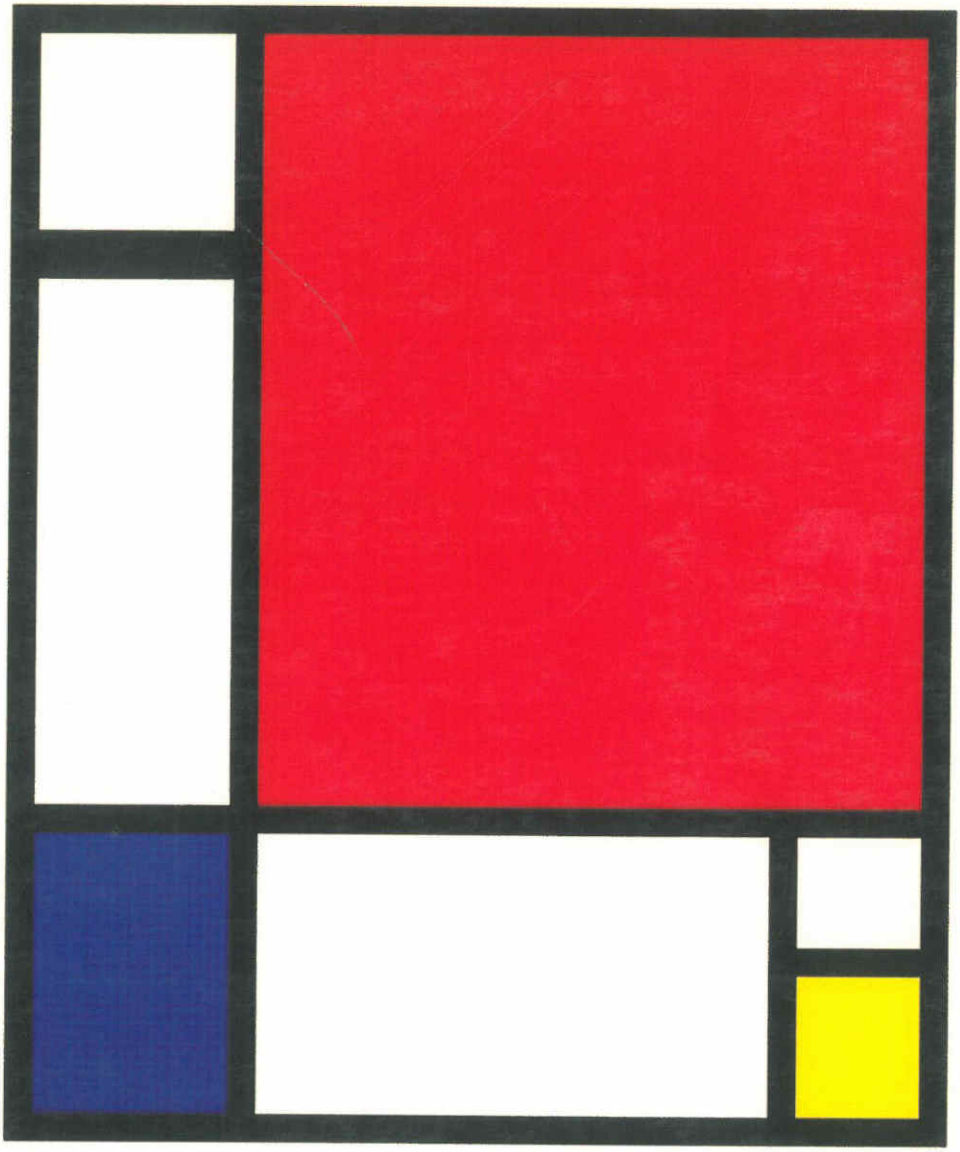


COMPUTACIÓN RECONFIGURABLE & FPGAs

Eduardo Boemo
Francisco Gómez Arribas
Sergio López-Buedo
Gustavo Sutter
(Eds.)

COMPUTACIÓN RECONFIGURABLE & FPGAS

Artículos seleccionados de las III Jornadas de Computación Reconfigurable y Aplicaciones Madrid, 10-12 de Septiembre de 2003

Eduardo Boemo Scalvinoni
Francisco Gómez Arribas
Sergio López-Buedo
Gustavo Sutter Capristo

Editores

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Título: CONFIGURACIÓN RECONFIGURABLE & FPGAs

I.S.B.N.: 84-600-9928-8
Depósito Legal: SE-2914-2003

Editores: E. Boemo, F. Gomez-Arribas, S. López-Buedo, G. Sutter
Escuela Politécnica Superior.
Universidad Autónoma de Madrid.
28049 Cantoblanco
Madrid - España.
www.ii.uam.es

Impresión: Publicaciones Digitales S.A.
C/ San Florencio, 2, 41018, Sevilla - España
Tel: 902 405 500 / (+34) 954 583 425 - Fax: (+34) 954 583 205
info@publidisa.com / www.publidisa.com

Capítulo 6: Redes Neuronales

Capítulo 7: Sistemas de Control

Implementación en FPGAs de una Plataforma de Simulación de Neuronas de Pulsos

Agís R., Ros E., Carrillo R., Ortigosa E. M., Pelayo F.J, Prieto A.

Dept. Arq. y Tec. de Computadores, E.T.S.I. Informática, U. Granada, 18071, España
{ragis, eros, rcarrillo, eva, fpelayo, aprieto} @atc.ugr.es

Resumen. En este artículo se presenta un ejemplo completo de implementanción de un núcleo paralelo de cálculo para simulación de capas de neuronas de pulsos. La implementación se ha desarrollado en plataforma de hardware reconfigurable que permite la evaluación y modificación del sistema in situ. El objetivo principal es valorar el coste de implementación de varias alternativas de diseño propuestas. El diseño presentado ha sido definido mediante un lenguaje de descripción hardware (HDL) de alto nivel (Handel-C). Esto ha hecho posible la extracción de resultados experimentales de forma fácil gracias a la versatilidad de modificación de los dispositivos reconfigurables. Este artículo también describe el esquema de simulación multiplexado para grandes redes de neuronas de pulsos. El objetivo a largo plazo es aplicar este estudio a sistemas de control automatizado como robots y procesos industriales que precisan aprendizaje y coordinación de movimientos.

1 Introducción

Este trabajo ha sido desarrollado en el contexto de SpikeFORCE [1]. SpikeForce es un consorcio formado por investigadores con experiencia en la construcción de sistemas bio-ispirados y su aplicación en el campo de la robótica y automática.

El estudio de redes neuronales bio-inspiradas basadas en pulsos está actualmente en auge por sus aplicaciones inmediatas en campos como: Visión [2, 3], sistemas auditivos [4], coordinación de movimientos [1], etc. Estos estudios resuelven problemas reales tales como la construcción de neuro-prótesis [5], retinas artificiales [6] y sistemas de control para robots [5]. El uso de neuronas de pulsos tiene ventajas e inconvenientes a la hora de su implementación física. En este ámbito destacamos la dificultad de simulación eficiente en arquitecturas convencionales (plataformas monoprocesador) [7]. En estos sistemas la simulación de arquitecturas neuronales con tamaños modestos tienen largos consumos de tiempo y son ineficaces. Con esta idea en mente se han desarrollado en los últimos años diversas plataformas orientadas a la implementación de sistemas de neuronas de pulsos [8, 9] basados en tecnología de FPGAs[10] y ASICs [11, 12, 13]. Además ha aparecido un enorme interés en la simulación de esquemas de procesamiento biológicos en diferentes campos [14, 15, 16]. En este artículo proponemos una implementación hardware de un modelo de red neuronal de pulsos y un estudio del coste de implementación en términos de porcentaje de ocupación de dispositivos reconfigurables y tiempos de computación. En el modelo que proponemos hemos incluido el concepto de HNR (Historia Neuronal Reciente) para emular la inyección-substracción gradual de carga debido a la dependencia del tiempo de la conductancia post-sináptica. En las sinapsis biológicas se inyecta o se resta carga

durante un intervalo de tiempo siguiendo una función temporal modulada por los canales de iones activados por neurotransmisores específicos. Aún no está claro si la inyección gradual de carga biológica no es más que una limitación en la velocidad de modificación del potencial de membrana pero esta modificación gradual podría evitar la saturación de grupos de neuronas y la coordinación de capas neuronales. Actualmente hay varios sistemas que han incorporado esta característica [8,17].

El modelo neuronal que hemos construido, simula *sinapsis con conductancias e inyección gradual de carga* [18]. La implementación de estos dos puntos suponen un conste nada despreciable en hardware que resumimos brevemente a continuación:

- *Sinapsis modeladas como conductancias:* Esto implica una alta resolución en la realización de los productos para el cálculo del potencial de membrana en cada intervalo de tiempo. Pero es útil para simular retardos para sincronización de poblaciones de neuronas y estructuras competitivas.
- *Inyección gradual de carga:* Para modelar este fenómeno biológico es preciso almacenar la historia reciente de las entradas en cada sinapsis. La inyección gradual tiene un gran interés en procesos de sincronización neuronal [19].

2 Descripción neuronal

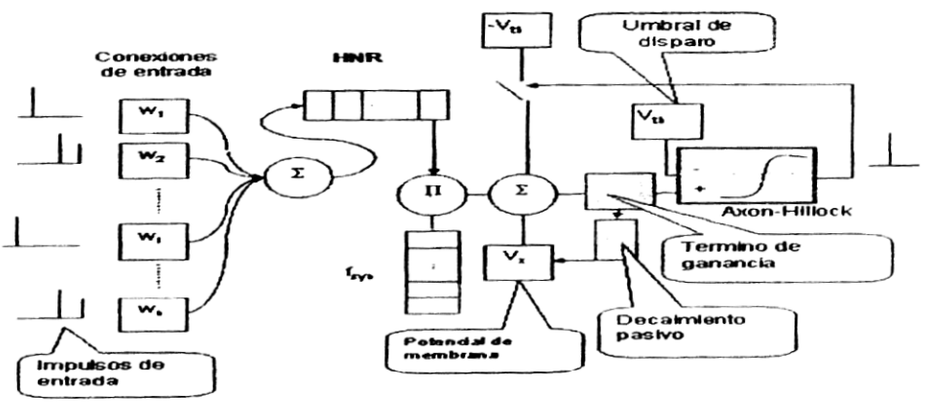


Fig. 1. Esquema Neuronal. La función de dependencia sináptica (f_{syn}) es multiplicada por la Historia Neuronal Reciente (HNR). El resultado se suma como contribución a V_x (potencial de membrana) multiplicado por un término de ganancia. Finalmente el término de descanso es aplicado al valor de V_x en cada intervalo de tiempo.

El cálculo interno de la neurona se realiza siguiendo los siguientes pasos:

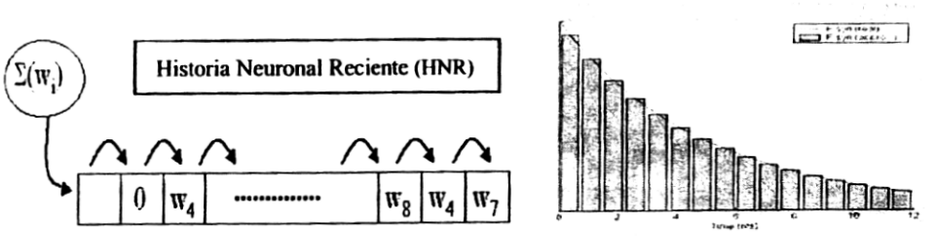
a. Contribución sináptica: Cada ciclo de tiempo una neurona recibe uno o varios pulsos por sus sinapsis. Estos pulsos son multiplicados por los pesos (w_i) siguiendo la exp. (1). El resultado se suma y se añade a la historia reciente (HNR) como una nueva contribución sináptica (vease Fig. 2 (a)). El cálculo de V_x (potencial de membrana) en cada ciclo se realiza siguiendo la ec. (2).

$$C_t = \sum_{i=1}^{L} (I_i * W_i) \quad (1)$$

$$V_x = V_x + (C_t \; C_{t-1} \; C_{t-2} \; \ldots \; C_{t-L}) \cdot \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \ldots \\ f_n \end{pmatrix} (V_{max} - V_x) \quad (2)$$

Donde C_{t-i} es la contribución recibida en el tiempo t-i. T es el periodo de cada intervalo y L es la longitud de la HNR. El termino f_i modula la contribución sináptica. V_{max} es el valor máximo que puede alcanzar el potencial de membrana. La diferencia (V_{max} - V_x) introduce un factor de ganancia proporcional al valor del potencial de membrana en cada instante y lo mantiene en el intervalo [V_{max}, V_{min}].

Para la implementación en nuestro modelo de la función de inyección gradual de carga ha sido preciso realizar una aproximación de la función característica. En la Fig. 2 (b) se ilustra la aproximación realizada con 16 elementos.



(a) (b)

Fig. 2 (a) En cada intervalo de tiempo llega una nueva contribución sináptica. Este nuevo valor se introduce en la (HNR) y desplaza los valores anteriores un lugar hacia la derecha. Si no se recibe contribución sináptica se introducen ceros. **(b)** Función de inyección gradual de carga. Representación de f_{syn} usando 16 valores para modular la HNR.

b. Potencial de descanso: Cuando no llegan estímulos en la entrada de la neurona, el potencial de membrana (V_x) tiende a decaer hasta el valor prefijado o potencial de descanso (V_{rest}) debido al término de descarga pasiva con una constante de tiempo t_{rest} siguiendo la ec. (3). En la implementación hemos elegido un valor τ_{rest}=1/128 para poder ser implementado como un desplazamiento en vez de realizar la operación de multiplicación.

$$V_x = V_x + \tau_{rest}(V_{rest} - V_x) \quad (3)$$

c. Axon Hillock; Generación de impulsos de salida: Cuando el potencial de membrana (V_x) alcanza el umbral de disparo (V_{th}) el Axon Hillock dispara un impulso en la salida, (Ec. (4)). Al mismo tiempo el valor de V_x se decrementa hasta alcanzar el valor post-disparo (V_{post}). En este caso no hemos introducido ningún mecanismo para emular el periodo refractario post-disparo.

$$S_i(t) = \sum_f \delta\left(t - t_i^{(f)}\right) \tag{4}$$

Donde $t_i^{(f)}$ es el instante de tiempo en que se produce el disparo tras superar el umbral [20].

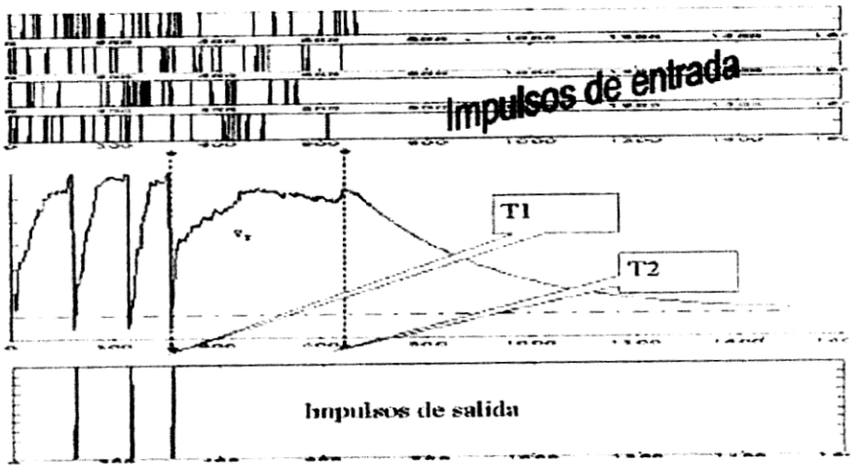


Fig. 3 Dinámica de la Neurona. La neurona recibe entradas (Gráfica de la parte superior) con diferentes frecuencias de pulsos excitatorios.

En la Fig. 3 se distinguen tres etapas con excitaciones distintas;

Etapa 1: [0, T1] El porcentaje de impulsos excitatorias a la entrada es suficiente como llevar V_x por encima del umbral de disparo. Se producen tres disparos (gráfica inferior).

Etapa 2: [T1, T2] El número de impulsos a la entrada no es suficiente para provocar el disparo de la neurona. El valor de V_x se mantiene estable próximo al umbral de disparo.

Etapa 3: [T2, 1600] El número de impulsos excitatorios en la entrada de la neurona es cero. En esta situación el único factor que afecta al valor del potencial de membrana es el término de decaimiento pasivo que lleva el valor de V_x hasta el potencial de descanso.

3 Implementación hardware

El modelo construido es un modelo de prueba que se ha programado sobre una Virtex XCV2000E [21]. Para poder simular grandes redes de neuronas de pulsos hemos optado por multiplexar el esquema de simulación. Este esquema consiste en mantener un núcleo de cálculo paralelo de tamaño (4, 8 ó 16) muy rápido al que se le pasa el contexto de cada neurona que se mantiene almacenado en bloques de memoria embebida (Embedded Memory Blocks o EMBs) dentro del dispositivo programable. De este modo en cada ciclo de evaluación computamos varias neuronas a la vez.

La evaluación de la red neuronal se lleva a cabo mediante la repetición de los pasos siguientes, en la Fig. 4 se ilustra este esquema de procesamiento:

1. El contexto de cada neurona contenido en memoria es almacenado en un Buffer de PreCarga (BPC). Esta carga se hace de forma secuencial dada la limitación secuencial de acceso a memoria del dispositivo programable.
2. La salida de cada neurona de la capa anterior es cargada en un vector de precarga.
3. Todas las unidades funcionales computan el nuevo estado neuronal en paralelo y el resultado es cargado en el BPL.
4. Todos los contextos de valores son almacenados en memoria embebida (EMBs).
5. Volver al paso 1 tantas veces como se desee.

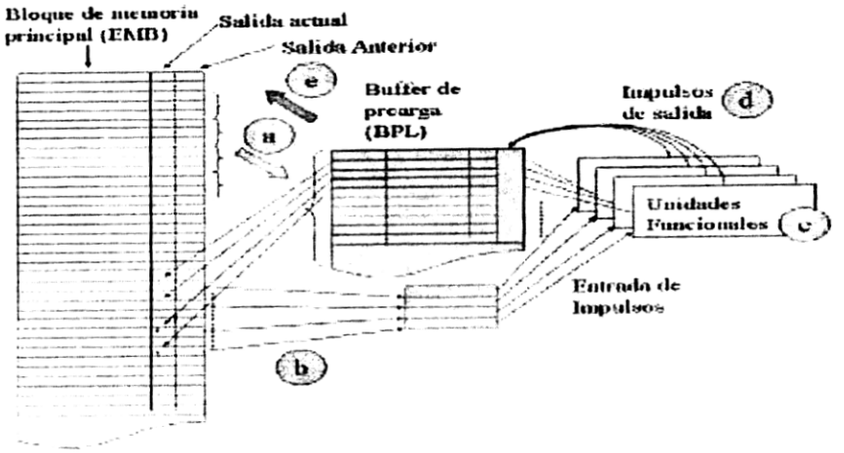


Fig. 4. Esquema general de computación multiplexado. Diferentes etapas: (a) Buffer de pre-carga (BPC) se carga con el contexto de las nuevas neuronas. (b) Las nuevas entradas se cargan desde las neuronas fuente. (c) La unidad funcional calcula el nuevo potencial de membrana y las salidas (d) El nuevo contexto es salvado en el BPC (e) y almacenado en los bloques de memoria embebida.

Un aspecto importante en la evaluación de la red es la no violación de la coherencia entre grupos de neuronas. Si una neurona está conectada con dos o más neuronas, esta neurona tendrá que esperar a que las neuronas que conectan con ella sean evaluadas para mantener la coherencia temporal. Cada neurona incluye en su contexto la dirección de la neurona que conecta con ella en cada sinapsis. De este modo en cada ciclo de evaluación se accede a las posiciones de memoria donde se almacenan las nuevas entradas a las sinapsis.

Una neurona completa necesita almacenar varios valores para mantener su contexto. Estos valores son los siguientes:

- **Dirección de las fuentes de conexión, las sinapsis y los pesos asociados.** La neurona toma como entrada la salida de varias neuronas de otros grupos. Estas entradas son codificadas como direcciones y almacenadas en cada contexto. Cada entrada sináptica tiene asociado un peso que más tarde puede ser modificado por la ley de aprendizaje. En este diseño hemos restringido los pesos a 8 bits y las direcciones de interconexión a 14 bits.

- **Historia Neuronal Reciente (HNR).** Es un registro que podemos diseñar de diferentes longitudes. La longitud determina cuánto queremos que influya la

historia pasada en el cálculo de la nueva contribución en V_x. En este ejemplo se han utilizado 16 elementos. Cada vez que se introduce un nuevo valor en la HNR es automáticamente desplazado hacia la derecha. De este modo el elemento más antiguo es discriminado. Cada elemento de HNR tiene una longitud de 11 bits.

- **Potencial de membrana (V_x).** Es el estado del potencial de membrana en cada instante. Nosotros usamos 17 bits para almacenarlo.
- **Salida actual de la neurona.** La salida de una neurona se almacena tras la evaluación del contexto. Nosostros utilizamos 1 bit para ello
- **Salida anterior de la neurona.** Para poder evaluar cualquier neurona de la red sin importar el orden en que se realice es preciso mantener el estado de la salida anterior. Así todas las neuronas conectadas a una dada tomarán en el ciclo actual la salida anterior de la neurona.

En el sistema construido tenemos la posibilidad de elegir y cambiar la función de inyección gradual de carga (f_{syn}) de forma fácil, puesto que está implementada con una tabla de consulta (LUT). Esto nos da la posibilidad de implementar diferentes dependencias temporales, sin más que cambiar el contenido de la tabla.

En la Fig. 5 se incluyen algunos resultados de la implementación en FPGA.

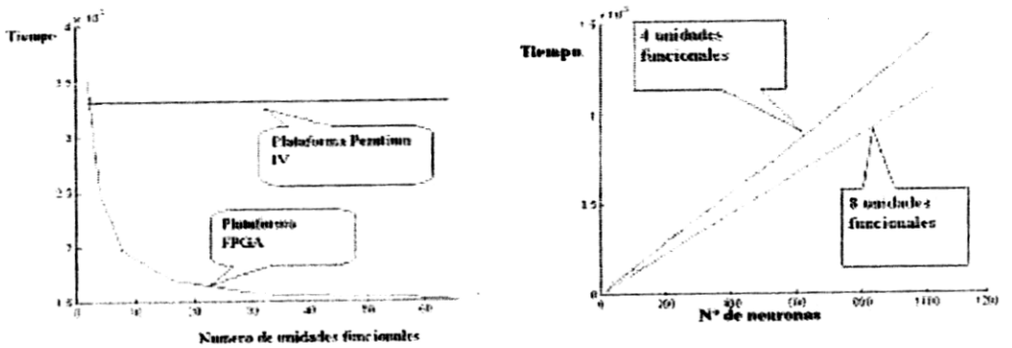


Fig. 5. (a). Ilustra el funcionamiento de la FPGA comparada con un Pentium IV convencional a (1.6 GHz). La prueba está realizada con 1760 neuronas que precisan 3.3 ms en un Pentium IV y 1.95 ms con la FPGA con 8 unidades funcionales paralelas. **(b).** Evolución del tiempo de cómputo en función del número de neuronas de la red y el número de unidades funcionales paralelas.

El cálculo interno de cada núcleo, una vez almacenado el contexto en el BPC, se realiza en los siguientes pasos:

1. La nueva contribución sináptica se calcula siguiendo la expresión (1).
2. Esta nueva contribución se carga en la primera posición del registro HNR. El resto de valores se cargan del BPC en las siguientes posiciones
3. Se realiza el producto C·f según la Ec. (2).
4. Realizamos el cálculo del término de decaimiento pasivo, siguiendo la Ec. (3).
5. Se calcula la salida de la neurona comparando V_x con el umbral de disparo.

Observación: Este proceso se realiza con varios núcleos de forma paralela.

Todos estos parámetros han sido extraídos usando Handel-C [22] como lenguaje de descripción de hardware de alto nivel. El entorno de desarrollo utilizado ha sido DK1.1 y la placa de prototipazo ha sido la RC1000 de Celoxica [22]. Esta placa incorpora un dispositivo

programable Virtex-E 2000 de Xilinx [21]. El código construido ha sido desarrollado de forma modular. Esta característica nos permite realizar de forma sencilla cambios de configuración.A continuación presentamos una tabla que resume las características de la implementación de cada diseño.

Ent. por neurona	Unid. funcionales	N°total neuronas	N° de slices	Max. Frec. (Mhz)	T. de comp. (ms)	Bloques (BMEs)
2	2	4	1832 9%	23.3	0.0055	24 15%
2	2	1024	1966 10%	20.2	1.4	65 60%
8	4	8	5476 28%	20.9	0.0011	36 22%
8	4	1760	5595 29%	20.5	2.9	160 100%
8	8	16	12011 62%	18.7	0.0018	36 22%
8	8	1760	12010 62%	18.7	1.9	160 100%

Tabla 1. Coste de implementación y tiempos de computación de diferentes estructuras neuronales.

4 Conclusiones

Hemos presentado una implementación reconfigurable de neuronas de pulsos que incorporan la dependencia del tiempo de inyección de carga y la conductancia sináptica como modelo sináptico. La conductancia sináptica ha sido simplificada en la mayoría de los modelos de simulación presentados en la literatura porque consume mucho tiempo. En esta aportación nosotros evaluamos los recursos hardwares necesarios para ello. Los modelos de respuesta sináptica que incluyen esta característica [20] son de especial interés para el estudio de los mecanismos de sincronización y coordinación de movimientos [19,20].

El esquema propuesto no utiliza esquemas de simulación dirigido por eventos (*event-driven simulation schemes*) [23], por lo tanto es sólo de interés para estructuras de neuronas con un alto porcentaje de actividad sináptica. También hemos descrito y evaluado un esquema multiplexado de evaluación neuronal usando núcleos funcionales paralelos. Este esquema de evaluación podría ser mejorado utilizando arquitecturas de pipeline para mejorar los tiempos de carga de los buffer de precarga y el tiempo de computación de las unidades funcionales.

Agradecimientos

Este trabajo ha sido financiado por SpikeFORCE (IST-2001-35271).

Referencias

1. SpikeFORCE, (EU Project: IST-2001-35271). http://www.spikeforce.org

2. Eckhorn, R.; Reitböck, H.J.; Arndt, M.; Dicke, D.: Feature Linking via Synchronization among Distributed Assemblies: Simulations of Results from Cat Visual Cortex. Neural Computation 2, pp. 293-307, (1990).
3. Delorme, A.; Thorpe, S.J.: Face identification using one spike per neuron: resistance to image degradations, Neural Networks, 14(6-7), pp. 795-804, (2001)
4. Smith, L.S.: A one-dimensional frequency map implemented using a network of integrate-and-fire neurons. ICANN98, (Springer-Verlag), LCNS pp. 991-996, (1998)
5. CORTIVIS: http://cortivis.umh.es/
6. Boahen, K. : A Retinomorphic Chip with Parallel Pathways : Encoding ON, OFF, INCREASING, and DECREASING Visual Signals. Journal of Analog Integrated Circuits and Signal Processing, Vol. 30, No. 2, pp. 121-135, (2002)
7. Jahnke, A.; Schoenauer, T.; Roth, U.; Mohraz, K.; Klar, H.: Simulation of Spiking Neural Networks on Different Hardware Platforms. ICANN97, LCNS, pp. 1187-1192 (1997)
8. Schoenauer, T.; Atasoy, S.; Mehrtash, N.; Klar, H.: NeuroPipe-Chip: A Digital Neuro-Processor for Spiking Neural Networks, IEEE Trans. Neu. Net., 13(1), pp. 205-213, (2002)
9. Hartmann, G.; Frank, G.; Schäfer, M.; Wolff, C.: SPIKE128K- An Accelerator for Dynamic Simulation of Large Pulse-Coded Networks, MicroNeuro97, (1997)
10. Hartmann, G.; Frank, G.; Schäfer, M.; Wolff, C.: SPIKE128K- An Accelerator for Dynamic Simulation of Large Pulse-Coded Networks, MicroNeuro97, (1997)
11. Shoenauer, T.; Mehrtash, N.; Janke, A.; Klar, H.: MASPINN: Novel Concepts for a Neuro-Accelerator for Spiking Neural Networks. VIDYNN'98, Stockolm (1998).
12. Janke, A.; Roth, U.; Klar, H.: A SIMD/dataflow architecture for a neurocomputer for spike-processing neural networks (NESPINN). Proc. MicroNeuro'96, pp. 232-237, (1996).
13. Schoenauer, T.; Atasoy, S.; Mehrtash, N.; Klar, H.: NeuroPipe-Chip: A Digital Neuro-Processor for Spiking Neural Networks, IEEE Trans. Neu. Net., 13(1), pp. 205-213, (2002)
14. Opher, I.; Horn, D.; Quenet, B.: Clustering with spiking neurons, ICANN'99, LCNS (Springer-Verlag), pp. 485-490, Edinburgh, (1999)
15. Hopfield, J.J.; Pattern recognition computation using action potential timing for stimulus representation, Nature, 376, pp. 33-36, (1995)
16. Rochel, O.; Martinez, D. ; Hugues, E. ; Sarry, F. : Stereo-olfaction with a sniffing neuromorphic robot using spiking neurons, Eurosensors, (2002)
17. Ros, E.; Pelayo, F.J.; Rojas, I.; Fernández, F.J.; Prieto, A.: A VLSI approach for spike timing coding, LNCS, (Springer-Verlag), 1629, 610-620, (1999)
18. Ros,E; Agis, R; Carrillo,R; Ortigosa, E; Pelayo, F.J; Prieto, A: Post-synaptic Time-dependent Conductances in Spiking Neurons: FPGA Implementation of a Flexible Cell Model. Accepted for publication in the LNCS, 2003.
19. Eckhorn, R.; Bauer, R.; Jordan, W.; Brosh, M.; Kruse, W.; Munk, M.; Reitböck: Coherent oscillations: A mechanism of feature linking in the visual cortex? Biol. Cyber. 60, pp. 121-130, (1988)
20. Gerstner, W., Kistler, W.: Spiking Neuron Models. Cambridge University Press, (2002)
21. Xilinx, http://www.xilinx.com/
22. Celoxica, http://www.celoxica.com/
23. Watts, L.: Event-driven simulation of networks of spiking neurons. Advances in Neural Information Processing Systems, Vol. 6, pp. 927-934, (1994)