diseñarán e implementarán otros módulos, como el de entrenamiento off-line de la PNN a partir de las distintas estrategias de aprendizaje supervisado existentes. Una vez entrenada la PNN podría realizar alguna tarea en tiempo real, por ejemplo, la clasificación de fonemas. Otro aspecto a considerar es la estrategia de codificación adecuada de señales analógicas en términos de pulsos temporales, para permitir señales de entrada arbitrarias en tiempo real, por ejemplo a partir de un generador de señales o bien utilizando señales obtenidas en ambientes reales.

6. Agradecimientos

Agradecemos al Dr. Alfredo Rosado Muñoz por su colaboración y asesoramiento, que hicieron posible la realización del presente trabajo.

Referencias

[1] S.M. Bohte, J.N. Kok, and H. La Poutré. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[2] Pedro Ferreira, Pedro Ribeiro, Ana Antunes, and Fernando Dias. Artificial neural networks processor: A hardware implementation using a FPGA. In Jürgen Becker, Marco Platzner, and Serge Vernalde, editors, *Field Programmable Logic and Application*, volume 3203 of *Lecture Notes in Computer Science*, pages 1084–1086. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30117-2-132.

[3] Wulfram Gerstner, Ritz Raphael, and van Hemmen Leo. Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, 69:503–515, 1993.

[4] Terrence J. Precision of pulse-coupled networks of integrate-and-fire neurons. *Network: Computation in Neural Systems*, 12(2):215–233, 2001.

[5] Andrzej Kasinski and Filip Ponulak. Comparison of supervised learning methods for spike time coding in spiking neural networks. *Int. J. Appl. Math. Comput. Sci*, 16(1):101–113, 2006.

[6] W. Maass. Paradigms for computing with spiking neurons. In J. L. van Hemmen, J. D. Cowan, and E. Domany, editors, *Models of Neural Networks. Early Vision and Attention*, volume 4, chapter 9, pages 373–402. Springer (New York), 2002.

[7] Wolfgang Maass. Computing with spiking neurons. In W. Maass and C. M. Bishop, editors, *Pulsed Neural Networks*, pages 55–85. MIT Press (Cambridge), 1999. Ref. 32.

[8] Wolfgang Maass and Anthony M. Zador. Computing and learning with dynamic synapses. 1999.

[9] Maurizio Mattia and Paolo Del Giudice. Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, 12:2305–2329, 2000.

[10] Horacio Rostro-Gonzalez, Jose Hugo Barron-Zambrano, Cesar Torres-Huitzil, and Bernard Girau. Low-cost hardware implementations for discrete-time spiking neural networks. In *Cinquième conférence plénière française de Neurosciences Computationnelles, Neurocomp'10*, Lyon, France, Aug 2010. paper ID #34 Partially supported by the CorTex-Mex project and the SEP and the CONACYT of Mexico.