TESIS
DOCTORAL

RODRIGO C. AGIS MELERO

ARQUITECTURAS PARA EL PROCESAMIENTO DE SISTEMAS NEURONALES PARA EL CONTROL DE ROBOTS BIOINSPIRADOS

Universidad de Granada
Departamento de Arquitectura
Y Tecnología de Computadores
Granada, 2007

Editor: Editorial de la Universidad de Granada
Autor: Roberto Carlos Agis Melero
D.L.: Gr. 1974- 2007
ISBN: 978-84-338-4461-3

ARQUITECTURAS PARA EL PROCESAMIENTO DE SISTEMAS NEURONALES PARA EL CONTROL DE ROBOTS BIOINSPIRADOS

Rodrigo C. Agis Melero

TESIS DOCTORAL

Directores: Eduardo Ros Vidal
Francisco J. Pelayo Valle
Eva Martínez Ortigosa

Granada 2007

Dpto. Arquitectura y Tecnología de Computadores.

D. Eduardo Ros Vidal, Profesor Titular de la Universidad de Granada, D. Francisco J. Pelayo Valle, Catedrático de la Universidad de Granada y Dª. Eva Martínez Ortigosa, Profesor Contratado Doctor de la Universidad de Granada,

CERTIFICAN

que la memoria titulada

ARQUITECTURAS PARA EL PROCESAMIENTO DE SISTEMAS NEURONALES PARA EL CONTROL DE ROBOTS BIOINSPIRADOS

ha sido realizada por D. Rodrigo C. Agis Melero bajo nuestra dirección en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada, para optar al grado de Doctor.

Granada, a 20 de Julio de 2007

Fdo. Eduardo Ros Vidal Ddo. Fdo. Francisco J. Pelayo Valle

Eva Martínez Ortigosa

ARQUITECTURAS PARA EL PROCESAMIENTO DE SISTEMAS NEURONALES PARA EL CONTROL DE ROBOTS BIOINSPIRADOS

Memoria presentada por

Rodrigo C. Agis Melero

Para optar al grado de

Doctor en Informática por
la Universidad de Granada

Fdo. Rodrigo C. Agis Melero

A Maria, Felipe y Tete

"El aspecto más triste de la vida actual es que la ciencia gana en conocimiento más rápidamente que la sociedad en sabiduría"

"The saddest aspect in the current life is that the science wins more quickly than the society in knowledge in wisdom"

Isaac Asimov (1920-1992)

Agradecimientos

Más que agradecimiento, mi elogio se transforma en reconocimiento sincero hacia aquellas personas que con su paciencia y esfuerzo me han ayudado de forma totalmente altruista a saltar los "baches" del camino y llegar a la culminación de este trabajo.

En especial mi reconocimiento incondicional a Eduardo Ros y Francisco Pelayo, sin que el orden en su nombramiento sirva de medida de su dedicación, por su apoyo, entrega incansable, tesón, ilusión y experiencia vertida en mí durante todo el proceso de investigación. No puedo olvidar a Eva Martínez por introducir siempre una nota de humanidad durante las innumerables cargas de trabajo.

Que decir de mis compañeros de proyectos Europeos. Destaco la eficiencia crítica llevada hasta el límite de Richard, el tesón investigador incansable de Javier Díaz, el espíritu empresario de Sonia, el espíritu trabajador de Rafa y como no, todo el compañerismo y la profesionalidad de la gente del proyecto Cortivis; Samuel, Antonio y Chistrian. Gracias a todos por las innumerables opiniones científicas que más de una vez me han sacado de algún atolladero. En especial mi reconocimiento a Javier Díaz por su ayuda en el desarrollo de ideas para arquitecturas hardware eficientes.

Finalmente dar las gracias al Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada por darme la oportunidad de pertenecer a él durante el desarrollo de este trabajo de Tesis doctoral.

Esta Tesis ha contado con la financiación de los proyectos Europeos SPIKEFORCE (*Spiking Neurons for Robot Control*) y SENSOPAC *Sensory motor structuring of perception and action for emerging cognition*) [117] [115].

Resumen

La presente Tesis doctoral profundiza en el estudio de redes neuronales artificiales basadas en pulsos. En este ámbito la Tesis propone diferentes arquitecturas hardware para su implementación en dispositivos reconfigurables (FPGA), bien siguiendo esquemas de procesamiento *time-driven* (simulación dirigida por tiempo). O bien siguiendo esquemas *event-driven* (simulación dirigida por eventos).
Aporta soluciones de control alternativas en el campo de la robótica y profundiza en el estudio de arquitecturas para la simulación de redes neuronales masivas de millones de neuronas procesando información senso-motor en tiempo real. Estas soluciones se han materializado en plataformas de codiseño hardware-software con distintos grados de autonomía del hardware (FPGA) respecto del software (un PC como Host) que simulan en tiempo real el funcionamiento de dichas redes.

En una segunda etapa, esta Tesis apuesta por la simulación con robots reales, como mecanismo de prueba de los sistemas neuronales artificiales de control. La idea parte de la dificultad de describir la interacción con el mundo real de manera exacta en un entorno de simulación. Para ello profundiza en el concepto de "*Embodiment*" que pone de manifiesto la necesidad de disponer de un cuerpo físico como mecanismo de aprendizaje para la generación emergente de conocimiento.
En este ámbito, la Tesis describe el funcionamiento de dos plataformas robóticas construidas y adaptadas para su control mediante pulsos neuronales. Los resultados obtenidos muestran que imitando en mayor o menor medida a la biología, es posible construir circuitos neuronales lo suficientemente funcionales como para ser tenidos en cuenta para el control de sistemas robóticos bio-mórficos con estructuras físicas complejas.

Para una mejor comprensión del enfoque del trabajo realizado se ha decido dedicar un capítulo a parte. Véase el Capítulo 1 para mayor de talle.

Summary

This PhD Thesis is focussed in the study of spiking neural networks. In this framework the presented work presents different hardware architectures that are implemented in reconfigurable devices (FPGAs). Different approaches are proposed adopting *time-driven* or alternatively *event-driven* processing schemes.

The work presents alternative control approaches in the field of robotics and studies computing architectures for the simulation of massive spiking neural networks of millions of neurons processing sensorimotor information in real-time. These proposed approaches have been implemented in two hybrid Hardware/Software platforms with different levels of autonomy of the hardware (stand-alone and co-processing strategy) with respect to the software modules (in a PC as a host computer) that simulated in real-time these large scale networks.

In a second stage, this Thesis focuses on experiments with real-robots, as validation methodology of the control neural networks under study. The choice of working with real robots instead of simulated ones in motivated by the difficulty of describing in a realistic way the interaction with the real-world in a simulated framework. Therefore, the work here also adopts the "Embodiment concept" which stresses the necessity of having a physical body as learning mechanism for the knowledge emergence generation.

In this field, the Thesis describes two robotic platforms built and adapted for being controlled by spiking neural systems. The obtained results show that imitating in more or less detail the biology is feasible building neural circuits which represent valid alternatives to be considered for control of biomorphic robots with complex physical structures.

Introduction and motivation

Building artificial machines capable of performing tasks with "certain level of intelligence" (as humans do) has been one of the scientific goals in recent scientific history. Even having computers capable of performing millions of computations per second

and diverse programming languages this goal remains as an open challenge. This is partly because current computing technology adopts mainly sequential processing strategies while natural systems (animals) use massive parallel computing in their central nervous systems for multi-sensorial perception tasks.

The interaction between artificial systems and the world is a challenge for classical problem solving methods in computation. Most of these interactions cannot be described as algorithms (nor can be simulated). Building an artificial model of the world or "environment" with a certain level of complexity is beyond current computers capabilities. Instead of building a global model of the environment, biological systems have developed mechanisms to dynamically adapt to the "experimented environment" and how to interact with it. Biological systems learn dynamically how to optimize their interactions with the environment through experience. This is done in the Central Nervous System (CNS) through neural circuits that receive, exchange, process and send information to their environment. This is done by means of spikes that encode all the signals being transferred or processed in the CNS. Spikes represent a good way of transmitting information with a low power cost and robust to noise. Building computing systems inspired in biological systems is not straightforward since we are trying to emulate a massive parallel computing system. For this purpose in this work we have chosen reconfigurable hardware (FPGA devices) that allows implementing parallel computing schemes. Even so, the number of computing elements that can be implemented with this technology is much reduced (far from the millions of units of biological systems). Nevertheless, we describe how to adopt opportunistic computing strategies to exploit efficiently certain characteristics of current technology (for instance using multiplexing techniques to take full advantage of high clock frequencies of current digital circuits).

Framework of research task

Before starting the work of this PhD different robotic platforms were developed that helped the author to gain expertise in this field. Furthermore we realised about the inherent difficulty of "creating new agents" and the challenge of devising new control

schemes. This previous creative effort has highly motivated the work presented in this Thesis.

The Thesis has been done in the framework of two EU projects:

- **SpikeFORCE:** Real-Time Spiking Networks for Robot Control (FP5-IST-2001-35271). (01-05-2002 till 30-09-2005)

- **SENSOPAC:** Sensory motor structuring of perception and action for emerging cognition (FP6-IST-028056). (01-01-2006 till 30-12-2010)

The goal of SpikeFORCE was to develop models of spiking neural structures biologically plausible and study mechanisms for their utilization in robot control tasks. SENSOPAC represents a step beyond the control task and addresses the study of how, using biomorphic robots and bio-inspired control schemes we can evaluate efficient control and active sensing strategies for exploration tasks. In this project is studied how "cognitive notions" are structured by means of abstracted models in biological models, for instance in olivo-cerebellar structures and other neural subsystems.

For both projects the development of efficient spiking neural network simulation technologies to be used in real-time robot control tasks represents an issue of critical importance. Furthermore, the study of new computation schemes radically different to the conventional sequential processors based techniques remains as breaking through point in itself. The new "computing architectures" presented in this work can be considered as scalable architectures in which the input/output information is represented by means of neural spikes. The processing architecture is a specific purpose processor with a single "instruction": process a spike.
In this kind of architectures in possible to study the scalability issue and new parallelization techniques that become of specific interest provided the large number of computing resources currently available on single chips (reconfigurable hardware devices).

Main original contributions

The main original contributions of this work are the following:

1. Based on biologically plausible neural models a reference model has been adopted. This constitutes a trade-off between biological plausibility, computation complexity and customization capability adjusting specific functions and parameters (passive membrane potential decay, gradual injection of charge, etc.).

2. Based on the previous model, a time-driven processing architecture for simulating spiking neurons has been developed. This has been done designing a segmented datapath in order to take full advantage of the inherent parallelism of the computing resources of the FPGA devices. The system data (mainly neural spikes) has been structured in diverse configurations based on on-chip embedded memory and extended SRAM memory chips on-board. This allows the architecture to be fully scalable and the performance can be multiplied replicating functional units on the same chip.

3. A hybrid Software/Hardware platform has been designed for the utilization of the time-driven simulation engine. The network topology and the learning are simulated in software while the neural state variables are updated through dedicated hardware in the co-processing board (developed processing architecture). The gain in performance provided by the use of the co-processing board has been evaluated simulating in real-time an artificial cerebellum.

4. A spiking neural processing architecture driven by events has been developed. In this case the whole system has been implemented in the FPGA device (supported by several external memory chips). Software modules have been developed for monitoring the simulations. The different risks in the segmented datapath have been debugged. The whole system has been implemented in the reconfigurable platform. This facilitates its use as embedded control system. Outstanding performance rates

(with more than two million spikes per second) have been obtained.

5. Two robotic platforms have been designed and complemented with specific communication modules for the developed simulation engines. In these platforms the direct access in real-time to sensors and motors has been facilitated. These platforms constitute very useful validation tools for bio-inspired processing schemes in which the continuous (real-time) interaction of the agent with its environment is required.

6. Making an analogy between the muscle functions in motor tasks and the dynamics of systems based on servo motors, a conversion mechanism of signals has been implemented to allow the movement control of the biped robot through the simulation engine.

Objectives and organization

This Thesis presents different alternatives for designing massively hardware architectures. These architectures allow building neural circuits that emulate with a certain level of abstraction biological cell properties. Concretely, we will focus on simulating the cerebellum, due to its importance for coordination, movement control and motor learning.

Contrary to other classic artificial neural models, in which the internal dynamics of cells are just described by a predefined activation function (for instance a sigmoid function) the research work presented here focus on biological models of cells characterized by properties such as passive membrane potential decay, electrical coupling, gradual injection of charge, etc. In this way, it is possible replicating certain functionalities that seem to have a functional role in biological systems.

Many authors support the idea that the real potential of biological neural systems is facilitated by the network topologies and not by specific cell properties [130] [74]. We think that both characteristics (cell temporal dynamics and network topologies) play complementary roles in the computing capabilities of biological nervous sys-

tems. This continuously motivates interdisciplinary works that try to build bridges between technology and neurobiology. Simulating biologically plausible cells and networks suggest new system properties that can open new research lines in the neurophysiologic field.

This thesis represents an effort also in building and dealing with real robots for evaluating and validating the neural computing engines that are described. This effort is partially motivated by the "embodiment" concept [64] [119] that suggest that is necessary a physical body to study mechanism of how knowledge emerges.

Another important aspect that is addressed in this work is the computing complexity of simulating massive neural systems (thousands to millions of neurons). To give an order of magnitude the human brain consists in approximately 20 thousand millions of neurons (with at least 1000 different types and 60 to the power of 14 synapses). Interestingly enough the cerebellum has 5 times this number of neurons (approximately one hundred thousand millions of neurons). The cerebellum is described as a set of computing elements with a well defined structure (climbing fibers, mossy fibers reaching granular cells, parallel fiber interconnecting Purkinge cells, etc). Diverse studies prove that only between a 5 to 10% of the cells are active at the same time (that brain uses sparse coding, especially in certain areas such as the Granular layer) which is also limited by the power consumption.

With the goal of simulating biological neural systems this work describes high performance computing engines for spiking neural networks and also how they deal with real robots.

For facilitating the easy reading of the Thesis, here we briefly describe a summary of the different chapters:

- **Chapter 1:** We briefly introduce the objectives and problems addressed in the different chapters of the work for the simulation of bio-inspired control systems.

- **Chapter 2:** We introduce different models of bio-inspired neurons (integrate and fire model, Spike Response Model,

etc) and it briefly describes also certain Spike Time Dependent Plasticity (learning rules). This chapter tries to clearly distinguish conventional artificial neural networks and bio-inspired spiking neural networks with a certain level of biological plausibility.

- **Chapter 3:** We describe a co-processing computing architecture defined in reconfigurable hardware. We address in detail the characterization of the computing platform and its performance evaluation compared to sequential processing schemes. This computing engine addresses time-driven simulations of spiking neural networks.

- **Chapter 4:** We describe in detail an event-driven computing engine with a pipelined datapath. We analyze its performance and the computing risks in its computing datapath and how the affect the system performance. One of the major contributions of this chapter is the parallel event selection architecture that enhances significantly the performance of the platform.

- **Chapter 5:** The robotics and its utilization to evaluate the hardware computing engines is one significant contribution of this Thesis described in this chapter. Here we describe different mechanic systems (robotic platforms) developed to study specific issues such as how to deal with robotic dynamics.

- **Chapter 6 and Chapter 7:** In these chapters we briefly describe the main contributions of the research work presented in this Thesis. We also indicate the scientific production of this research effort.

Note: Some paragraphs from this summary as such “Framework of research” or “Main scientific production” have been repeated in the charter 7. In this way, we think the charter 7 is more completes and easier to understand the conclusions of this Thesis.

Main scientific productions

Main scientific productions with noteworthy scientific impact (SCI) are presented in this part. A complete list of papers also with SCI is presented in the Chapter 7.

1. Eduardos Ros, Eva M. Ortigosa, Rodrigo Agis, Richard Carrillo and Michael Arnold, "**Real-Time Computing Platform for Spiking Neurons (RT.Spike)**", IEEE transactions on Neural Networks, July 2006, Volume 17, Number 4, pp. 1050-1063, (ISSN 1045-9227).

2. E. Ros, R. Carrillo, E. M. Ortigosa, B. Barbour, and R. Agís. **"Event-Driven Simulation Scheme for Spiking Neural Networks Using Lookup Tables to Characterize Neuronal Dynamics"**. Neural Computation. (2006) 18(12): 2959-2993.

3. Rodrigo Agís, Javier Díaz, Eduardo Ros, Richard Carrillo, Eva. M. Ortigosa, "**Hardware Event-driven Simulation Engine for Spiking Neural Networks**", International Journal of Electronics (Taylor & Francis Group), Volume 94, Issue 5, May 2007, pp. 469-480.

Índice General

Capítulo 1
Introducción y motivación

Este capítulo, a modo de resumen, aborda el paradigma de la simulación de redes neuronales basadas en impulsos, así como el concepto de "*Embodiment*" como base para la simulación y aprendizaje con robots reales. También pone de manifiesto la necesidad de la implementación de arquitecturas hardware de altas prestaciones como alternativa real para la simulación de redes de millones de neuronas. Por último, se ponen de manifiesto el conjunto de objetivos y contenidos que aborda este trabajo de Tesis doctoral.

1.1 Introducción

Conseguir construir máquinas capaces de realizar tareas con cierta "inteligencia" al igual que lo hacen los seres humanos, ha sido uno de los principales objetivos de los científicos a lo largo de la historia. Paradójicamente, a pesar de disponer de herramientas para el diseño de máquinas inteligentes (lenguajes de programación avanzados, materiales de alta tecnología etc.), no se ha conseguido hasta ahora. En parte porque toda la filosofía de diseño se soporta sobre la utilización de máquinas de procesamiento con arquitectura secuencial (arquitecturas de tipo Von Neumann) mientras que tratamos de emular un sistema (sistema nervioso central) constituido por millones de elementos de cómputo en paralelo.

La interacción de sistemas artificiales con el mundo real es un desafío para los métodos clásicos de resolución de problemas en computación. Este tipo de interacciones no pueden ser descritas fácilmente mediante un enfoque algorítmico tradicional. Sencillamente porque un algoritmo, que describe un comportamiento de un sistema artificial interactuando con el mundo real, posee una descripción parcial y reducida. Esto es debido a la enorme dificultad de describir toda la dinámica del mundo real utilizando métodos y lenguajes de programación, clásicos.

La solución la aporta la biología, cambiando la dirección del enfoque. En vez de describir el entorno y las interacciones con él mediante reglas predefinidas u otros mecanismos, la biología aprende qué reglas definen el entorno al interactuar con él, para proporcionar una respuesta coherente y beneficiosa.
Con esta idea en mente, los circuitos neuronales biológicos, reciben, intercambian, procesan y emiten información interactuando con el entorno, mediante señales de pulsos codificados en frecuencia y fase. Estos pulsos, denominados "*Spikes"*, representan eventos temporales y son una forma ventajosa de transmitir información con un coste energético reducido y con una inmunidad frente al ruido muy significativo.
Sin embargo, la construcción de sistemas basados en la biología requiere recursos de computación que exploten eficientemente el paralelismo inherente en las interacciones de los sistemas físicos con el mundo real. Estos requisitos son difícilmente cubiertos a través de tecnologías secuenciales basadas en procesadores clásicos.

Para dar soporte a este nuevo enfoque, la tecnología actual aporta una alternativa. Esta alternativa está basada en la utilización de hardware reconfigurable con dispositivos CPLD "*Complex Programmable Logic Device"* ó FPGA "*Field Programmable Gate Array*", que permite no sólo el tratamiento de la información de forma paralela sino la implementación directa en hardware de los circuitos de control necesarios.

A pesar de ello, el volumen de unidades de procesamiento de un circuito neuronal biológico así como su conectividad es difícilmente alcanzable por la tecnología actual. Sin embargo, como se describe a lo largo de este trabajo de Tesis doctoral, es posible adoptar diversas estrategias de procesamiento ventajosas que ayudan a equilibrar la balanza y conseguir sistemas de una complejidad significativa.

1.2 Objetivos y organización de la memoria

La Tesis analiza en profundidad diferentes alternativas para el desarrollo de arquitecturas hardware de procesamiento masivo. Estas arquitecturas permiten la construcción de circuitos neuronales artificiales capaces de emular fielmente la funcionalidad de los circuitos biológicos. Concretamente se hace un especial hincapié en el estudio del cerebelo, por su importante papel como mecanismo de coordinación, control de movimientos y aprendizaje a nivel motor.

En contraposición con los modelos neuronales artificiales clásicos, donde el funcionamiento interno de las neuronas se rige por una suma ponderada y una función de activación predefinida (entiéndase por función típica; sigmoidal, escalonada etc.) esta Tesis aborda el estudio de modelos biológicos donde aparecen términos como decaimiento pasivo, potenciales de membrana, acoplamiento eléctrico etc. De este modo es posible replicar, de forma artificial, un circuito que la biología ha perfeccionado durante millones de años, aproximando su funcionamiento con cierto nivel de detalle.

Muchos autores creen que el verdadero potencial de los sistemas neuronales biológicos, no se centra tanto en el funcionamiento interno de una neurona (recordemos que una neurona puede ser vista

como un elemento integrador bastante ruidoso [130] [74]) sino en la topología de interconexión para formar circuitos anatómicos precisos. Sin embargo, no cabe duda que la unión de ambos (modelos de funcionamiento interno neuronal con topologías de circuitos neuronales específicas) proporciona una herramienta de estudio y evaluación prometedora.
Con esta idea en mente y estudiando ciertas áreas del cerebro humano involucradas en tareas de control, es posible establecer un puente de realimentación entre la tecnología y la neurobiología. Es decir, los hallazgos funcionales descubiertos durante el proceso de simulación de sistemas neuronales, que no han sido observados en los estudios neurobiológicos, pueden abrir nuevas líneas de investigación en el campo de la neurología, neurobiología y neurofisiología.

Esta Tesis apuesta por la simulación con robots reales, como mecanismo de prueba de los sistemas neuronales artificiales de control. La idea parte de la dificultad de describir la interacción con el mundo real de manera exacta en un entorno de simulación. Para ello profundiza en el concepto de "*Embodiment*" [64] [119] que pone de manifiesto la necesidad de disponer de un cuerpo como mecanismo de aprendizaje para la generación emergente de conocimiento. Es más, este concepto sólo entiende emergencia de "inteligencia" como una forma de adaptación de un agente a su entorno para diversas tareas.

Otro aspecto importante que se aborda es la magnitud del problema de cómputo que supone la simulación de redes neuronales masivas (entiéndase por redes neuronales de millones de neuronas). Como ejemplo, el cerebro humano (incluyendo cerebelo) tiene aproximadamente 100.000 millones de neuronas con al menos 1000 tipos diferentes y 60^{14} sinapsis (estas cifras dependen de la edad del individuo y por tanto es un dato promedio aproximado, pero en media son para un cerebro de aproximadamente 1350 gramos) [132] [19]. Paradójicamente, el cerebelo cuenta con más de la mitad de todas las neuronas del cerebro a pesar de ocupar tan sólo un 10% del volumen total [59] [19]. En un milímetro cuadrado de corteza cerebral se encuentran entre 10.000 y 100.000 conexiones sinápticas.
A modo de inciso, destacar que las neuronas no se encuentran solas, existen otras células denominas células *Gliales* que dan soporte a

las neuronas estableciendo divisiones y aislando grupos neuronales. El número de estas células es aproximadamente entre 10 y 50 veces el número de neuronas y, hasta donde se sabe, solamente realizan tareas meramente de soporte (como alimentación) sin intervenir en la elaboración de la información.
Con esta idea en mente, si se describe el cerebro como la unión de un conjunto enorme de unidades de procesamiento distribuidas y caracterizas por una arquitectura de interconexión que sigue ciertos patrones regulares, (entiéndase por estructuras neuronales dedicadas a tareas concretas) y aunque solamente un porcentaje pequeño de ellas estuviesen activas al mismo tiempo, entonces es fácil imaginar el volumen de procesamiento de información que se maneja.

Diversos estudios demuestran que solamente entre un 5 y un 10 % (tan sólo se llega a porcentajes próximos a 90% en ataques epilépticos) de las neuronas que componen el cerebro están activas al mismo tiempo (debido principalmente a limitaciones de consumo energético). Aun así el procesamiento de la actividad generada por un circuito neuronal de estas dimensiones, requiere la construcción de circuitos de altas prestaciones que exploten eficientemente velocidad y distribución de carga.

Como se estudia a lo largo de los capítulos de esta Tesis, existen diversas alternativas para conseguir máquinas de procesamiento masivo orientadas a simulaciones neuronales. A modo de ejemplo se plantean la utilización de “Clusters” de computadores, donde cada nodo puede simular una parte de la red neuronal mientras que las comunicaciones entre ellos simulan el envío y recepción de eventos [89] [129]. Esta opción puede ser válida como primera aproximación, pero resulta una solución parcial debido a otros requisitos tales como portabilidad (recordemos que necesitamos sistemas empotrados para su uso con robots autónomos) o consumo energético. La otra alternativa apuesta por la tecnología de hardware reconfigurable capaz de obtener resultados igual de prometedores [72] [38] [124] [44] [25].

Con objeto de facilitar y describir el trabajo realizado, a continuación se enumera y resume la organización en capítulos de este trabajo:

- **Capítulo 1:** De forma introductoria describe los objetivos y problemas que se abordan en los diferentes capítulos de la Tesis para la simulación de sistemas de control bio-inspirados.

- **Capítulo 2:** Introduce diferentes modelos para la simulación de neuronas bio-inspiradas (modelo de integración y disparo, SRM "*Spike Response Model")* así como el paradigma de las leyes de aprendizaje. Hace especial hincapié en la diferencia entre redes neuronales clásicas y redes neuronales (biológicamente más plausibles) que procesan pulsos. De este modo se establece una clara diferencia entre los modelos neuronales de computación clásica y los modelos neuronales más realistas.

- **Capítulo 3:** De forma detallada, aborda la utilización de dispositivos de hardware reconfigurable como dispositivo coprocesador para la simulación de redes neuronales masivas. Se abordan conceptos de paralelismo y diseño de hardware estableciendo comparativas de rendimiento en relación con procesadores secuenciales de última generación. Se aborda el estudio en profundidad de un simulador dirigido por tiempo.

- **Capítulo 4:** Describe en detalle un simulador de alto rendimiento dirigido por eventos con arquitectura en "*pipeline"* con un camino de datos segmentado analizando en profundidad el rendimiento del sistema así como los riesgos que aparecen en el cauce. Como aportación más destacable, se describe una arquitectura de ordenación de eventos paralela que multiplica las prestaciones.

- **Capítulo 5:** La robótica y la utilización de sistemas autónomos artificiales son una de las herramientas esenciales para evaluar los sistemas hardware desarrollados en este trabajo de Tesis doctoral. El capítulo aborda diversas arquitecturas mecánicas desarrolladas al efecto así como el estudio de implementaciones de control alternativas como es el caso de los algoritmos genéticos demostrando soluciones sencillas en este ámbito.

- **Capítulo 6 y Capítulo 7:** De manera concisa resume, en castellano y en inglés, las principales aportaciones obtenidas con la realización del presente trabajo de Tesis doctoral.

Capítulo 2
Modelos neuronales de pulsos y aprendizaje

La funcionalidad de una neurona puede ser descrita a través del análisis de diferentes procesos electroquímicos, sin embargo la dinámica de estos procesos es compleja. Una aproximación más sencilla consiste en la utilización de modelos eléctricos, que desde una perspectiva funcional, imitan de forma realista las características de las células nerviosas. En este capítulo se describen diferentes modelos neuronales (integración y disparo, SRM "*Spike Response Model"*, etc.) así como las diferencias entre los modelos de computación neuronal clásico y los modelos biológicos objeto de estudio de este trabajo. Por último se aborda el paradigma de las leyes de aprendizaje que aporta la biología, como base de desarrollo y optimización de los circuitos neuronales.

2.1 Introducción

Cualquier célula del organismo humano tiene un potencial de membrana, es decir una diferencia de voltaje entre el interior y el exterior de la célula. Sin embargo, sólo las neuronas junto con las células musculares son capaces de generar señales eléctricas que pueden ser conducidas a largas distancias. Estas señales eléctricas son generadas por el flujo de iones que traspasa la membrana celular a través de canales iónicos especializados. Recordemos que los canales iónicos son proteínas que controlan el paso de corriente a través de la membrana celular.
Estos canales actúan como compuertas que se cierran o se abren de acuerdo a los estímulos externos y producen el incremento o decremento del potencial de membrana.
Las neuronas se interconectan entre si mediante sinapsis. Existen sinapsis aferentes y sinapsis eferentes dependiendo del sentido del flujo de información (entrada o salida). Las sinapsis son el lugar donde se produce la estimulación de los canales iónicos y puede establecerse en las dendritas (sinapsis aferentes) que no son más que prolongaciones del protoplasma celular (capa externa de la célula) o en las ramificaciones al final del Axón (sinapsis eferentes).
Es importante hacer notar que las neuronas en general no se tocan físicamente unas con otras sino que existe un espacio muy pequeño en la sinapsis denominado "espacio sináptico" que aísla unas de otras. Aunque existen también sinapsis eléctricas, que son conexiones a través de las cuales las células pueden intercambiar directamente iones.

El mantenimiento o almacenamiento de un potencial de membrana determinado depende, entre otros factores, del tamaño de la célula. A modo de condensador las neuronas acumulan carga eléctrica conforme llegan los impulsos nerviosos procedentes de otras células debido a que éstos desencadenan la apertura y cierre de los canales iónicos. A partir de cierto umbral, intrínseco a cada tipo de célula nerviosa, se produce una descarga repentina a través del *cuello axónico* produciéndose un pulso, evento o *Spike*. Este pulso viaja hasta la sinapsis de la siguiente célula nerviosa (véase la Ilustración 1) produciendo la apertura de los canales e incrementado o decrementando el potencial en la membrana de la célula objetivo.

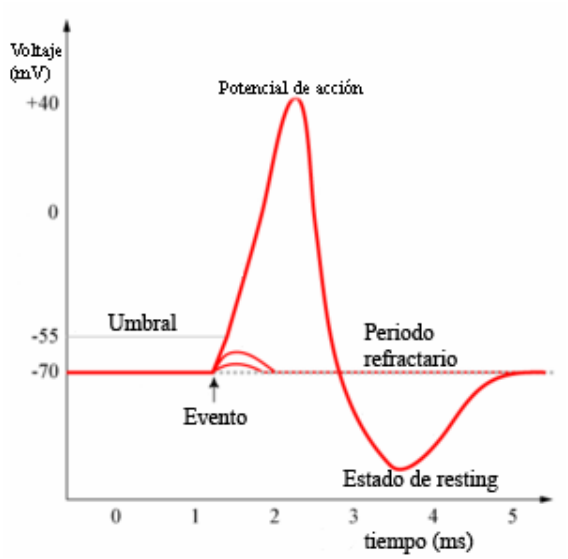


Ilustración 1: Comportamiento del potencial de membrana con la llegada de un conjunto de eventos con distinta relevancia sináptica.

Como hemos visto de forma muy simplificada, una neurona real basa su funcionamiento en procesos electroquímicos. Debido a la complejidad de estos procesos, la teoría de computación de redes neuronales clásicas ha modelado la neurona desde un punto de vista funcional macroscópico, como unidades de proceso muy simplificadas. En este ámbito clásico, todas ellas comparten el disponer de una unidad de conexiones de entrada, una función de activación y, por último, una unidad de generación de salida. Esta última suele ser analógica (valor continuo de salida en un intervalo predefinido) o discreta.

Fuera de éste ámbito, los modelos de neuronas que se abordan en este estudio, son simplificaciones funcionales de células nerviosas reales. Aunque se siguen utilizando modelos simplificados que hacen viable implementaciones eficientes, en esta línea de estudio se consideran aspectos sumamente biológicos tales como: potenciales de membrana, conductancias sinápticas, umbrales de disparo dinámicos, periodo refractario etc. También se establecen importantes diferencias referentes a la generación de la salida, quedando definida mediante trenes de pulsos o *Spikes* codificados en frecuencia y en tiempo (entiéndase con correspondencia temporal).

En las secciones siguientes se describen diferentes modelos funcionales de simulación de neuronas, que son aproximaciones simplificadas del funcionamiento real de las células nerviosas.

2.2 Modelo neuronal de integración y disparo

El modelo neuronal de integración y disparo puede ser visto como una simplificación del modelo de Hodgkin y Huxley [46] para las células nerviosas. Hodgkin-Huxley utilizaron como base de experimentación células nerviosas de un calamar gigante, pero sus conclusiones pueden ser generalizadas para cualquier tipo de células nerviosas. El modelo de Hodgkin y Huxley describe las leyes del movimiento de los iones en la membrana de las células nerviosas durante un potencial de acción a través de 4 pares de ecuaciones diferenciales y otras 8 ecuaciones complementarias dependientes del estado de los parámetros del modelo.

El modelo de integración y disparo está basado en un sencillo circuito compuesto por: un condensador *(C),* emulando la capacidad de la membrana celular, una resistencia *(R),* imitando la resistencia intrínseca de la célula y una fuente de intensidad *I(t)* que modela la inyección de corriente externa procedente de otras células (véase la Ilustración 2). Obsérvese que en este caso no se modela la conductancia sináptica de forma separada, sino que se integran todos los eventos sinápticos como un todo (se utiliza una única fuente de corriente).

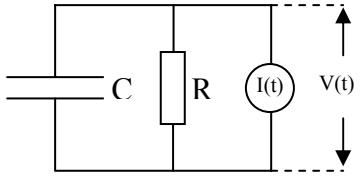


Ilustración 2: Modelo electrónico de integración y disparo válido para potenciales de membrana por debajo del umbral de disparo.

Utilizando las leyes de Kirchhoff es posible resolver el circuito obteniendo la Ecuación 1. Es decir, la intensidad total inyectada en la neurona es igual a la suma de la intensidad que pasa por la resistencia (*R)* más la intensidad que carga el condensador (*C)*.

$$I(t) = C\frac{dV}{dt} + \frac{V}{R}$$

Ecuación 1

Para incorporar un comportamiento acorde con la biología, es necesario incluir un umbral de disparo cuando el potencial de membrana supera cierto valor θ en el instante t_s (véase la Ecuación 2).
En este instante se produce un impulso nervioso y el potencial de membrana cae hasta un potencial de descanso o reposo ω. En este caso definido con valor *0*.

$$Si V(t_s) \geq \theta \; entonces \; \lim_{\varepsilon \to 0, \varepsilon \rangle 0} V(t_s + \varepsilon) = \omega = 0$$

Ecuación 2

El comportamiento sub-umbral queda definido por la Ecuación 1, es una ecuación diferencial. Es decir, una ecuación que relaciona la función *V(t)* con su derivada *d(V)/dt.* Obsérvese que la relación es lineal hasta el umbral de disparo θ. A partir de entonces *V* deja de serlo debido a singularidad introducida por la Ecuación 2.

Para buscar una solución a la Ecuación 1 es necesario establecer un valor inicial V_0 para el potencial de membrana en el instante t_0 (véase la Ecuación 3).

$$V(t_0) = V_0$$

Ecuación 3

Suponiendo que en la Ecuación 1 en el instante *t* no se produce actividad sináptica, es decir *I(t) = 0* se puede obtener la Ecuación 4.

$$C\frac{dV}{dt} + \frac{V}{R} = 0 \qquad \frac{dV}{V} = -\frac{dt}{RC}$$

Ecuación 4

Integrando en ambas partes de la Ecuación 4 entre V_0 y *V(t)* y entre t_0 y *t* se obtiene la Ecuación 5.

$$\int_{V_0}^{V(t)} \frac{dV}{V} = -\int_{t_0}^{t} \frac{dt}{RC}$$

Ecuación 5

Resolviendo la integración en la Ecuación 5 y despejando *V(t)* resulta la Ecuación 6

$$V(t) = V_0 \cdot e^{-\frac{t-t_0}{RC}}$$

Ecuación 6

La Ecuación 6 representa la solución para el caso en que *I(t) = 0*. Es decir, cuando no se produce actividad sináptica. La solución para cualquier *I(t)* puede ser obtenida conociendo la respuesta impulso del sistema estableciendo una condición de contorno específica para la solución. (Recordemos que la función impulso o función de Dirac *δ(t)* es una función de altura infinita y anchura 0).
Es decir, para el caso genérico en el que *I(t)≠0* la condición de contorno se establece definiendo *I(t)=δ(t)*. A partir de aquí, la ecuación diferencial (Ecuación 1) se transforma en homogénea por lo que para encontrar una solución es necesario definir un operador lineal integral *K*, en este caso se utiliza una función de Green definida en la Ecuación 7 (obsérvese que se obtiene a partir de la Ecuación 6).

$$G(t) = \begin{cases} 0, & t < 0 \\ e^{-\frac{t}{RC}}, & t \geq 0 \end{cases}$$

Ecuación 7

Sustituyendo la función *V* por la función *G* la ecuación diferencial (Ecuación 1) puede ser escrita como se muestra en la Ecuación 8.

$$C\frac{dG}{dt} + \frac{G}{RC} = \delta(t)$$

Ecuación 8

A partir de aquí y utilizando la función de Green, es posible obtener la solución tal como se muestra en la Ecuación 9

$$V(t) = K[I(t)] = \frac{1}{C}\int_0^t G(t-t')\cdot I(t')\cdot dt'$$

Ecuación 9

La solución mostrada en la Ecuación 9 es válida para el comportamiento sub-umbral partiendo de la premisa de que $V(0) = 0$. No obstante hay que tener en cuenta, en el ámbito de una implementación realista, la no linealidad en el instante en que el potencial de membrana supera el umbral de disparo. En la práctica consiste en inicializar las variables a un estado predefinido.
El estado predefinido puede no ser el inicial en el caso de incorporar un comportamiento refractario (periodo de tiempo inducido por la baja concentración de neurotransmisores e iones en proceso de despolarización). En este caso la dinámica neuronal se detiene durante cierto intervalo de tiempo T (periodo refractario) antes de reiniciar el proceso de integración con las condiciones iniciales del estado predefinido. Usualmente se asigna un valor U_r al potencial de membrana que puede ser negativo, aunque lo normal es que sea 0.
En general el periodo refractario establece un límite inferior en el periodo de disparo de la neurona, es decir el intervalo entre eventos o "*Interspike Interval"* se ve limitado como valor mínimo en un valor T [87] [76] [90] [23].

Como hemos visto, el periodo refractario es uno de los factores que limita la frecuencia máxima de disparo neuronal y dependiendo del tipo de neurona y sobre todo de las concentraciones de neurotransmisores. En la actualidad se cree que contribuye a la sincronización de trenes de pulsos y a la modulación temporal de la actividad global de la red.

2.3 Modelo de respuesta a eventos (SRM) *"Spike Response Model"*

El modelo de respuesta a eventos, SRM en adelante, es una generalización del modelo de integración y disparo en su versión no lineal (recordemos que este último era el modelo donde se incluía la discontinuidad cuando el potencial de membrana supera el valor umbral de disparo) sin embargo, es posible distinguir las siguientes diferencias:

- En el modelo de integración y disparo los parámetros son dependientes del voltaje de la célula mientras que en SRM dependen del intervalo de tiempo transcurrido desde el ins-

tante actual hasta el instante en que llegó el último evento. Ya que este influye en el potencial de membrana.

- Otra diferencia concierne a la formulación de las ecuaciones. Mientras el modelo de integración y disparo es definido en términos de ecuaciones diferenciales, el modelo SRM expresa directamente el potencial de membrana en el instante *t* como una integral de la historia pasada. Esto se hace así para incluir expresamente los impulsos de entrada.

Además de las diferencias mencionadas, el modelo SRM presenta la característica de poder modelar directamente el estado refractario, observado en las neuronas biológicas, como:

1. Baja sensibilidad en las entradas sinápticas después del potencial de activación
2. Un incremento en el umbral de disparo justo después del disparo neuronal.

En general en el modelo SRM el estado de una neurona *i* es descrito con una variable U_i representando el potencial de membrana. En ausencia de estímulos de entrada el valor de U_i será igual al valor del potencial de descanso U_r siendo generalmente igual a cero. Cada estímulo pre-sináptico alterará U_i incrementando o decrementándo su valor hasta que la neurona produzca un potencial de activación o bien, vuelva al estado inicial por ausencia de eventos pre-sinápticos (decaimiento pasivo).

$$U(t) = A + B + C$$

Ecuación 10

$$A = \eta(t - t_d)$$

Ecuación 11

$$B = +\sum_{j=1}^{n} W_j \cdot \sum_{j=1}^{n} \in_j (t - t_d, t - t_k)$$

Ecuación 12

$$C = \int_0^{\infty} K(t - t_d, t - t_k) \cdot I^{ext}(t_k))$$

Ecuación 13

La Ecuación 10 y sus componentes (Ecuación 11 hasta Ecuación 13), describe el comportamiento del potencial de membrana U para un instante cualquiera t. Obsérvese que la Ecuación 10 está compuesta por la suma de tres componentes A, B y C. Estos tres componentes dependen, de forma individual, de una función característica. Estas funciones características se les denomina funciones "*kernels*", las cuales describen el efecto de emisión y recepción de los eventos sobre la variable U_i.

El componente A, definido en la Ecuación 11, representa la forma del potencial de acción en el instante $(t-t_d)$, donde t_d es el instante de la última vez que la neurona disparó. La contribución de la función η se puede ver, de forma gráfica, como si ésta fuese copiada cada vez que el potencial de membrana alcanza el umbral de disparo.

La expresión B, definida en la Ecuación 12 está compuesta por: W_j que representa la eficiencia sináptica o "peso" de cada una de las sinapsis desde $j=1$ hasta la sinapsis n. La función ϵ modela el potencial post-sináptico excitativo o inhibitorio (de la neurona fuente hacia la neurona objetivo). $(t-t_k)$ representa el tiempo de duración de un potencial post-sináptico producido por el disparo de una neurona pre-sinaptica en el instante t_k hasta el actual t.

El componente C, definido en la Ecuación 13, representa la respuesta lineal del potencial de membrana debida a una entrada de corriente. La función describe cómo cambia el potencial de membrana desde el estado de reposo cuando llega un único evento. En otras palabras, es la respuesta impulso que se suma en cada instante que llega un evento. Obsérvese que el primer argumento de la función K es $(t-t_d)$. Esto indica que la función K tiene en cuenta el instante en que llega un evento pre-sináptico. De este modo se puede ajustar la relevancia del evento cuando la neurona se encuentra en periodo refractario. Recordemos que durante el periodo refractario, los canales iónicos permanecen abiertos lo que implica una reducción de la resistencia de la membrana y por tanto el cambio en el voltaje frente a la llegada de un pulso, es menos significativo.
El segundo argumento representa la corriente, procedente de los eventos pre-sinápticos, inyectada en el instante t_k. Recordemos que t_k representa el instante en que llegó el evento pre-sináptico.

A modo de resumen la tres ecuaciones, A, B y C, modelan el comportamiento del potencial de membrana U después de disparar (función A). La forma del potencial de acción de los eventos presinápticos (función ϵ) y el comportamiento del potencial de membrana con la llegada de los eventos teniendo en cuenta si se está o no en tiempo refractario (función C). Obsérvese que este tiempo refractario está directamente relacionado con la definición de las funciones ϵ y K.

La elección de las funciones "*kernels*" depende del comportamiento que se desee simular y que a su vez es dependiente del experimento. A modo de ejemplo en [134] se describe el comportamiento de una neurona motora donde se definen las funciones "*kernels*" en base a funciones exponenciales.

2.4 Aprendizaje y leyes de aprendizaje neuronales

2.4.1 Introducción

En el apartado 2.2 se habló de potenciales de acción. Es decir, del pulso eléctrico que viaja a través de los axones neuronales y que sirve como mecanismo de comunicación entre las células nerviosas. Sin embargo, como veremos, el potencial de acción y más concretamente la sincronización entre su llegada a la célula y la generación de un nuevo evento, también proporciona el mecanismo de modificación del peso sináptico.
Es importante hacer notar que el potencial de acción desencadena procesos químicos al final del axón, justamente en la sinapsis neuronal, constituyendo un mecanismo de transmisión de señales eléctricas a químicas. Estos procesos electroquímicos producen incrementos o decrementos en el potencial de la membrana celular de destino, siendo proporcionales a su permeabilidad (entiéndase por "la dificultad" al flujo de iones a través de los canales receptores localizados en las sinapsis). Este flujo de iones es dependiente de las concentraciones de neurotransmisores que definen el peso de la sinapsis. Es importante hacer notar que el que una sinapsis sea excitativa o inhibitoria depende del tipo o tipos iones que se canalizan

en los flujos post-sinápticos, que a su vez es función del tipo neurotransmisores que intervienen en la sinapsis.

La forma en que las señales sinápticas modifican la fuerza o peso de las sinapsis constituyen la ley de aprendizaje. Un modelo ampliamente utilizado en simulación para células biológicas es el definido por Hebb en [43]. Básicamente Hebb postula que si en un instante dado un estímulo pre-sináptico coincide en tiempo o produce un estímulo pos-sináptico, el peso de la conexión pre-sináptica es incrementado. En otras palabras, cuando el disparo de una célula activa otra, el peso de la conexión entre ambas tiende a reforzarse. La ley de Hebb es inherentemente inestable porque los pesos sinápticos son incrementados "por ellos mismos" [123] pero sin embargo funciona muy bien en la biología debido a la intervención de factores de estabilización como la degradación de la permeabilidad sináptica con el tiempo o la topología de las señales sensoriales de entrada a la red [123] [93] [94] [95].

Los cambios en la permeabilidad sináptica o fuerza sináptica pueden ser a corto o largo plazo:

Modificación a corto plazo (STD y STP): En la modificación a corto plazo no se producen cambios permanentes en las estructuras sinápticas neuronales, siendo la duración máxima de la potenciación o depresión de segundos o minutos. En este caso se habla de "potenciación a corto plazo" o STP "*Short Term Potentiation"* y de "depresión a corto plazo" o STD "*Short Term depresión"*. En el primero se produce un incremento de la relevancia sináptica, mientras que en el segundo, se produce un decremento.

Modificación a largo plazo (LTP y LTD): También es posible una modificación del peso sináptico permanente con una duración indefinida, producida por la activación continuada o repetida de la sinapsis. En este caso la activación reiterada induce la síntesis proteica en el núcleo de la neurona, alterando la estructura de la propia neurona induciendo cambios duraderos de mayor tiempo. A este tiempo de modificación en la relevancia sináptica se le denomina "Potenciación a largo plazo" o LTP "*Long Term Potentiation"* y "Depresión a largo plazo" o LTD "*Long Term Depresión"*.

Diversos estudios [131] [79] sugieren que LTP y LTD están involucrados en procesos relacionados con la memoria a largo plazo, aunque otros investigadores sugieren que este proceso se complementa con topologías neuronales específicas realimentadas que posibilitan el almacenamiento temporal y duradero de la información [13].

2.4.2 Ejemplo de aprendizaje Hebbiano

A modo de ejemplo se tiene el siguiente experimento:

Supongamos que tenemos un perro al cual se le somete a un flujo de aire en las pestañas (EI) *estímulo incondicionado*. De forma reactiva el perro responderá cerrando los ojos rápidamente (RI) *respuesta incondicionada*. Ahora suponemos que un intervalo de tiempo antes del flujo de aire en las pestañas se emite un sonido (EC) *estímulo condicionado*. Inicialmente el peso sináptico asociado al sonido es pequeño porque no existe correlación (véase la Ilustración 3). Tras un número razonable de estímulos en las pestañas, el peso de la conexión asociado al sonido se irá incrementando (recuérdese que según la ley de Hebb el peso se verá reforzado cuando el estímulo produzca un potencial de activación) de tal modo que finalmente el perro cerrará los ojos (respuesta condicionada) al oír el sonido e incluso en el momento justo del flujo de aire (véase la Ilustración 4 y la Ilustración 5).

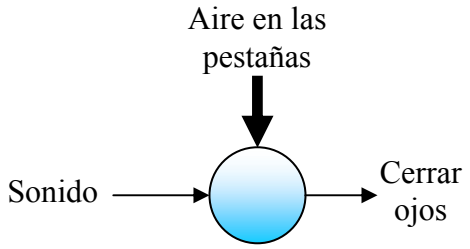


Ilustración 3: Estado de los pesos sinápticos antes del aprendizaje. El grosor de la flecha en las conexiones pre-sinápticas indica la relevancia del peso. La circunferencia ilustra una neurona.

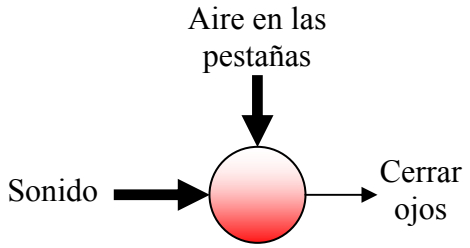


Ilustración 4: Estado de los pesos sinápticos después del aprendizaje. El grosor de la flecha en las conexiones pre-sinápticas indica la relevancia del peso. La circunferencia ilustra una neurona.

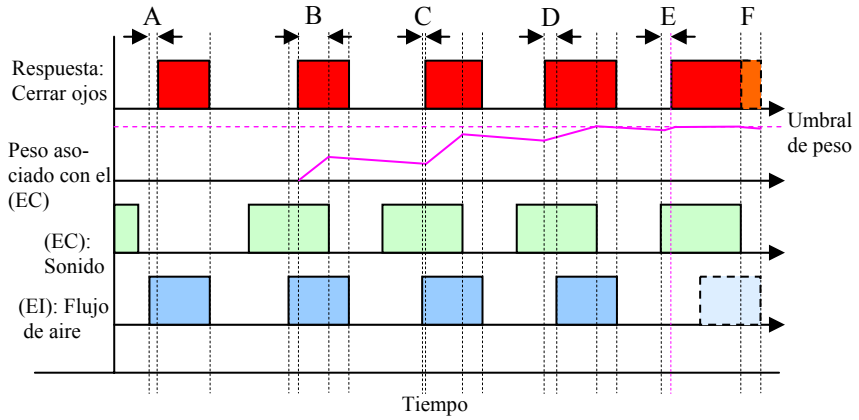


Ilustración 5: Proceso de aprendizaje siguiendo la ley de Hebb. Los rectángulos simbolizan eventos agrupados. El incremento de peso se produce cuando existe una relación temporal entre el (EI) y el (EC).

Obsérvese que para que el sistema funcione es precisa una premisa relacionada con el estímulo incondicionado (estímulo en las pestañas). Es decir, ha de existir un circuito neuronal preestablecido (sinapsis o conjunto de ellas) que relacione estímulo y respuesta (estímulo en las pestañas con cerrar los ojos). De igual modo, ocurre con el estímulo condicionado. Ha de existir algún tipo de relación (conexión neuronal) para que exista una asociación entre el sonido y la respuesta.

En este caso la evolución biológica soporta estas condiciones proporcionando estructuras para circuitos neuronales específicas que se observan iguales para todos los individuos. Estas estructuras han sido desarrolladas durante millones de años y están muy bien definidas y particularizadas en el cerebro. Pensemos por ejemplo en lo

que ocurre cuando una persona escucha un ruido ensordecedor inesperado. La respuesta instintiva incluirá, entre otros; dilatación repentina de las pupilas, cierre parcial de los ojos, tensión brusca de los músculos para buscar una posición de protección, incremento de la presión arterial etc. Luego ha de existir un conjunto de conexiones sinápticas en los centros nerviosos que entrelacen y asocien los circuitos neuronales involucrados en estas tareas.

La Ilustración 5 muestra el proceso de aprendizaje sináptico, el cual ha sido dividido en los siguientes seis intervalos (A…F):

- **Intervalo A:** El intervalo A, representa el tiempo de integración hasta el inicio del potencial de acción que producirá el cierre de los ojos. Obsérvese que la respuesta es reactiva ya que no se produce (EC) en el momento preciso y tampoco se produce modificación del peso en las sinapsis asociada.

- **Intervalos B:** El intervalo B, muestra el tiempo durante el cual se dan las condiciones de la ley de Hebb para el incremento del peso sináptico. Es decir, existe un (EC) y una correlación entre el (EC) y la generación de un potencial de acción.

- **Intervalo C:** Muestra cómo el incremento de peso en la sinapsis asociada con el (EC) produce una respuesta condicionada anticipada en el tiempo. Es importante observar la forma de pendiente decreciente que toma el peso durante el tiempo en que no se produce aprendizaje. Con esta pendiente se ha querido reflejar el decaimiento en el peso sináptico de la sinapsis asociado con la degradación en la eficiencia de los canales iónicos con el tiempo. Es decir, en este caso la sinapsis "olvida" de forma moderada. Esta característica, según algunos autores, es uno de los mecanismos de ayuda utilizados para estabilizar la inestabilidad inherente [123] del aprendizaje Hebiano.

- **Intervalo D:** Al igual que en el intervalo C, la respuesta condicionada se anticipa al (EI) y se produce un incremento en el peso sináptico asociado con el (EC). Obsérvese, que al

igual que ocurría en los intervalos anteriores, también se produce una leve degradación en el peso sináptico durante los periodos donde no aparece aprendizaje.

- **Intervalo E y F:** Aquí se observa el final del aprendizaje. El pesó sináptico asociado con el (EC) ha llegado a su máximo valor admisible de tal forma que la (RC) se produce antes y de forma anticipada al (EI). Se observa también, que ya no es necesario la existencia de un (EI) para producir una respuesta ya que las sinapsis asociada con el (EC) tiene la suficiente fuerza cómo para producir la misma respuesta que la sinapsis asociada con el (EI). Es importante destacar que el peso de la sinapsis codifica la respuesta temporal anticipada de la (RC). De igual modo, es importante observar cómo el (EI) pasa a un segundo plano pudiendo ser sustituido por el (EC) constituyendo un mecanismo de predicción más eficaz. A modo de ejemplo, supongamos que sustituimos la señal sonora por la señal procedente de un circuito neuronal encargado del reconocimiento de cierto patrón visual, de tal modo que se puede producir una respuesta anticipada para evitar una lesión ocular cuando se reconoce dicho patrón (por ejemplo un objeto cerca del ojo).

Es decir, a modo de hipótesis, la ley de Hebb proporciona un mecanismo de mejora en el aprendizaje partiendo de sistemas reactivos básicos. Extrapolando esta idea a sistemas neuronales mas complejos, como el caso del cerebelo, es posible la creación de nuevos modelos partiendo de modelos básicos que a su vez pudieron surgir de respuestas reactivas en las interacciones del individuo con el entorno o con su propia arquitectura corporal. Esta idea será ampliamente discutida en el apartado 5.2).

Hasta este momento se ha hablado del aprendizaje producido en la sinapsis asociada con el (EC) pero ¿que ocurre en la sinapsis asociada con el (EI)?. ¿Cómo se ha producido el aprendizaje reflejo o instintivo? No cabe duda que existen factores genéticos que predisponen a nivel celular ciertos “valores preconfigurados” como lo demuestran la existencia de los reflejos primarios (por ejemplo; agarrar un objeto cuando se estimula la palma de la mano de un recién nacido).

Otra posible respuesta involucra el estudio y la comprensión de las sensaciones a las que llamamos "dolor" o "placer" [21]. Podemos ver el dolor y el placer no sólo como un mecanismo de preservación física sino como un mecanismo global de aprendizaje de "alto nivel". En este ámbito entendemos el dolor, no sólo como una sensación producida por la destrucción de ciertos tejidos durante una lesión, sino como la evaluación negativa o no del resultado de una respuesta neuronal frente a un estímulo. Con esta idea en mente, dolor y placer se fusionan en una misma cosa como una medida de lo bueno o malo que ha resultado la respuesta dada frente a un conjunto de estímulos.

Volviendo al ejemplo, propuesto al inicio de esta sección, inicialmente cuando los pesos sinápticos de la sinapsis asociada con el (EI) son bajos, la respuesta incondicionada (cerrar los ojos) llega tarde. Poco a poco el peso sináptico se irá incrementando debido a la estimulación continuada de las pestañas que en algún momento provocará la coincidencia temporal entre el estímulo de entrada y la emisión de un potencial de acción. A una escala superior, abarcando circuitos neuronales más complejos, el dolor jugará un papel de aprendizaje cognitivo a más alto nivel. Por ejemplo hará que se recuerden que estímulos desencadenaron ese dolor, para evitarlos.

2.5 Conclusiones

- A lo largo de este capítulo se han analizado diferentes modelos para la simulación de neuronas biológicas basadas en pulsos, describiendo las relaciones funcionales entre el modelo y las células nerviosas reales. En este ámbito se han mostrado las características eléctricas así como se ha puesto de manifiesto la necesidad de utilizar modelos simplificados debido a la complejidad intrínseca de los procesos electroquímicos que tienen lugar.

- Se ha descrito el mecanismo de aprendizaje postulado por Hebb para las redes biológicas, incluyendo un ejemplo en detalle. Este ejemplo nos ha servido para ilustrar un par de hipótesis en las que se fundamenta esta tesis:

- El sistema nervioso central presenta una configuración topológica básica, muy eficiente y universal orientada a la interacción con el entorno que se modela y adapta mediante el aprendizaje.

- Es posible que los sistemas nerviosos partan de estados iniciales con configuraciones reactivas y que poco a poco, mediante el aprendizaje, adapten este estado inicial para conseguir nuevas configuraciones más beneficiosas.

Capítulo 3
Desarrollo de una plataforma híbrida software/hardware para simulación de redes neuronales en tiempo real

La simulación en tiempo real de redes neuronales a gran escala precisa de la utilización de sistemas de hardware específico debido, entre otros factores, al alto grado de paralelismo y la conectividad ínter-neuronal masiva inherente en las redes neuronales biológicas. En este capítulo se propone y desarrolla un prototipo (dirigido por tiempo) de arquitectura hardware/software de propósito específico para la simulación de redes neuronales masivas. Se estudian las ventajas y desventajas, el rendimiento del sistema comparado con los procesadores de propósito general disponibles en el mercado así como las posibles mejoras a tener en cuenta en futuras implementaciones.

3.1 Introducción

La construcción de un emulador neuronal con un esquema de computación eficiente, pasa por el análisis de las propiedades intrínsecas de los sistemas nerviosos. Estas propiedades pueden ser resumidas en:

- Las redes biológicas están compuestas por recursos masivos de computación paralela, organizadas con una alta densidad de conexiones topológicas. Estas conexiones se realizan en un espacio tridimensional, por lo que se densifica al máximo el volumen de interconexiones neuronales.

- La información que se intercambia entre los diferentes elementos de computación (neuronas) se codifica mediante trenes de pulsos.

- La frecuencia de disparo de las neuronas biológicas es baja (unos pocos centenares de Hertzios como máximo).

- El volumen de información de entrada y salida en las redes neuronales biológicas depende de la actividad global de la red (definida como el volumen de eventos de entrada/salida por unidad de tiempo). Como la frecuencia de disparo está limitada, la capacidad de procesamiento también es función del tamaño de la red.

Por otro lado la tecnología actual, tecnología de silicio, posee multitud de características ventajosas que pueden ser explotadas adoptando arquitecturas apropiadas de computación:

- En general, los circuitos digitales son capaces de trabajar a una elevada frecuencia de reloj (Mhz o Ghz). Esto permite utilizar circuitos secuenciales de alta velocidad para emular procesos biológicos paralelos de menor velocidad.

- El cerebro humano no alcanza una actividad instantánea mas allá de un 5-10% debido principalmente al consumo de energía [91]. Es decir, el consumo energético por unidad de

volumen en el cerebro es un factor físico limitante de su nivel de actividad [104] [20] [8]. Sin embargo, ese umbral limitante es superior en los circuitos electrónicos basados en silicio y por tanto, puede ser una ventaja en las prestaciones finales. A modo de comparativa destacar que un circuito VLSI de altas prestaciones permite consumos de unos 23W por cada 100 mm^2 de superficie, mientras que el consumo de todo el cerebro humano está en torno a 20W [20] [8].

- La densidad de integración de los circuitos VLSI actuales es muy elevada, inclusive mayor que la densidad de integración neuronal en el cerebro en determinadas áreas (entiéndase como la comparación entre número de neuronas y número de transistores por unidad de área). A modo de ejemplo; un milímetro cuadrado de corteza cerebral contiene unas 100.000 neuronas, mientras que en un milímetro cuadrado de un procesador Pentium 4 hay en torno a 4.2 millones de transistores. A pesar de ello, no olvidemos que una neurona es funcionalmente mucho más complejo que un transistor.

- El consumo medio de un cerebro humano es de aproximadamente 20W [20] [8] y está refrigerado por líquido (sangre). Sin embargo, es posible construir circuitos digitales con consumos energéticos superiores (si afirmamos que la velocidad del sistema va en proporción a su consumo energético y que su eficiencia energética sea al menos igual a la de un circuito neuronal biológico) pudiéndose evacuar dicho calor mediante sistemas de refrigeración más eficientes.

Como inconveniente, en los circuitos digitales la conectividad (a nivel de silicio) está limitada a dos dimensiones por lo que se consumen importantes recursos en el diseño de la topología. Por lo tanto, la alta densidad de las conexiones topológicas neuronales es muy difícil de implementar en chips con tecnologías VLSI.

Centrándonos en los procesos que ocurren a nivel de red, resulta interesante el estudio de los fenómenos de sincronización que ocurren en poblaciones de neuronas regidos por la dinámica temporal de las sinapsis [99] [98] [97]. En este ámbito, este capítulo se centra en el estudio de modelos de respuesta con pulsos (SRM) "*Spike-*

Response-Model" (este modelo fue ampliamente discutido en el apartado 2.3).

Otro aspecto importante es el estudio y modelado de sinapsis basadas en conductancias. En éste ámbito la corriente que se inyecta en el núcleo celular depende (entre otros; si existe o no inyección gradual, pesos sinápticos, etc.) de la resistencia de la sinapsis y por tanto de la diferencia de potencial entre el potencial de membrana y el pulso de entrada. Esta característica es muy biológica y aunque no está claro que tenga una especial relevancia en la funcionalidad de una red neuronal [32] [33], se pueden obtener ciertos comportamientos de autorregulación difíciles de obtener con un modelo lineal [41]. Además [90] muestra como las sinapsis basadas en conductancias permiten cambios en el potencial de membrana más rápidos que en modelos lineales (recordemos que la evolución del potencial de membrana tiene un comportamiento logarítmico en estos sistemas), lo que permite explicar la velocidad de respuesta cierto circuitos neuronales como por ejemplo el pre-procesamiento visual localizado en la retina humana.

El proceso estándar de integración neuronal es computacionalmente ineficiente, cuando se utilizan arquitecturas hardware convencionales (plataformas mono o multi-procesador) [1]. Algunos investigadores han orientado esta implementación hacia plataformas hardware de integración específicos [35] [54] [112] [120] [71]. Estos estudios implementan modelos de SRM propuestos por Eckhorn y otros en [98].

Teniendo en cuenta las características mencionadas, se ha desarrollado una arquitectura de cómputo orientada a la simulación de redes neuronales en hardware específico. Esta arquitectura es una primera aproximación para establecer claramente las limitaciones que aparecen en la construcción de simuladores de circuitos neuronales biológicos.

3.2 Sistemas dirigidos por tiempo y sistemas dirigidos por eventos

La bibliografía aporta dos esquemas de computación para la simulación de sistemas neuronales biológicos. Sistemas *dirigidos por tiempo* [25] [75] y sistemas *dirigidos por eventos* [7] [75]. Cada uno aporta ventajas e inconvenientes, pero en general las últimas tendencias aportan mayor relevancia hacia los esquemas *dirigidos por eventos* debido a que brindan la posibilidad de simulación de redes neuronales masivas en procesadores secuenciales.

3.2.1 Sistemas dirigidos por eventos

Hay diversos estudios de investigación que proponen o utilizan esquemas de computación dirigido por eventos para la simulación eficiente de redes neuronas de pulsos [18] [82] [109]. La computación en estos esquemas (integración de los pulsos pre-sinápticos y generación de pulsos post-sinápticos) ocurre únicamente cuando llega un pulso a una sinapsis en un determinado instante. De esta forma solamente se computan aquellas neuronas que han recibido o emitido algún evento. En este tipo de esquemas el tiempo de simulación entre ciclo y ciclo de computación es dinámico debido a que el paso de tiempo entre cada estado neuronal está regido por la diferencia temporal no constante entre los pulsos pre-sinápticos (pertenecientes o no a una misma neurona).
Esta característica los hace muy eficientes en simulaciones masivas de neuronas, debido al bajo índice de actividad que presentan los circuitos biológicos en relación al número de neuronas que los componen, pero requiere el establecimiento de políticas de cómputo para el cálculo de la historia pasada así como la predicción de disparo futuro.
Entiéndase como “historia pasada” el estado neuronal desde la llegada del último evento hasta el actual que se está procesando. De igual modo, es preciso conocer si la neurona disparará en el intervalo de tiempo que transcurrirá desde el instante actual hasta el siguiente tiempo de evaluación (definido por otro evento). A esto se le ha descrito como “predicción de disparo futuro”.

Estas características imponen una fuerte restricción en el tipo de neuronas que se pueden simular sin que ello altere la eficiencia que proporcionan los esquemas dirigidos por eventos. Esta restricción ocurre cuando la constante de tiempo neuronal (velocidad con que cambia el potencial de membrana) es suficientemente pequeña como para ser inferior al paso de tiempo entre dos eventos con resolución temporal máxima. Además, la predicción ha de ser precisa para simular de sincronización neuronal, codificación de poblaciones neuronales, etc.

Existen otras características que son difíciles de simular con esquemas dirigidos por eventos. En este ámbito la inclusión de la "inyección gradual de carga" es una de ellas, aunque recientes estudios han incluido aproximaciones basadas en tablas [109]. Donde a pesar de un elevado coste en recursos de memoria, se consiguen resultados muy interesantes.

3.2.2 Esquemas dirigidos por tiempo

Existe otra estrategia de procesamiento dirigida por tiempo. En este caso, en cada paso de tiempo fijo (periodo de muestreo de simulación) se actualizan los estados de todas las neuronas de la red, exista o no actividad en ellas. Al tener que actualizar todas las neuronas de una red esta estrategia requiere importantes requisitos de cómputo en tiempo real para redes masivas. Estos requisitos son difícilmente alcanzables con máquinas de procesamiento secuencial a pesar de la utilización de circuitos con relojes de alta velocidad.
A pesar de esta dificultad la estrategia dirigida por tiempo brinda una primera aproximación para la construcción de simuladores neuronales eficientes, dado que en un caso extremo de una simulación dirigida por eventos con un nivel de actividad muy elevado, el rendimiento decrecerá hasta ser similar a una simulación dirigida por tiempo. El esquema de simulación dirigido por tiempo tiene la ventaja de que la velocidad de cómputo es independiente del nivel de actividad de la red y además la interacción con sensores y motores es inherentemente dirigida por tiempo. Por ello, es una aproximación interesante para sistemas empotrados de tiempo real.

La estrategia dirigida por tiempo se estudia ampliamente en este capítulo mientras que la aproximación dirigida por eventos se analiza en profundidad en el Capítulo 4.

3.3 Caracterización del sistema de procesamiento neuronal

El sistema se ha implementado siguiendo una filosofía de co-diseño hardware/software que explota eficientemente las ventajas de cada uno. En este ámbito, la parte hardware está formada por una tarjeta PCI de prototipado estándar de hardware reconfigurable RC1000 desarrollada por Celoxica [12]. Esta tarjeta funciona como coprocesador y computa los estados internos neuronales. Mientras que la parte software realiza el mantenimiento de: la topología de la red, el aprendizaje y la gestión de comunicaciones PC/tarjeta PCI.
El sistema está orientado para la simulación de modelos neuronales de pulsos (SRM) (recordemos que este modelo fue descrito en el apartado 2.3) con sinapsis modeladas como conductancias.

La dinámica temporal del proceso de integración sináptica es modelada mediante inyección gradual de carga. Esto significa que la carga no se inyecta instantáneamente en el núcleo neuronal sino que se inyecta siguiendo una función temporal (esta función es descrita más adelante en la sección 3.5). Esta característica implica un consumo computacional importante en esquemas de procesamiento secuencial pero se puede solucionar utilizando hardware dedicado. En el hardware dedicado los diferentes estados neuronales son procesados en paralelo mediante unidades de cómputo replicadas que se sincronizan para dar un resultado temporal coherente.
Las unidades de cómputo replicadas (paralelas) utilizan una arquitectura en cauce segmentado *"pipeline"* compuesta por cinco etapas lo que permite la posibilidad de poder añadir nuevas características neuronales (tales como canales NMDA, mecanismos de disparo alternativos o complementarios, propiedades intrínsecas resonantes [22], etc.) añadiendo nuevas etapas de computación en el camino de datos. De este modo, el tiempo total de computación no se ve incrementado (siempre que la nueva etapa no se convierta en la etapa más lenta del cauce). Sólo se alarga la latencia que no es un factor

determinante ya que las conexiones sinápticas tienen latencias del orden de milisegundos.

La paralelización de los esquemas de procesamiento dirigido por tiempo es posible debido a las latencias generadas en las interconexiones sinápticas. Esto permite la evaluación de diferentes neuronas al mismo tiempo en un número restringido de unidades de cómputo. Para ello, se utilizan intervalos de tiempo dedicados. A esta técnica se le denomina multiplexación temporal y consiste en asignar un intervalo de tiempo de uso de las unidades de computación, a cada neurona. Siguiendo esta técnica, el rendimiento del sistema depende directamente del número de unidades de procesamiento disponibles así como de la velocidad de dichas unidades. Es decir, en un esquema dirigido por tiempo es fácilmente paralelizable utilizando varias unidades funcionales.

En las siguientes secciones se define la división entre los componentes hardware y software, una descripción detallada de la implementación hardware del modelo neuronal y su paralelización, la utilización de recursos hardware (registros, bancos de memoria etc) y un estudio acerca del rendimiento del sistema. En éste último caso, el rendimiento del sistema se ha medido mediante una implementación para tiempo real de un modelo del cerebelo aplicado a una tarea sencilla de seguimiento de trayectorias, utilizando un brazo robot de dos grados de libertad.

3.4 Esquema de computación

Como se mencionó en el apartado 3.2, la plataforma de computación tiene un esquema hibrido hardware-software. El componente hardware consiste en un sistema acelerador que proporciona recursos de cómputo (dispositivo FPGA trabajando como neuroprocesador) y memoria.
El componente software se ejecuta en el PC y la comunicación se realiza a través del bus estándar PCI. En esta versión del sistema, la función del hardware está restringida a la computación de los estados neuronales (procesamiento más costoso). El componente software es responsable de mantener la conectividad de la red (topología), el envío de los pulsos a las neuronas y el aprendizaje.

La comunicación entre los componentes hardware y el software está restringida únicamente a los eventos; el componente software envía paquetes de eventos formados por las direcciones pre-sinápticas destino al componente hardware. A su vez el componente software recibe los paquetes post-sinápticos del ciclo de cómputo previo (véase la Ilustración 6).

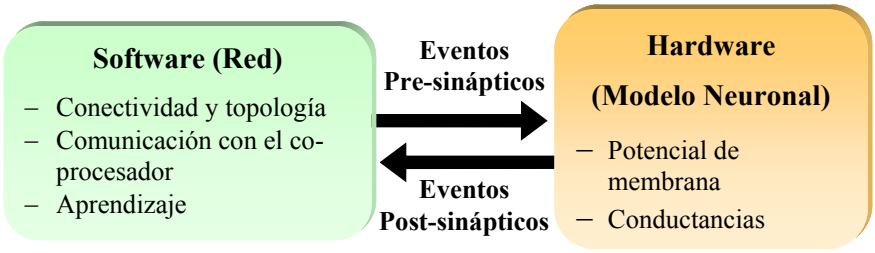


Ilustración 6: Arquitectura hardware/software para simulación dirigida por tiempo. El software gestiona la topología de la red así como el aprendizaje. El hardware computa los potenciales de membrana y las conductancias del modelo neuronal. Para comunicar ambos componentes (hardware y software) se utiliza un esquema de comunicación dirigido mediante paquetes de pulsos enviados regularmente.

Los estados neuronales se almacenan en la tarjeta coprocesador ya que la parte software no necesita esta información. De este modo, la parte software genera eventos pre-sinápticos y la parte hardware genera eventos post-sinápticos.

Eventos pre-sinápticos: Cada evento pre-sináptico es una estructura de datos formada por; un identificador (dirección de la neurona destino, un peso inhibitorio más un peso excitativo y, por último, el instante o tiempo en la simulación en que se ha producido.

Eventos post-sinápticos: Cada evento post-sináptico es una estructura de datos formada por un identificador (dirección de la neurona origen) y el instante en que se produjo.

En esta arquitectura, cada neurona puede tener dos tipos de sinapsis; una es excitativa y otra inhibitoria. Aunque con pequeñas modificaciones es posible incluir mayor variedad permitiendo simular conexiones AMPA, NMDA, GABA o GABAb. Es decir, modelos sinápticos con distintas características temporales.

La comunicación entre el PC y el hardware se hace a través del bus PCI del PC. El establecimiento de la comunicación así como su finalización requieren de un coste fijo de tiempo independiente del volumen de datos que se van a transferir. Para reducir al máximo este retardo constante en cada transferencia, la comunicación se establece cada cierto intervalo de tiempo o época.
En cada época se transmiten y reciben los paquetes de eventos (transmisión de eventos pre-sinápticos para la siguiente simulación y recepción de los eventos post-sinápticos de la anterior) formados por 20 pasos de tiempo de 100μs. En cada época de procesamiento se actualizan los estados neuronales de todas las neuronas que están siendo simuladas.

Durante la transferencia de eventos entre el hardware y el PC existe una sincronización temporal mutua. Mientras el coprocesador calcula los eventos de salida, el software prepara los eventos pre-sinápticos para la siguiente época de computación. En ocasiones es posible que el coprocesador termine antes que la parte software y tenga que iniciar ciclos de espera. Para minimizar este tiempo e intentar optimizar al máximo el rendimiento global, la parte software se implementa con hebras y memoria compartida. Se utiliza una hebra para calcular la topología de la red según los eventos pre-sinápticos y post-sinápticos. Otra para la comunicación con el coprocesador hardware (transferencia de paquetes) y por último una para el aprendizaje (cálculo de pesos).

Para organizar la información procesada por las hebras, se ha utilizado una variante de la técnica de sincronización (de grano grueso) por barreras (véase la Ilustración 7A) para modelos de programación paralelos BSP "*Bulk Synchronous Parallel Computation"* [49]. La librería de funciones Oxford BSP_lib describe y facilita esta tarea proporcionando una alternativa a MPI "*messaging passing interface"* y a PVM "*parallel virtual machina*".

Concurrentemente con la computación y la comunicación, el componente software introduce una latencia en el cálculo de la ruta de un evento entre dos neuronas (véase la Ilustración 7 B). Esta latencia tienen que ser inferior al mínimo retardo sináptico entre dos neuronas por lo que supone una limitación en simulaciones que se pueden llevar a cabo donde la resolución temporal es crítica. En este

ámbito la velocidad de comunicación a través del bus PCI es un factor determinante fijando el volumen máximo de eventos que se pueden transmitir en tiempo real.
Debido a la posibilidad de que exista más de un evento pre-sináptico que tenga como objetivo la misma neurona, una de las estrategias adoptadas para minimizar tiempos de espera, consiste en enviar al hardware los eventos ordenados siguiendo su marca temporal. De este modo el hardware procesa la información directamente sin tener que realizar ningún tipo de ordenación previa.

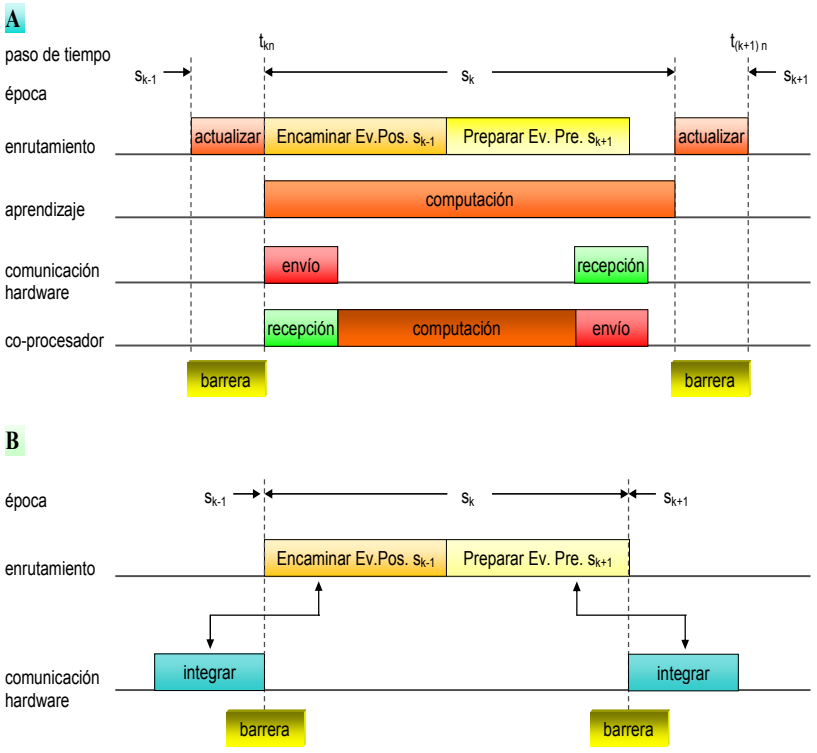


Ilustración 7: Paralelización dentro del componente software. **(A)** Implementación multi-hebra basado en el algoritmo BSP. Parte superior 3 hebras; topología, aprendizaje y comunicaciones con el coprocesador. En la parte inferior, la línea temporal representa el tiempo consumido por el coprocesador. La comunicación entre hebras se realiza en épocas s_k (consistiendo en n pasos de tiempo) y ocurre en la barrera donde el dato compartido de cada hebra es copiado. La barrera representa una sincronización explicita que ocurre en cada época. La comunicación y la computación se superponen dentro del sistema global, pero es secuencial dentro del co-procesador. **(B)** La comunicación y computación introduce una latencia en el enrutamiento de los eventos post-sinápticos (desde la época s_{k-1}) y los eventos pre-sinápticos (hasta la época s_{k+1}). Esta latencia es igual al periodo de duración de dos épocas y debería ser inferior al mínimo retardo sináptico.

La comunicación y la computación con el hardware son secuenciales realizándose a través de los bancos de memoria de la tarjeta (Véase la Ilustración 8). El dispositivo FPGA lee los eventos pre-sinápticos desde los bancos de memoria donde ha escrito la parte software y una vez calculados, escribe los eventos post-sinápticos. Este tipo de esquema no restringe el tipo de aprendizaje sináptico que puede ser implementado.

De forma paralela se ha desarrollado un emulador en software para la parte hardware del sistema. Con este emulador es posible probar la funcionalidad del sistema en un PC que no dispone de la tarjeta aceleradora. Este componente emula el hardware a bajo nivel y utiliza las mismas primitivas de comunicación que con la parte hardware. Con esta herramienta es posible validar la aritmética en punto fijo que utiliza el componente hardware ya que durante el diseño es crítica la elección del ancho de bits para las diferentes variables o registros del sistema. De este modo se pueden realizar estudios de precisión comparando los resultados obtenidos con el simulador en punto flotante con los obtenidos por el hardware en aritmética en punto fijo.

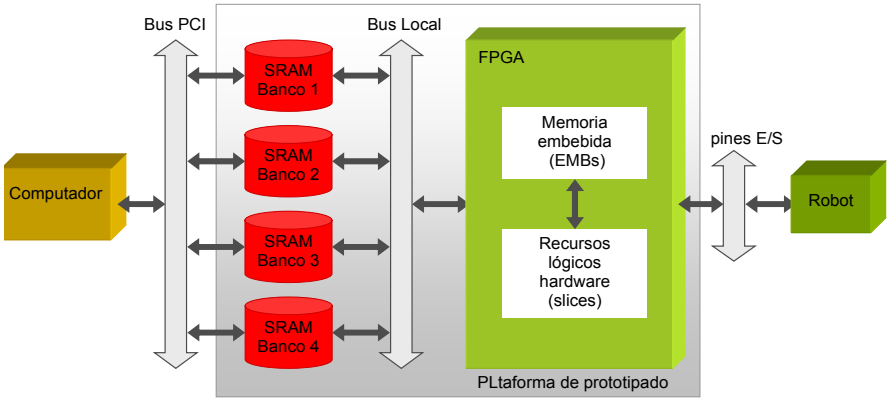


Ilustración 8: Esquema de comunicación hardware/software. Se utilizan bancos de memoria SRAM como interfaz entre el software y el hardware. Todas las variables neuronales (potenciales de membrana y conductancias) son permanentemente almacenadas en los bancos de memoria de la plataforma. Los eventos pre-sinápticos y los eventos post-sinápticos son almacenados en los bancos de memoria tanto por el componente software como por el componente hardware. Se utiliza memoria embebida interna al chip como buffer para la gestión de las unidades paralelas dentro del chip FPGA.

3.5 Modelo Neuronal

El modelo neuronal está formado por un elemento para el cálculo del potencial de membrana junto con un mecanismo de generación de eventos post-sinápticos [37].

$$V_x^t = V_x^{t-1} - \frac{1}{\tau_r}\left(V_x^{t-1} - V_{rest}\right) + G^{t-1}\cdot S_{exc}^{t-1}\cdot(U_{max} - V_x^{t-1}) - G^{t-1}\cdot S_{inh}^{t-1}\cdot\left(V_x^{t-1} - U_{min}\right)$$

Ecuación 14: Cálculo del potencial de membrana. *t-1* representa el instante anterior del paso de simulación.

Para las neuronas con sinapsis excitativas e inhibitorias el potencial de membrana Vx es calculado siguiendo la Ecuación 14 donde: τ_r es la constante de tiempo del termino de reposo. S_{exc} y S_{inh} son las contribuciones sinápticas y G es el término de ganancia modelado como término determinante del periodo refractario. U_{max} y U_{min} son el máximo y el mínimo valor del potencial de membrana y $V_{resting}$ es el potencial de membrana de reposo (potencial de membrana después de un intervalo sin recibir actividad).

Funciones exponenciales	Funciones aproximadas
$S_{exc}(t) = \begin{cases} 0 & , \ t < t_0 \\ S_{exc}\cdot e^{-(t-t_0)/\tau_{exc}} & , \ t \geq t_0 \end{cases}$	$S_{exc}(t) = A_{exc}\cdot w_{exc} + B_{exc}\cdot S_{exc}(t-1)$
$S_{inh}(t) = \begin{cases} 0 & , \ t < t_0 \\ S_{inh}\cdot e^{-(t-t_0)/\tau_{inh}} & , \ t \geq t_0 \end{cases}$	$S_{inh}(t) = A_{inh}\cdot w_{inh} + B_{inh}\cdot S_{inh}(t-1)$
$G(t) = \begin{cases} 0 & , \\ G\cdot e^{-(t-t_0)/\tau_{refrac}} & , \end{cases}$	$G(t) = 1 - g(t)$ $g(t) = A_{refrac}\cdot Salida + B_{refrac}\cdot g(t-1)$

Tabla 1: Filtros IIR definidos mediante funciones exponenciales conjuntamente con sus aproximaciones. El término "Salida" es 1 cuando la neurona dispara ó 0 en caso contrario.

Los términos dinámicos S_{exc}, S_{inh} y G son modelados mediante los filtros temporales IIR con las constantes de tiempo τ_{exc}, τ_{inh} y $\tau_{refract.}$ Éstos son definidos mediante funciones exponenciales decrecientes (véase la columna izquierda de la Tabla 1) y son implementados utilizando aproximaciones funcionales (véase la columna derecha de la Tabla 1) donde t_0 es el instante en el que se recibió un pulso por la sinapsis implicada (S_{exc} y S_{inh}) y para $G(t)$ el instante en que se produce el disparo neuronal.

El modelo así definido, es equivalente a un modelo de integración y disparo con "pérdida" donde las sinapsis se modelan como conductancias con eficacia sináptica que depende del tiempo trascurrido desde que se recibió el evento pre-sináptico (este modelo emula la inyección gradual de carga).

La neurona (véase la Ilustración 9) es implementada mediante una unidad de procesamiento con los siguientes componentes:

- Dos filtros de respuesta impulso infinita (IIR) que emulan el acoplamiento sináptico no instantáneo con diferentes constantes (véase Tabla 1 e Ilustración 9) para las entradas inhibitorias (GABAergic) y sinapsis excitativas (AMPA).

- Un registro de integración que almacena el potencial post-sináptico que es computado en cada ciclo de tiempo usando la Ecuación 14.

- Un filtro de salida (IIR) con constantes de tiempo que controlan el periodo refractario post-sináptico a través del término de ganancia (G).

- Un mecanismo de generación de eventos de acuerdo con el potencial de membrana. Esta aproximación incluye un único umbral de disparo.

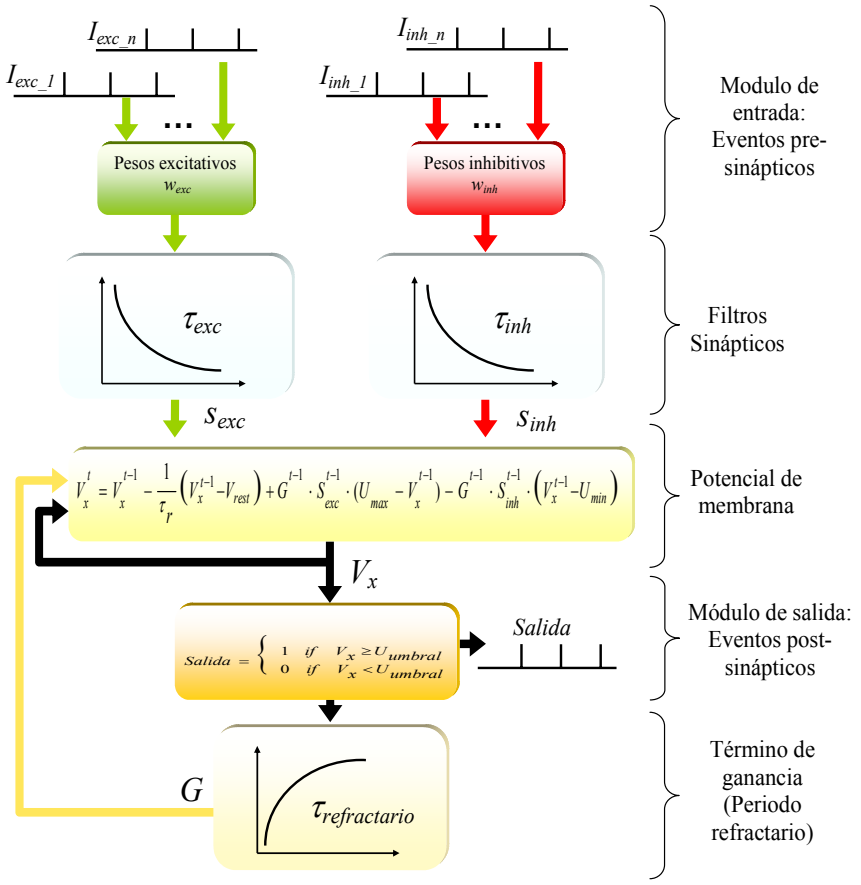


Ilustración 9: Esquema de la unidad de procesamiento neuronal. Obsérvese cómo puede ser dividido en etapas para adoptar un esquema de procesamiento con segmentación de cauce.

3.6 Estrategias de computación paralela

La implementación secuencial del modelo neuronal requiere 17 ciclos de tiempo para toda la computación. Esto se consigue adoptando las siguientes estrategias de diseño para explotar eficientemente los recursos del dispositivo FPGA.

A. Computación con segmentación de cauce

Utilizando la técnica de segmentación de cauce se consigue reducir el número de pasos de tiempos necesarios para realizar opera-

ciones complejas. De acuerdo con la explotación del paralelismo inherente de los dispositivos FPGA, se ha diseñado una estructura de computación con un camino de datos segmentado con $N_{etapas} = 5$ etapas (S_1 hasta S_5, véase la Ilustración 10).

Estas etapas son:

1. **Captación:** Recupera los estados neuronales desde una tabla almacenada en EMBs. También captura el evento de la tabla de eventos. En el caso de que no exista un evento de entrada en la neurona se utilizará un evento vacío.

2. **Computación neuronal I:** Calcula la contribución sináptica utilizando los filtros IIR (S_{exc} y S_{inh}).

3. **Computación neuronal II:** Calcula el potencial de membrana (V_x).

4. **Computación neuronal III:** Se calcula el término de reposo, el término del periodo refractario (G) y el evento post-sináptico (en caso necesario).

5. **Estado de actualización:** Almacena los eventos de salida y los estados neuronales ya calculados en la tabla correspondiente.

Es importante destacar la posibilidad de incluir fácilmente en la computación en segmentación de cauce otras características del modelo tales como canales NMDA, umbrales de disparo modificables dinámicamente etc., sin que ello suponga una degradación en la eficiencia del sistema (siempre que las nuevas etapas añadidas no tengan tiempos de computación superiores a la etapa más costosa).

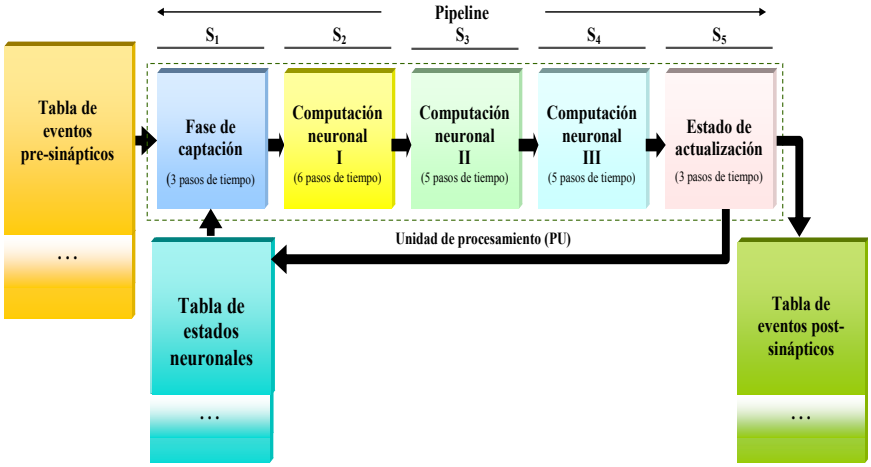


Ilustración 10: Se han definido 5 etapas en el camino de datos. El número de ciclos de cada etapa se incluye entre parénTesis en la figura indicando qué etapa limita la velocidad del conjunto.

B. Escalabilidad

La plataforma de computación utiliza gran cantidad de unidades de procesamiento local (UP) empotradas en la FPGA funcionando en paralelo. Se ha adoptado un esquema de computación escalable mediante la utilización de recursos de memoria embebida dividida en bloques personalizados y asignados a las diferentes unidades de cómputo. En esta línea todas las unidades de procesamiento pueden acceder a su entrada al mismo tiempo y escribir su salida en paralelo sin conflictos, por dependencia de recursos (véase la Ilustración 11).

Además de las N_{up} unidades de procesamiento existe otro circuito trabajando en paralelo con ellas. Este circuito es el (PEG) *captador de eventos sinapticos* (véase la Ilustración 11) y sirve como unidad de pre-captación de lectura de los bancos de memoria.

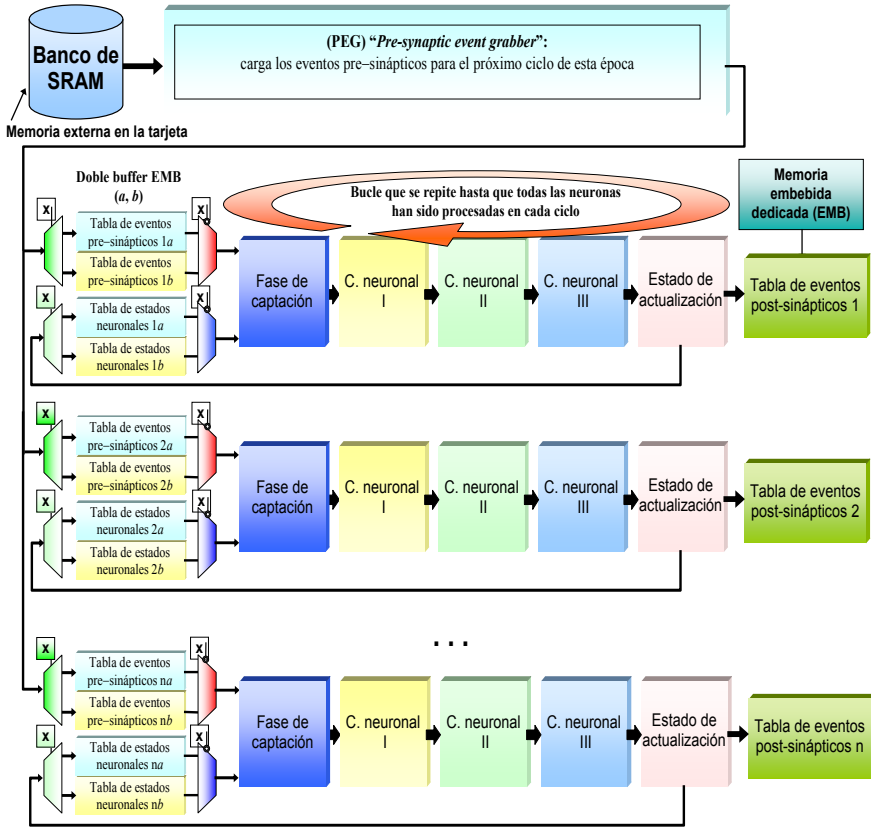


Ilustración 11: Paralelismo a nivel de unidad de procesamiento; El PEG (*captador de eventos sinápticos*) y las unidades de procesamiento funcionan en paralelo. Los ciclos consumidos por el PEG dependen del número de eventos pre y post-sinápticos mientras que los ciclos consumidos por las unidades de procesamiento dependen del número de neuronas simuladas (véase Ilustración 16). Mientras la unidad PEG está usando los buffer de entrada *a*, las unidades de procesamiento utilizan los buffer *b*.

La unidad (PEG) trabaja dentro del ciclo global de computación definido de la siguiente forma. En cada paso o ciclo de simulación, la unidad de procesamiento computa un nuevo estado neuronal como consecuencia de la llegada de un evento pre-sináptico. De este modo los estados neuronales son actualizados. Al mismo tiempo, la unidad PEG, lee el evento para el siguiente ciclo desde los bancos de memoria (SRAM) externos al chip FPGA y lo escribe en los bancos de memoria embebida preparando su procesamiento en los siguientes ciclos de computación.

Para garantizar la escalabilidad de la unidad de procesamiento, cada una de estas unidades tiene un banco de EMBs dedicado. De este modo las unidades de procesamiento pueden acceder en paralelo a sus datos de entrada. El acceso a los bancos, se realiza mediante una arquitectura de doble buffer (llamado *a* y *b* en la Ilustración 11). Mientras la unidad PEG escribe un evento pre-sináptico para el siguiente ciclo de simulación en *a*, la unidad de procesamiento utiliza el evento almacenado en *b* y escribe el nuevo estado neuronal en *a*. (obsérvese que existen dos buffers *a* y dos buffer *b* tanto para eventos como para estados neuronales). En el siguiente ciclo de simulación los registros *a* y *b* intercambian su funcionalidad. Este proceso se repite hasta la finalización de la época o ciclo de simulación temporal.

El esquema de comunicación implementado, permite distribuir los motores de simulación en diferentes chips FPGA adecuando el direccionamiento dentro del protocolo de simulación. Este esquema multi-chip permite aumentar las posibilidades de escalabilidad de la plataforma más allá de las capacidades de un único dispositivo de cómputo. Otras investigaciones, como [100] [126], están actualmente explorando esquemas similares.

3.7 Estudio de consumo de recursos de computación

La carga computacional (ecuaciones del modelo neuronal descritas en la sección 3.4) ha sido distribuida en diferentes etapas del cauce para permitir el procesamiento paralelo a lo largo del camino de datos (los recursos computacionales de cada una de estas etapas se resumen en la Tabla 2).

La plataforma utilizada para esta implementación, tal como se mencionó en el apartado 3.3, ha sido la RC1000 del fabricante Celóxica [12]. Ésta es una tarjeta PCI con 4 bancos de memoria SRAM externa (2MB cada uno) y posee una FPGA del fabricante XIlinx con 2.5 millones de puertas lógicas (Virtex- 2000E) [11].

Los recursos de utilización del dispositivo FPGA, siguiendo la Ilustración 12, han sido calculados mediante la realización de compilaciones parciales. Cuando el sistema es compilado en conjunto (sistema completo) la asignación de recursos lógicos es optimizada de forma automática por la herramienta de diseño. Es decir, es posible que se compartan circuitos entre las distintas unidades funcionales que no trabajen al mismo tiempo. Es importante hacer notar que el volumen de neuras, el número de sinapsis por neurona y el número de eventos están limitados por la memoria disponible en el PC y el tamaño de los bancos de memoria de la tarjeta RC2000, donde se realiza el trasvase de información. No obstante existe una limitación física en el número de eventos que pueden ser procesados en tiempo real que será analizada ampliamente en el apartado 3.8.2.

Nº boques de memoria embebida	% bloques de memoria embebida	Nº Slices	% Slices	Nº Total de puertas lógicas equivalentes
80	50	4581	23.86	1389247

Tabla 2: Recursos de computación de la unidad completa con segmentación de cauce: La cantidad total de recursos utilizados y porcentaje de ocupación del dispositivo Xilinx XCV2000-E. Suponiendo que incluye solamente una unidad de procesamiento neuronal y recursos de memoria embebida (160 bloques de 4Kb cada uno). Esta unidad de procesamiento con segmentación de cauce es capaz de computar 6 neuronas en paralelo.

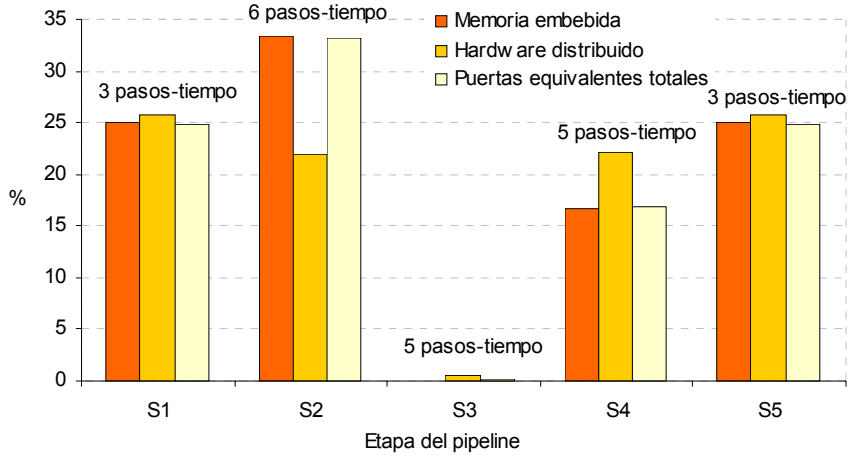


Ilustración 12: Recursos de computación de cada etapa del cauce, cada barra representa el porcentaje de recursos de una única unidad de procesamiento

La Tabla 2 muestra el consumo de recursos de una unidad de procesamiento completa. Esta unidad consume menos del 25% de los recursos lógicos del dispositivo, por tanto, es posible integrar un máximo de 4 unidades de computación en un único chip trabajando en paralelo tal como se indica en la Ilustración 13 y en la Ilustración 14. La Ilustración 12 muestra cómo los estados S_1 y S_5 consumen recursos similares debido a que son los encargados de la captura y el envío de los datos que son introducidos en el camino de datos.

La etapa S_2 es la etapa que consume mayor número de recursos debido a que ésta incluye los filtros IIR (*Respuesta impulso infinita*) computados en paralelo. Las multiplicaciones requieren una alta precisión (14 bits para los términos de entrada) para mantener el filtro estable. Para ello se utilizan dos multiplicadores en punto fijo de 14 bits cada uno. Además, necesita almacenar los estados temporales de los filtros (dos por neurona) los cuales requieren recursos de memoria significativos.

La etapa S_3 utiliza las variables de estado de las otras etapas (reutilización de recursos de memoria) y requiere únicamente un sumador.

La etapa S_4 incluye la generación de los eventos de salida, el periodo refractario, el término de ganancia y el término de reposo (el cual está implementado con un registro de desplazamiento). S_4 necesita la mitad de los recursos de memoria de S_2 debido a que es la única etapa donde es necesario almacenar el estado refractario de cada neurona que se está computando.

El número de pasos de tiempo consumidos en cada etapa del cauce se muestra en la Ilustración 10. De este modo es posible conocer qué estados del cauce restringen la velocidad efectiva de la computación. Debido a que todos los estados trabajan en paralelo de forma sincronizada, la eficiencia del sistema (*datos por segundo*) está limitada por la etapa más larga (S_2 en este caso). Esta etapa consume NTS_{lps} = 6 pasos de tiempo (véase la Ilustración 10). Como cada unidad de procesamiento produce una actualización neuronal cada NTS_{lps} pasos. Existe una latencia (L) de 30 pasos de tiempo en la computación del estado neuronal ($L = NTS_{stages} \cdot NTS_{lps}$). Si F_{clk} es la frecuencia de reloj del sistema, entonces el rendimiento del sistema o número de datos actualizados por segundos queda definido por la Ecuación 15.

$$R_{sis} = \frac{F_{clk}}{NTS_{lps}}$$

Ecuación 15: Rendimiento del sistema. F_{clk} es la frecuencia de reloj global, NTS_{lps} es el número de pasos para realizar una actualización neuronal en la etapa más larga del cauce.

A modo de ejemplo, para una implementación con $NTS_{lps} = 6$ pasos de tiempo y $F_{clk} = 25$ Mhz resulta que el número de actualizaciones por segundo $R_{sis} = 4.16 \cdot 10^6$ actualizaciones/segundo.

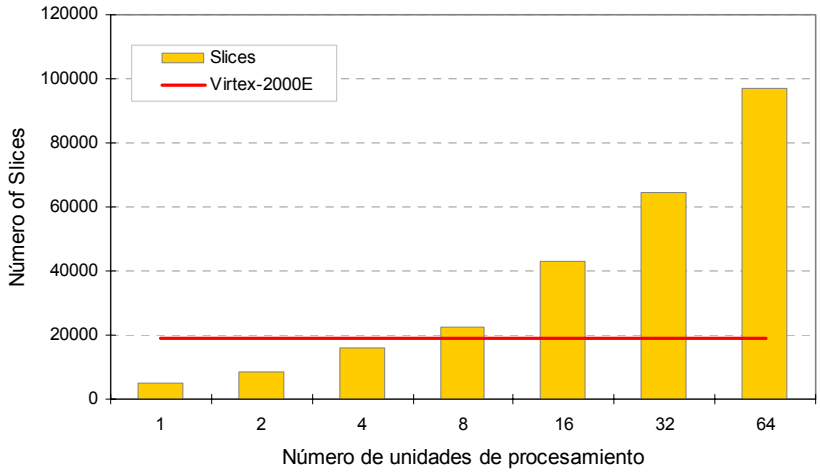


Ilustración 13: Recursos hardware consumidos en Slices; Crecimiento en el consumo de Slices en función del número de unidades de procesamiento funcionando en paralelo en una FPGA. En una FPGA Virtex-2000E es posible introducir hasta 4 unidades de procesamiento pudiendo simular una red de 1024 neuronas en tiempo real, con un paso de tiempo de simulación de 100µs.

Observado la Ilustración 13 y la Ilustración 14, donde se detallan los recursos hardware consumidos para una red de 1024 neuronas, pueden extraerse las siguientes conclusiones:

- Los recursos hardware se incrementan con el número de unidades de procesamiento.

- El número de "Slices" (unidades lógicas minimales de Xilinx) se incrementa aproximadamente linealmente con el número de unidades de computación incluidas en el chip (Ilustración 13).

- La escalabilidad de los recursos de memoria se incrementa ligeramente hasta 8 unidades de procesamiento, después de este, el incremento es mucho mayor (Ilustración 14). Hasta 8 unidades de computación los recursos de memoria son usados eficientemente, mientras que para un número mayor son sub-óptimos debido a que el número de neuronas simuladas en cada unidad de procesamiento es pequeño (para una red de 1024 neuronas).

- Utilizando una plataforma RC1000 es posible simular un sistema de 1024 neuronas e integrar 4 unidades de procesamiento paralelo en el chip.

- Los recursos internos del dispositivo FPGA, no dependen del número de neuronas que se simulan ni del número de eventos de entrada salida (pulsos neuronales). Esto es debido a que éstos son almacenados en recursos de memoria de la plataforma y luego son transferidos en paquetes hasta el interior de la FPGA. No obstante, la unidad de comunicación con los bancos de memoria PEG (cargador de evento sinápticos, Ilustración 11) tiene un coste de tiempo que depende del número de entras/salidas de eventos que son transferidos. Esto se describe en las siguientes secciones y en particular en la Ilustración 16.

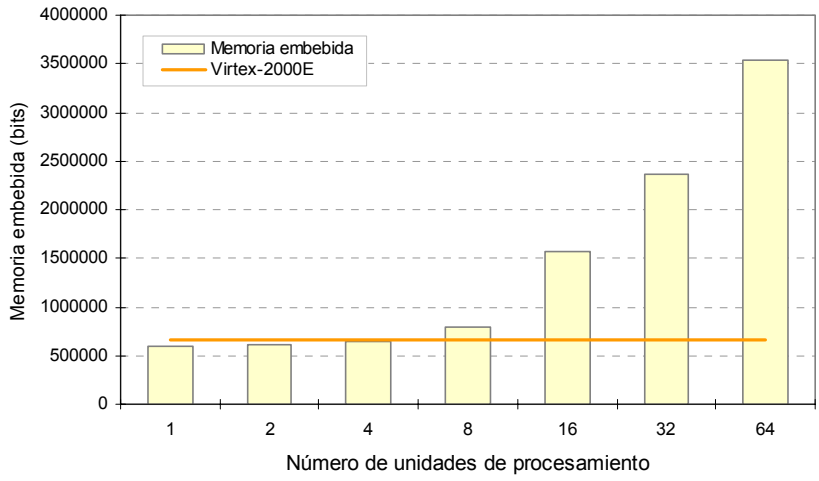


Ilustración 14: Recursos de memoria embebida consumidos; Para al menos 4 unidades de procesamiento, los recursos de memoria apenas se incrementan,

porque las EMB utilizadas son suficientes. Para más de 8 unidades de procesamiento el incremento es mucho mayor porque las EMB son sub-óptimas para este tamaño de red neuronal (1024).

3.8 Rendimiento del sistema

En la sección 3.4 se describió el esquema de computación basado en épocas de N_t ciclos de tiempo y cómo adoptando esta estrategia de computación se reducía la comunicación a través del bus PCI. En este apartado se profundiza en este ámbito haciendo especial hincapié en los requisitos para tiempo real.

3.8.1 Tiempo de computación por épocas

El tiempo necesario por el hardware para computar cada época $t_{época}$ se describe en la Ecuación 16, donde; $t^i_{tiempo\_ciclo}$ es el tiempo de computación de los ciclos de tiempo i (i = 0... N_t –1) y $NTS^i_{tiempo\text{-}ciclo}$ es el número de pasos de tiempo necesarios para computar un ciclo de computación i y F_{clk} es la frecuencia de reloj.

$$t_{época} = \sum_{i=0}^{N_t-1} t^i_{tiempo\_ciclo} = \frac{1}{F_{clk}} \sum_{i=0}^{N_t-1} NTS^i_{tiempo\_ciclo}$$

Ecuación 16: Tiempo de computación de una época de procesamiento

La unidad PEG (cargador de eventos sinápticos) y la unidad de procesamiento están trabajando en paralelo (véase la Ilustración 11) donde el límite en la velocidad de computación efectiva $NTS^i_{tiempo\_ciclo}$ es descrito por la Ecuación 17 y la Ecuación 18.

$$NTS^i_{tiempo\text{-}ciclo} = \mathrm{MAX}\left(NTS^i_{PEG}, NTS^i_{unidad-procesamiento} \right)$$

Ecuación 17

$$NTS^i_{tiempo\text{-}ciclo} = \mathrm{MAX}\left(1+\left(N^i_{pre-pre\text{-}sináptico} \cdot NTS_{lectura\text{-}pre}\right), L+\left(\frac{N_{neuronas} \cdot NTS_{ips}}{N_{UP}}\right)\right)$$

Ecuación 18

Por otro lado, el tiempo utilizado por la unidad PEG es variable dependiendo del número de eventos pre-sinápticos recibidos en cada ciclo ($N^i_{pre\text{-}sinápticos}$) y del número de pasos de tiempo necesarios para capturar un evento desde memoria externa hasta memoria embebida ($NTS_{lectura\text{-}pre}$).
El tiempo gastado por la unidad de procesamiento es fijo y depende de:

- número de neuronas ($N_{neuronas}$)
- número de unidades de procesamiento (N_{UP})
- número de pasos de tiempo de la etapa más larga del la estructura del cauce de datos (NTS_{lps})
- la latencia ($L = N_{estados} \cdot NTS_{lps}$)

El tiempo de computación de cada época ($t_{época}$) se describe en la Ecuación 19.

$$t_{época} = \frac{1}{F_{clk}} \sum_{i=0}^{N_t - 1} \text{MAX}\left(1 + \left(N^i_{pre-\text{sináptico}} \cdot NTS_{lectura-pre}\right), L + \left(\frac{N_{neuronas} \cdot NTS_{lps}}{N_{UP}}\right)\right)$$

Ecuación 19

Asumiendo que la unidad de procesamiento tarda más tiempo de cómputo que la unidad PEG y eligiendo la siguiente configuración; $N_t = 20$ ciclos, $N_{neuronas} = 1024$ neuronas, $NTS_{lps} = 6$ pasos de tiempo $F_{clk} = 25$ Mhz, entonces el tiempo consumiendo por el hardware (utilizando una única unidad de procesamiento) para computar una época es de $t_{época} \approx 4.94$ ms.
Es importante indicar que el tiempo de computación total de una época es ligeramente superior ya que en éste último cálculo no se ha tenido en cuenta el tiempo necesario para la comunicación, para la captura de los eventos pre y post sinápticos, el tiempo de intercambio de datos así como la sincronización de las hebras en la parte del componente software.

3.8.2 Rendimiento para tiempo real

Un posible valor acorde con la biología para el paso de tiempo es de 100μs. Para simulaciones de tiempo real, una época de 20 ciclos de tiempo tendría que ser computada en menos de 2ms.

Para lograr esto, se utilizan 4 unidades de computación obteniendo el rendimiento mostrado en la Ilustración 15. En ellas se puede ver cómo el tiempo de computación se decrementa con el número de unidades de procesamiento (tomando como ejemplo 1024 neuronas, 20 ciclos por época, un paso de tiempo de integración de 100μs y un número variable de eventos pre-sinápticos).

El diseño descrito es escalable y el procesamiento es inherentemente paralelo cuando el número de eventos pre-sinápticos no es muy elevado.

A modo de resumen, a continuación se enumeran el conjunto de factores que limitan el rendimiento óptimo del sistema:

- Si el número de eventos pre-sinápticos es muy grande, entonces la velocidad está limitada por la unidad PEG.

- Si todas las neuronas reciben un evento pre-sináptico en cada ciclo de un época (peor caso véase la Ilustración 15) entonces la unidad PEG nuevamente limita el rendimiento ya que el tiempo de evaluación es independiente del número de unidades de procesamiento.

- Si los eventos pre-sinápticos no son recibidos en cada época de evaluación (véase mejor caso de la Ilustración 15 y la Ilustración 16), entonces la velocidad de la unidades de procesamiento limita el rendimiento global del sistema y el diseño es perfectamente escalable.

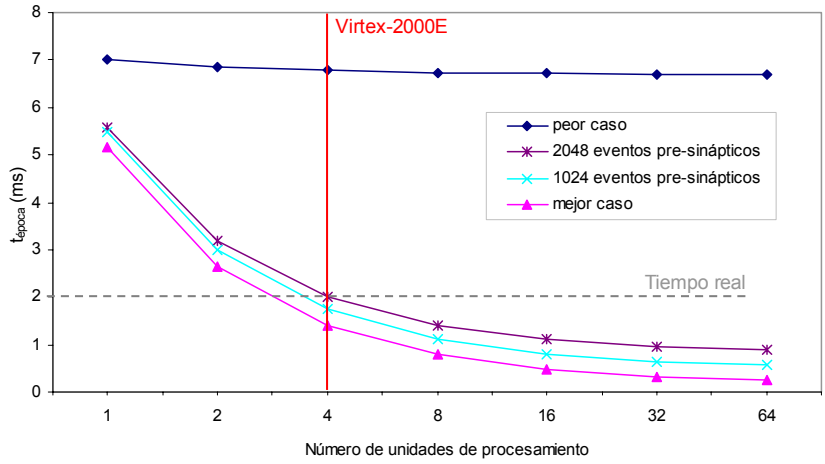


Ilustración 15: Estudio de escalabilidad; El coste para 1024 neuronas es de 20 ciclos por época computando a 25 Mhz, con un número variable de eventos pre-sinápticos (peor caso con 2048 eventos pres-sinápticos, el mejor caso tiene 0 eventos pre-sinápticos). El coste a través del bus PCI (100MB/s usando DMA) es de 0.2ms para 1024 eventos pre-sinápticos y de 0.3ms para 2048.

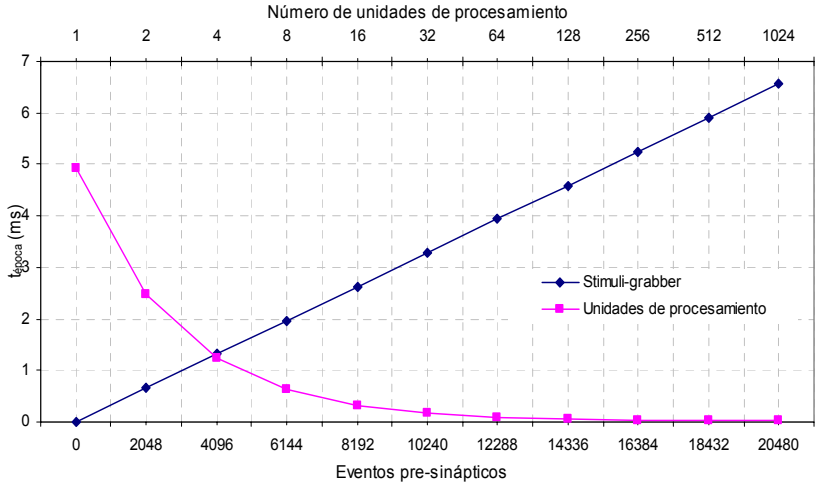


Ilustración 16. Estudio de escalabilidad: Tomando 1024 neuronas, 20 ciclos/época y 25 Mhz de frecuencia de reloj. El coste (omitiendo los tiempos de comunicación a través del bus PCI) depende del coste máximo de la unidad PEG y de la unidad de procesamiento ya que estas unidades funcionan en paralelo. La configuración óptima es 4 unidades de procesamiento computando 4096 eventos pre-sinápticos en tiempo real. En este caso las dos coinciden.

La Ilustración 16 representa el coste en tiempo para distintas unidades de procesamiento paralelo. En ella se observa que el coste de la

unidad PEG depende del número de eventos (ver eje inferior), mientras que el coste de las unidades de procesamiento depende del número de unidades funcionando en paralelo (eje superior).
Éste gráfico muestra como la tarjeta de co-procesamiento puede computar 1024 neuronas en tiempo real cuando recibe menos de 6144 eventos pre-sinápticos por época (una época tiene 20 pasos de tiempo de 100μs de tiempo de simulación, para tiempo real una época necesita ser computada en menos de 2ms). La configuración óptima se consigue con 4 unidades de procesamiento en la FPGA y 4096 eventos pre-sinápticos por época.

El coste en tiempo mostrado en la Ilustración 15 y la Ilustración 16 se ha calculado omitiendo el tiempo de transmisión a través de bus PCI (100MB/s de ancho de banda en medido utilizando DMA).
La transmisión de los eventos de salida es también insignificante ya que consumen en torno a 0.1ms para un paquete de 1024 eventos. El máximo número de eventos de salida por época está limitado por el número de neuronas simuladas. Por otro lado, el tiempo de comunicación para los eventos de entrada es significativo (0.3ms para 2048 eventos pres-sinápticos y 0.6ms para 4096). Este tiempo de transmisión necesita ser tenido en cuenta cuando se determina la capacidad real del coprocesador hardware.

3.9 Simulaciones del Cerebelo en tiempo real

El rendimiento del sistema depende de la conectividad de la red neuronal y de la carga; Esta última interpretada como el número de eventos pre-sinápticos por época. Pero es importante probar el sistema en un contexto biológico realista. Para esto se ha simulado un modelo basado en el sistema biológico formado por la oliva-inferior y el cerebelo. En concreto la Oliva Inferior y el Cerebelo están implicados en tareas de aprendizaje de movimientos sincronizados así como la comparación entre movimientos deseados, procedentes de la corteza motora, y movimientos ejecutados correctamente [13]. A modo de inciso existen aplicaciones realistas que utilizan esquemas neuronales basados en el cerebelo (véase [9]).

El Cerebelo y la Oliva inferior están conectados mediante las *fibras trepadoras* transmitiendo señales de error, por ejemplo durante el

aprendizaje del reflejo vestíbulo-ocular motor (movimiento automático del ojo durante el movimiento voluntario de la cabeza para mantener fijo un objetivo).
Otra tarea donde está involucrado el cerebelo y la oliva inferior es en el seguimiento de objetos por los ojos o TVSP (*Tracking Visual Smooth Pursuit*) [11] [53].

A modo de aclaración, en el primer experimento (reflejo vestíbulo-ocular motor) se fija la vista en un punto y se mueve la cabeza horizontalmente. De forma automática, el circuito neuronal responde moviendo los ojos para mantenerlos fijos en un punto, dentro del campo visual. Para comprender mejor este reflejo fije la vista en un punto objetivo y a continuación muévase la cabeza de lado a lado. Se dará cuenta que los ojos se mueven de forma "automática" para mantener el punto objetivo centrado en el campo visual.

El segundo experimento (TVSP) consiste en la estabilización del movimiento de los objetos proyectados en la retina para ayudar a percibirlos en detalle. Pensemos, por ejemplo, en lo que sucede cuando un tenista sigue con la mirada la pelota para golpearla. El tenista de forma "automática" conoce la posición siguiente de la pelota a partir de las posiciones anteriores. En este tipo de experimentos, la trayectoria es predicha para desplazar el ojo de forma correcta y fijar el objetivo dentro del campo visual receptivo. Esta predicción no puede hacerse de forma voluntaria (entiéndase como una tarea de la corteza motora sin la ayuda del cerebelo) debido principalmente a la velocidad con que ocurren los acontecimientos.
En este tipo de experimentos el cerebelo juega un papel fundamental como sistema "preditor de trayectorias" y por ello resulta interesante su estudio con este ejemplo.

A modo de detalle en la Tabla 3 se describen el número de células, su conectividad y los parámetros de la configuración a nivel celular. El cerebelo es un ejemplo de sistema que necesita ser simulado en tiempo real. El cerebelo está involucrado en el control y coordinación fino de movimientos. Todos los detalles del modelo y su aplicación al seguimiento de objetos se detallan en [66]. Las conclusiones referenciadas en [66] pueden resumirse en los siguientes puntos:

1. El número de conexiones a cada neurona varía en gran medida. El cerebelo no representa una red heterogénea de neu-

ronas, pero contiene patrones de conectividad extremadamente altos y bajos. Cada célula granular recibe menos de diez entradas mientras que una célula de Purkinje recibe cientos de miles de entradas.

2. El número de eventos pre-sinápticos por época (y la carga computacional) es ampliamente distribuida en la simulación como muestra en la Ilustración 17 A. Las células de la oliva inferior disparan en torno a 1Hz, esto en la capa de células granulares parece ser en su mayor parte una capa silenciosa con pequeñas ráfagas de 100Hz. Mientras que para las células de Purkinje la media de disparo es de 80Hz.

La versión simplificada y artificial del problema de seguimiento de objetos puede ser fácilmente simulada utilizando la plataforma hardware. Más aún, proporcionando un contexto suficientemente rico es posible dilucidar nuevas funciones cerebelosas que pueden aparecer probando el modelo en tiempo real e interactuando con el mundo real. Todas las neuronas son procesadas en el co-procesador neuronal a excepción de las fibras que llegan al cerebelo (*Mossy fibres*) y las entradas realimentadas. Considerando la información referente a las tareas de seguimiento de objetos, tales como posición y movimientos de los ojos o movimientos de objetos en imágenes en la retina. Estas señales son traducidas a trenes de eventos utilizando software. El modelo ha sido construido para explorar el papel de la plasticidad de las "Fibras Trepadoras" y el aprendizaje en la corteza cerebelar. Esto utiliza un algoritmo Hebbiano para el aprendizaje entre las fibras paralelas y las células de Purkinje entre las fibras Mossy y el núcleo cerebeloso y entre las fibras Mossy y las células granulares. Estos detalles del aprendizaje son ampliamente descritos en [66].

El incremento de velocidad o "*speed-up"* debido a la introducción del co-procesador hardware se describe en la figura Ilustración 17B. El número total de células en el modelo se cambia modificando el número de células granulares (véase la tercera columna de la Tabla 3).

Tipo de célula	Actividad (Hz)	Nº	Conexiones de entrada	Conexiones de salida	$T_{refractario}$ (ms)	τ_{exc} (ms)	τ_{inh} (ms)
Granular	0-100	242-1010	5	10	10	2	2
Purkinje	30-120	2	Nº Granulares	2	20	5	5
Interneuronas	5-60	4	Nº Granulares	2	20	5	5
Golgi	5-60	4	Nº Granulares	Nº Granulares	20	5	5
DCN (*)	30-70	2	23	2	10	2	2
Oliva inferior	0-10	2	2	1	70	5	5
Fibras musgosas	20-80	21	X	Nº Granulares +6	0	0	0

Tabla 3: Tipos de de células y parámetros para el modelo de la oliva inferior. (*) Núcleo cerebelar profundo. La tercera columna indica el número de neuronas.

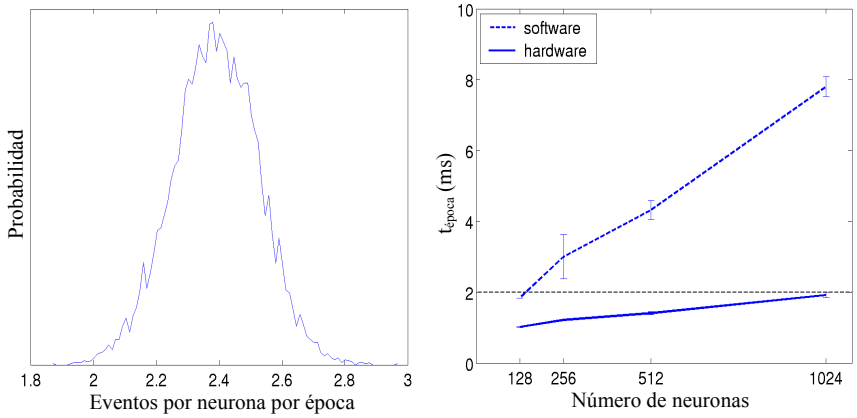


Ilustración 17: Rendimiento del modelo cerebelar: (A) El número de eventos pre-sinapticos por neurona por época para el modelo cerebelar con 242 células granulares aplicados a la tarea de seguimiento. (B) El rendimiento en velocidad del acelerador hardware sobrepasa al software con respecto del total de neuronas.

3.10 Discusión

En este capítulo se ha presentado un sistema híbrido hardware- software basado en una plataforma para la simulación en tiempo real de redes neuronales de eventos con un modelo neuronal que incluye inyección gradual de carga y sinapsis modeladas con con-

ductancias. Este esquema de computación permite simular modelos neuronales sin las restricciones impuestas por otros esquemas dirigidos por eventos. El coprocesador hardware está basado en hardware reconfigurable (FPGA) y se ha adoptado una estrategia de computación que hace uso exhaustivo recursos de computación paralelos. En particular se ha diseñado un esquema de procesamiento con segmentación de cauce con un rendimiento de 6 ciclos de reloj en la etapa más larga. Además se han segmentado los recursos de memoria que hacen posible la escalabilidad del sistema. En esta línea, varias unidades de procesamiento pueden funcionar eficientemente en paralelo evitando inter-bloqueos.

Es difícil comparar el sistema con otros simuladores de eventos tanto hardware como software debido a la resolución temporal del proceso de integración, sus objetivos y además los modelos neuronales son diferentes, pero comparando con otras plataformas de computación especificas [35] [54] [112] [120] [71] esta aproximación simula redes neuronales a pequeña y mediana escala aunque con una alta resolución temporal (100μs) y en tiempo real. La primera diferencia esencial estriba en que el sistema está pensado para tiempo real en contraposición con "simuladores rápidos" y que la motivación del desarrollo de la arquitectura es investigar el comportamiento de sistemas neuronales interaccionando con plataformas robotizadas, trabajando en ciclo cerrado (sensor-motor) en el mundo real donde el procesamiento en tiempo real es crítico.

Otras implementaciones hardware basadas en tecnología analógica VLSI ponen de manifiesto aplicaciones relacionadas con interfaces con neuronas reales (la intensidad que se inyecta en una neurona sigue un patrón que depende de la actividad sináptica de la célula) [113] [70]. En ellas se describen sistemas biológicos conectados a sistemas hardware que interaccionan conjuntamente para el estudio del comportamiento de diversos tejidos neuronales.
Las restricciones temporales en este campo de aplicación son muy exigentes y no demasiadas las restricciones tanto en tamaño como en topología de conexión.
La plataforma ha sido diseñada para aplicaciones de control en robótica y estudios de interrelaciones sensor-motor. En este campo es interesante la capacidad de simulación de diferentes topologías de red de pequeño y mediano tamaño.

El esquema de computación híbrido hardware/software permite simular redes heterogéneas, puesto que la topología se define en software. Como se describe en la Ilustración 7 hay tres procesos software ejecutándose concurrentemente en el PC. El componente de enrutamiento consulta la topología de la red y traduce los eventos post-sinápticos en eventos pre-sinápticos incorporando los pesos de las conexiones y retardos. Otro módulo dedicado a la comunicación con la plataforma hardware tiene en cuenta los retardos de las conexiones y las coincidencias en las direcciones de eventos presinápticos que van hacia la misma neurona para optimizar el ancho de banda entre el software y el hardware. Este esquema permite simular redes neuronales con distancias inter-neuronales diferentes, lo que se traduce en la posibilidad de establecer retardos inter-neuronales variables. Este aspecto es importante en la simulación del sistema de la oliva inferior y el cerebelo. Por ejemplo, las fibras paralelas que conectan con las células de Purkinge han sido especificadas mediante una distribución de retardos y la velocidad de las conductancias para las *fibras trepadoras* son ajustadas en función de la longitud del Axon para asegurar que todas las sinapsis tienen un retardo fijado [5].

Se ha elegido un modelo neuronal con sinapsis controladas por la actividad de entrada. Esto incrementa la complejidad computacional y requiere circuitos multiplicadores que son los elementos limitantes en cuanto a consumo de recursos lógicos en la FPGA. No obstante, esta característica no limita la capacidad de cómputo global del sistema debido a que se integra como una etapa más del cauce regido por la etapa S2 en la Ilustración 10. Este modelo sináptico (sinapsis modeladas como conductancias) es una importante característica de estudio tal como se describe en [67] [7] [112] y permite adoptar mecanismos de plasticidad sináptica a nivel de red como es el caso de la modulación de la ganancia en la capa granular, modelada por medio de inhibición [112].

El modelo neuronal utilizado incluye inyección gradual de carga en los módulos sinápticos. Esta característica permite el estudio de procesos de sincronización siendo un tópico importante de estudio tal como se describe en [97] [101] [6].

Multitud de estudios han demostrado que la sincronización entre conexiones neuronales conectadas de forma inhibitoria y recíprocamente se facilita gracias a retardos en las conexiones finitos y la inyección gradual de carga en las sinapsis [125] [7], aunque otros mecanismos como las sinapsis eléctricas juegan un papel complementario [81].
Estas conexiones con retardos definidos retrasan la generación del potencial de acción en una de las células desde el instante en que llega el pulso de inhibición sináptica [101].

La tecnología basada en circuitos FPGA ha experimentado un gran avance no sólo en la densidad de integración sino en el desarrollo de lenguajes de descripción hardware más eficientes. En este caso el sistema ha sido descrito con un lenguaje de alto nivel de descripción hardware [121], que ha facilitado la gestión eficiente de la memoria y demás recursos lógicos. El diseño es totalmente modular y puede ser fácilmente configurado con diferentes niveles de paralelismo. Con la plataforma utilizada (RC1000) se han integrado 4 unidades de procesamiento computando en tiempo real con un tiempo de integración de 100μs y 1024 neuronas recibiendo 4096 eventos presinápticos. Se ha probado el esquema de computación utilizando una plataforma más potente RC2000 [12], donde se han integrado 18 unidades de procesamiento (FPGA con 6000 millones de puertas lógicas) pudiendo computar 90 neuronas en paralelo (debido a las 5 etapas de cauce de cada unidad de procesamiento). Con esta plataforma se ha obtenido un factor 3 (dependiendo del tamaño de la red) en la eficiencia del sistema con respecto a la plataforma RC1000. No obstante el sistema se ha implementado con plataforma de propósito general pudiendo desarrollar otras de propósito específico que incorporen mayores recursos de memoria y mejor ancho de banda de comunicación con el PC.

Con la aproximación actual, la actividad media de la red es crítica. Si el número de eventos pre-sinápticos recibidos es muy alto, el tiempo de computación se incrementa. Con un tiempo de integración pequeño (100μs en los experimentos del apartado 3.9) la actividad por paso de tiempo será baja. En el modelo cerebeloso el número medio de eventos pre-sinápticos por neurona por cada 2ms es de 2.4 (véase [7]). De hecho la aproximación actual utiliza una computación interactiva entre el PC y la plataforma de modo que se

puede seguir la traza de la simulación de forma difícilmente alcanzable con esquemas dirigidos por eventos.

Existen dos factores principales que limitan la escalabilidad del sistema para la simulación de redes neuronales muy grandes:

1. El ancho de banda de comunicación Hardware/Sofware (esto ha motivado el desarrollo de plataformas con PCI-X).

2. La limitación en el paralelismo inherente en el dispositivo. Cada FPGA tiene un número limitado de recursos lógicos.

El objetivo de este desarrollo ha sido la simulación de circuitos neuronales de pequeña y media escala en tiempo real. El tiempo real impone restricciones en el tiempo de simulación y el número de eventos pre-sinápticos capaces de ser computados en cada ciclo de simulación. En la arquitectura actual el número máximo de neuronas que pueden ser simuladas depende de la disponibilidad de recursos de memoria de la plataforma. Una mejora importante sería la utilización de circuitos que trabajen con memoria DDRAM. Esto permitiría escalar el sistema hasta $64 \cdot 10^6$ neuronas utilizando un módulo de 1GB de memoria dinámica, mientras que el número de conexiones sinápticas estaría limitado por la capacidad de memoria del computador.
El incremento en velocidad del sistema comparado con simuladores software con redes de tamaño pequeño se muestra en la Ilustración 17 B. La paralelización de simuladores a gran escala utilizando sistemas multiprocesador de memoria distribuida (por ejemplo, cluster de computadores) con tarjetas FPGA a modo de aceleradores neuronales es una posibilidad que se encuentra bajo estudio. No obstante la eficiencia de estos sistemas dependerá, entre otros factores, de la topología de la red a simular, de la capacidad de comunicación entre nodos, y de la carga de trabajo de cada computador que simularía una parte de la red concreta. Una ventaja de estos sistemas radica en la posibilidad de incluir en la plataforma aceleradora módulos de aprendizaje, módulos de enrutamiento de eventos etc. de modo que liberen a la CPU del computador para dedicarla a la gestión de comunicación entre nodos.

La comparación entre la plataforma propuesta con otros simuladores dirigido por eventos no es sencilla debido a que se han adoptado

un modelo neuronal que implementan inyección gradual de carga, necesaria para definir sinapsis con diferentes constantes de tiempo. Esta característica es difícil de incorporar en simuladores dirigido por eventos pero existen aproximaciones que exploran esta posibilidad [109] [120].

El esquema de procesamiento con segmentación de cauce permite añadir nuevas funcionalidades al sistema tales como aprendizaje, sin que esto suponga un incremento en el tiempo de computación por neurona.

El cerebelo juega un papel en la dinámica de integración sensomotor necesaria para encontrar una coordinación motora. Estudiar esto con agentes reales requiere un motor de simulación de tiempo real. La elección de una neurona con sinapsis modeladas con conductancias variables fue adoptada para simular observaciones de las neuronas reales. Esto es, el modelo basado en conductancias fue obtenido a partir de la electrofisiología de diferentes tipos de células del cerebelo [136] [15] [102] [114]. Este modelo es muy costoso de simular computacionalmente por lo que el modelo simulado en hardware es una aproximación simplificada del biológico original.

El sistema desarrollado tiene como objetivo facilitar el estudio de la integración senso-motor en sistemas de control artificiales basados en arquitecturas biológicas y por tanto, este trabajo es un paso adelante en el estudio de los sistemas senso-motores neuromorficos. Esto es el desarrollo de un sistema de control neuronal que incluya todos los sensores y actuadores en un sistema robótico independiente. Esto se verá en capítulos posteriores.

Capítulo 4
Desarrollo de una arquitectura dirigida por eventos para la simulación de sistemas neuronales completos

En el capítulo anterior se introdujo un ejemplo de arquitectura para la simulación de circuitos neuronales dirigida por tiempo. En este capítulo se aborda el desarrollo de un nuevo simulador, más avanzado, que utiliza un esquema de procesamiento dirigido por eventos. Se realiza un estudio de eficiencia del sistema así como un análisis en profundidad de los riegos que aparecen en la estructura de cómputo con segmentación de cauce. También se propone una estrategia de búsqueda paralela de eventos (búsqueda de evento de tiempo menor) altamente eficiente.

4.1 Introducción

Los simuladores de circuitos neuronales están aún lejos de ser capaces de emular eficientemente sistemas nerviosos a gran escala, no obstante se han conseguido importantes avances en este campo [30] [2] [72] [38] [124] [44] [25].

En este capítulo se describe una arquitectura de computación que utiliza recursos de paralelismo masivo mediante dispositivos FPGA. El motor de simulación utiliza un esquema dirigido por eventos y un método de búsqueda de alto rendimiento del evento de menor tiempo.
Se ha utilizado una arquitectura con cauce de datos segmentado para computar los eventos en paralelo, minimizando recursos de computación ociosos por la espera resultados parciales de etapas anteriores del camino de datos.

El análisis teórico del sistema desarrollado demuestra que es capaz de computar 2.5 millones de eventos por segundo auque, como se irá describiendo a lo largo del las secciones siguientes, esta cifra quedará mermada en la práctica debido principalmente a dependencias de datos y acceso a recursos compartidos.

El conjunto de elementos de computación está compuesto solamente por un chip FPGA y cinco bancos de memoria SRAM externa al chip. Por lo tanto, esta aproximación es escalable y tiene un gran interés para la realización de experimentos que requieren motores de cómputo empotrados como por ejemplo, agentes autónomos en robótica.
A pesar de que existen aproximaciones hardware diseñadas en FPGAs que implementan esquemas de computación dirigidos por tiempo [110] [24] [39], las características enumeradas anteriormente han motivado el desarrollo de esquemas de procesamiento dirigido por eventos. Este tipo de esquemas de computación son usualmente implementados en software [109] [17] [18] [77] [61] pero han sido también implementados en hardware [82] [120]. En los esquemas de simulación dirigidos por eventos, las variables de los estados neuronales son actualizadas cuando la neurona emite o recibe un evento. Por lo tanto el motor de simulación salta de un evento a otro

sin que sea necesario simular el intervalo de tiempo entre dichos eventos. De este modo toda la actividad de la red es dispuesta en orden cronológico y procesada secuencialmente.
El esquema de computación dirigido por eventos es muy apropiado para plataformas de computo secuencial tales como PC convencionales. Por otro lado, debido a la necesidad de procesar los eventos de forma secuencial, es difícilmente paralelizable.

En general, los esquemas usuales dirigidos por eventos utilizan una cola de eventos ordenados cronológicamente. En este caso el objetivo es reducir al máximo el número de accesos necesarios para la inserción y extracción de los elementos de la lista de eventos. La forma más eficiente de implementar esta funcionalidad (de forma secuencial) es utilizar una estructura de datos basada en árboles binarios de búsqueda (entiendase como estructura de datos en el ámbito de las Ciencias de la Computación), como es el caso del Heap [122]. El Heap y otras estrategias de ordenación ventajosas para este problema, se analizan ampliamente en la sección 4.2.2.

El problema de utilizar una estructura como el Heap es que no se puede paralelizar. El Heap es inherentemente secuencial y por tanto resulta muy ineficiente en implementaciones hardware basadas en FPGA, debido a dos factores fundamentales:

a) No se explota la capacidad de paralelismo del circuito reconfigurable debido a la imposibilidad de implementación paralela.

b) La baja frecuencia de reloj con que trabajan los chips FPGA (usualmente entre 10 y 500 Mhz) hacen difícil la competencia con implementaciones realizadas en computadores con procesadores trabajando a Ghz.

Contrariamente a esta aproximación, el esquema de procesamiento implementado en el diseño, utiliza una lista desordenada de eventos. El controlador de la lista busca el siguiente evento a ser procesado utilizando módulos de comparación paralelos. De este modo es posible optimizar la inserción y extracción de eventos liberando uno de los cuellos de botella del sistema. No obstante, esta aproximación consumirá importantes recursos de hardware que han de ser tenidos en cuenta en el desarrollo del resto de etapas del sistema final.

Esta y otras estrategias de búsqueda de eventos, se estudian ampliamente en las secciones 4.2.1 y siguientes.

4.2 Descripción del esquema y ciclo de computación

El ciclo de computación neuronal se puede resumir en dos tareas de alto nivel que se repiten cíclicamente durante todo el proceso de simulación. Suponiendo que el sistema se encuentra configurado, es decir, las variables neuronales están almacenadas en memoria así como la topología de la red, las tareas de alto nivel quedan resumidas en:

1. Captura y envío de los paquetes de eventos correspondientes a un intervalo 100ms: Se capturan los eventos a procesar y se envían los ya procesados, bien transmitidos desde el PC o bien desde sistemas físicos conectados externamente al chip como por ejemplo un robot utilizando circuitos intermedios de adaptación se señales a eventos (Véase el Capítulo 5 para mayor detalle). Los eventos de entrada se insertan directamente en la lista de eventos a procesar.

2. Procesamiento y generación de eventos: (a) Extracción de evento de tiempo menor, (b) Lectura, procesamiento y escritura de las variables de estado de las neuronas origen, destino y conexiones sinápticas. En esta etapa también se generen los eventos de salida.

El esquema de computación engloba toda la arquitectura de procesamiento incluida en el interior del chip FPGA más un conjunto de recursos externos al chip, tales como bancos de memoria y buses de transferencia de datos (véase la Ilustración 18). Estos componentes quedan resumidos en:

- Una unidad de cómputo de eventos con segmentación de cauce de 9 etapas.
- Unidad de gestión y control de comandos.
- Interfaz de control para memoria externa.

- Bloque de procesamiento que implementa la búsqueda paralela del evento de tiempo menor.
- Módulo de gestión de eventos de entrada/salida.
- Unidad de adaptación de señales externas (para conexión de sistemas físicos).

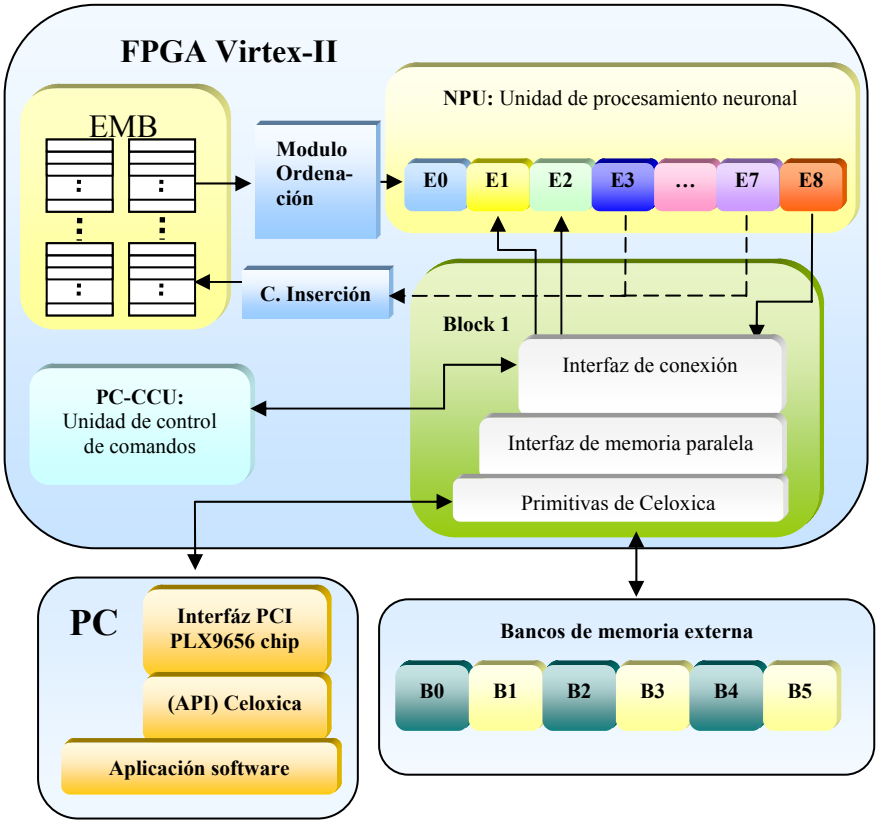


Ilustración 18: Diagrama de bloques de la arquitectura dirigida por eventos. La lista de eventos se almacena en recursos de memoria EMB (bloques de memoria embebida) para facilitar el proceso de inserción y búsqueda paralelo. Por otro lado la topología de red y los estados neuronales son almacenados en memoria SRAM externa.

4.2.1 Descripción de los elementos que componen el sistema de computación

- Unidad de cómputo de eventos con segmentación de cauce

Debido a la extensión y complejidad de esta unidad se ha decidido dedicarle una sección aparte. (Véase el apartado 4.3 para mayor detalle).

- La unidad de gestión y control de comandos:

Gestiona la interacción del sistema con el PC. Mediante esta unidad es posible analizar paso a paso qué está ocurriendo en cualquier unidad de procesamiento del sistema. A modo de ejemplo permite pausar el simulador, realizar volcados de memoria, recuperar registros estadísticos de colisiones en el cauce o capturar las variables de estado de cualquier neurona o conexión de la red.

- Interfaz de control de memoria externa:

Es un módulo hardware que implementa un acceso paralelo y con segmentación de cauce de los bancos de memoria externa. Este interfaz permite el acceso concurrente a más de un banco de memoria al mismo tiempo permitiendo una tasa de transferencia de una palabra de 32 bits por ciclo de reloj con una latencia inicial de un ciclo. El módulo permite trabajar a frecuencias de reloj diferentes entre el circuito de acceso a memoria y el circuito de procesamiento. De este modo se puede duplicar la tasa de transferencia entre memoria y el sistema.

- Búsqueda paralela:

Debido a la extensión y complejidad de esta unidad se ha decidido dedicarle una sección aparte. (Véase el apartado 5.2.3 para mayor de talle).

- Módulo de gestión de eventos de entrada/salida:

La arquitectura de computación tiene la posibilidad de trabajar en dos modos. Un modo de simulación y un modo de tiempo real. Al tratarse de una arquitectura dirigida por eventos que puede interactuar con un sistema externo, es necesario incluir dos módulos de gestión de eventos. El primero es un módulo empaquetador, el cual captura los eventos de entrada desde el sistema físico que han

ocurrido durante un período de 100ms. El segundo elemento, es un generador de eventos en tiempo real. Éste componente recibe los paquetes de eventos de salida del sistema y los emite, en orden cronológico en la ranura de tiempo exacta, hacia el sistema físico externo. Ambos módulos poseen buffers de eventos para mantener la coherencia temporal frente a perturbaciones en la actividad de la red. Es decir, la interacción con agentes externos se realiza adaptando un esquema dirigido por tiempo.

- Unidad de adaptación de señales:

La unidad de adaptación de señales realiza la conversión a eventos de señales físicas externas tanto de entrada como de salida. Este módulo está íntimamente relacionado con las características del agente físico externo que se conecta al sistema. Por ejemplo si se conecta un brazo robotizado es preciso transformar las posiciones, velocidades y pares de fuerza a trenes de eventos codificados en frecuencia, por ejemplo. Esto es ampliamente discutido en el Capítulo 5, donde se analiza en profundidad un ejemplo completo de interconexión con un sistema físico.

4.2.2 Estrategias de selección del siguiente evento: Posibles implementaciones

Es necesario procesar los eventos en orden cronológico. La estrategia de búsqueda del siguiente evento a procesar (evento con marca de tiempo menor) durante el proceso de cómputo, constituye un paso crítico para la búsqueda del máximo rendimiento del sistema debido a que es una etapa del camino de datos segmentado. Para realizar la búsqueda se puede elegir entre diferentes algoritmos (*Heap* [122], búsqueda secuencial, algoritmos de ordenación rápida *Quick-Short* [45]) que permiten seleccionar el elemento de tiempo menor. Sin embargo la mayoría de estos algoritmos están pensados para ser ejecutados en máquinas secuenciales y, por tanto, son difícilmente paralelizables de manera que exploten las posibilidades de los circuitos de hardware reconfigurable.

La mayoría de los autores que han implementado simuladores dirigidos por eventos en máquinas con arquitectura secuencial, han llegado a la conclusión que el algoritmo más eficiente para la bús-

queda del elemento de tiempo menor es el Heap [7] [109]. Sin embargo y como se demuestra en las siguientes líneas, el Heap no es una opción tan prometedora cuado se utiliza en implementaciones hardware a pesar de tener una eficiencia de orden logarítmico [122].

A modo de estudio se ha implementado el HEAP en un chip FPGA observando, que borrar o insertar un evento para una implementación a 33Mhz consume entre 132 y 2310 ciclos tal como se muestra en la Tabla 5 mientras que utilizando una técnica de búsqueda paralela como la que se describe en profundidad en el apartado 4.2.3 consume un número mucho menor de ciclos para un mismo volumen de elementos. Esta afirmación a priori podría parecer errónea debido a que estamos comparando un algoritmo con eficiencia logarítmica (Heap) con un algoritmo con eficiencia lineal (búsqueda paralela). Sin embargo, al calcular la eficiencia o número de pasos necesarios para realizar una operación de alto nivel (extraer, insertar elementos) no se tienen en cuenta el número de sub-pasos necesarios para realizar dicha operación (véase la Ilustración 19). Por ejemplo: La extracción del elemento menor de un HEAP supone realizar al menos log(n) pasos para mantener la estructura, pero a su vez cada paso requiere la realización de otras operaciones minimales (intercambio de variables, actualizaciones de direcciones o punteros etc) que necesitan un número de ciclos de reloj no contabilizados en el cálculo de la eficiencia. Por ello, a pesar de que podría encontrarse un número suficientemente alto de elementos donde la eficiencia del HEAP supera a la búsqueda paralela, en la práctica, para un conjunto de eventos razonablemente manejable para nuestro propósito resulta ser más eficiente la búsqueda paralela.

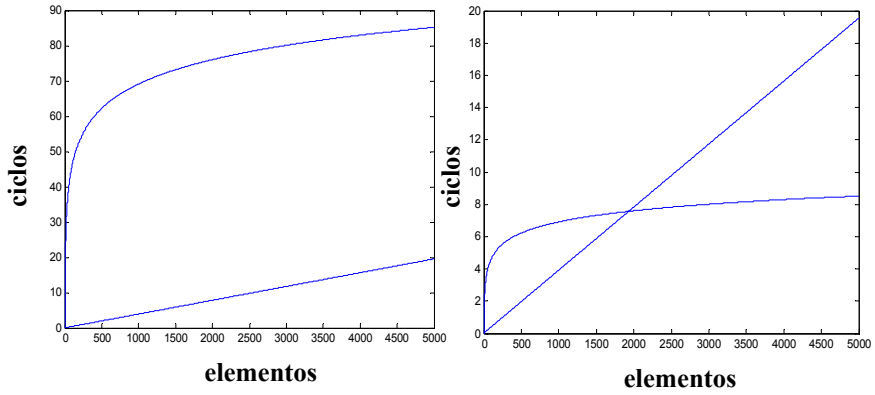


Ilustración 19: Ejemplos de comparación del algoritmo Heap con la búsqueda paralela**. (Izquierda)**; Supuesto que el Heap consume 10 ciclos de reloj por paso. Obsérvese que para un número de eventos razonablemente manejable la búsqueda paralela necesita realizar menor número de pasos que el Heap. **(Derecha);** Eficiencia del Heap frente a la búsqueda paralela, en número de pasos necesarios para completar una tarea frente al número de elementos para un caso en que el coste por paso sea de 1 ciclo de reloj.

Operación	Eficiencia peor caso
Encontrar el mínimo	O(1)
Borrar el mínimo	O(log n)
Insertar (un elemento)	O(log n)
Actualizar (un elemento)	O(log n)
Unir dos heap	O(n)

Tabla 4: Eficiencia teórica del algoritmo de ordenación Heap, Nótese que las operaciones minimales de inserción, borrado etc. se realizan únicamente con un elemento. En el caso de operar con N elementos tendría un coste de N·(eficiencia de la operación minimal).

Operación	Consumo
Borrar o Insertar 1 evento en el Heap	Entre 4 y 70 µs
	Entre 132 y 2310 ciclos

Tabla 5: Consumo en ciclos y tiempo del algoritmo de ordenación Heap para una implementación hardware basada en FPGA funcionando a 33 Mhz

En las secciones siguientes se analizan en profundidad diferentes alternativas de almacenamiento de los eventos en memoria para la optimización de la búsqueda del elemento de tiempo menor especialmente orientadas para su viabilidad en dispositivos de hardware reconfigurable.
La limitación en la eficiencia de este tipo de circuitos, depende del número de elementos direccionables en el mismo ciclo de reloj y, una vez seleccionado el evento, del número de ciclos necesarios para recuperar el resto de campos asociados. Si esta etapa es la más lenta, la latencia del cauce depende directamente de ella.

En resumen, dos factores son importantes para conseguir un rendimiento óptimo durante el proceso de búsqueda:

1. Máximo paralelismo en el acceso a la etiqueta de tiempo del evento

2. Máximo paralelismo en la lectura de los campos del evento de tiempo mínimo.

4.2.3 Arquitectura para la búsqueda del evento de tiempo menor: Búsqueda paralela

La arquitectura integrada en el chip FPGA del sistema final está formada por un conjunto de bancos de memoria embebida (EMBs), un conjunto de buffers intermedios y finalmente, un conjunto de comparadores paralelos (véase la Ilustración 20). Este conjunto de buffers segmenta el proceso de búsqueda en varias etapas de comparación, produciendo así una arquitectura micro segmentada.

La arquitectura de búsqueda maneja únicamente eventos. Un evento o pulso neuronal está caracterizado por cuatro campos de datos: etiqueta de tiempo, identificador de sinapsis, neurona fuente y neurona destino.
Suponiendo una implementación en la que solamente se utilizan bancos de memoria embebida de forma no entrelazada, las etiquetas de tiempo quedarían distribuidas en diferentes bloques, con todos los campos de un mismo evento agrupados secuencialmente uno

tras otro. Los bancos de memoria se conectan a un árbol de comparadores y buffer intermedios que conforman el motor de búsqueda. Los eventos están distribuidos en 128 bancos de memoria embebida (EMB) con doble puerto, lo que permite leer en paralelo 256 eventos en tan solo un ciclo de reloj. Dado que un evento está formado por 4 campos de 32 bits y cada banco de memoria embebida tiene un tamaño de 512X32, es posible manejar una lista de 2^{14} eventos. Estos elementos se cargan en un buffer de precarga construido con memoria distribuida (DMB) y permite que los 256 elementos puedan ser introducidos en 32 comparadores de 8 elementos cada uno, produciendo 32 candidatos que serán almacenados en un segundo buffer de precarga. Estos 32 candidatos se introducen, nuevamente, en otro circuito de 4 comparadores de 8 elementos cada uno produciendo 4 candidatos que serán almacenados en un tercer buffer de precarga. Por último, un sencillo comprador de 4 elementos proporciona la salida con el evento de tiempo menor de los primeros 256 elementos. Este proceso se repite con los siguientes 256 eventos hasta completar los 2^{14} obteniendo el evento de tiempo menor de toda la lista. Todo el cómputo se estructura en un cauce segmentado con lo que el evento de tiempo menor de una lista 2^{14} eventos se obtiene en 69 ciclos de reloj.

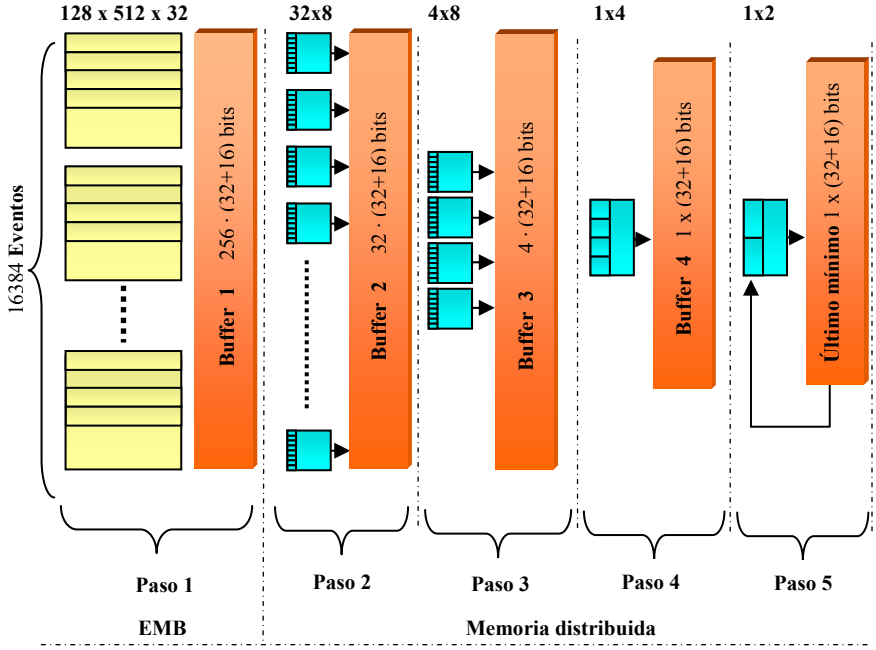


Ilustración 20: Árbol de búsqueda paralelo. Obsérvese el conjunto de comparadores en cada etapa del cauce segmentado.

Esta arquitectura puede ser escalada usando bancos de memoria externa para gestionar listas de eventos de mayor tamaño, a costa de reducir la velocidad de búsqueda significativamente, debido a la limitación de acceso secuencial y al ancho de palabra de los bancos de memoria externa. (Esto es ampliamente discutido en las secciones 4.2.4 y siguientes).

$$N_{ciclos-busqueda} = 4 + \left\lceil \frac{T_{EMB}}{256} \right\rceil$$

Ecuación 20

El número de ciclos de reloj requeridos en la estructura de búsqueda ($N_{ciclos\text{-}búsqueda}$) queda definido en la Ecuación 20. T_{EMB} es el número de eventos almacenados en todas las EMBs utilizadas por el módulo de búsqueda y $\lceil\ \rceil$ simboliza la función de redondeo al siguiente vector entero.
La constante 4 surge como consecuencia del número de ciclos consumidos en el llenado del cauce micro segmentado.

Al final del proceso de búsqueda, una vez que el elemento de tiempo mínimo ha sido encontrado, el sistema precisa de 5 ciclos más (no incluidos en la Ecuación 20) para recuperar los campos de datos del evento. Este último proceso es secuencial, debido a que los registros del evento han de ser leídos secuencialmente.

Esta arquitectura ha sido la elegida para su integración en el sistema final porque establece un buen equilibrio entre rendimiento y coste de recursos. Además es una arquitectura paralelizable y según el esquema de memoria adoptado, escalable en mayor o menor medida.

4.2.4 Estrategia escalable de selección del siguiente evento: Eventos almacenados completamente en recursos de memoria externos al chip FPGA

Es posible almacenar los eventos en bancos de memoria SRAM externos al chip FPGA tal como se muestra en la Ecuación 21, no obstante, en este caso el tiempo de acceso paralelo a las etiquetas de tiempo de los diferentes eventos está restringida. Por ejemplo, con una plataforma de computación con M_{SRAM} bancos de memoria SRAM (con 32 bit de longitud de palabra) accesible en paralelo, es posible recuperar M_{SRAM} eventos en cada acceso. De este modo el número de ciclos de reloj ($N_{ciclos\text{-}búsqueda}$) necesarios para buscar el evento de tiempo menor se muestra en la Ecuación 21. Ambos términos de la expresión están relacionados con el proceso de búsqueda. En este caso $N_{ciclos\text{-}búsqueda}$ depende linealmente del número de eventos N_E, del número de etapas del circuito segmentado C y del tiempo consumido para recuperar el resto de campos de los eventos almacenados en memoria externa.

$$N_{ciclos-búsqueda} = \frac{N_E}{M_{SRAM}} + C + \frac{d_f}{M_{SRAM}}$$

Ecuación 21

Por lo tanto, el acceso paralelo a los eventos está limitado por el número de módulos M_{SRAM}. Este esquema es muy escalable y puede almacenar un gran número de eventos.

Finalmente, el hecho de acceder a bancos de memoria externa está limitado por las características de los chips SRAM utilizados (latencia, máximo rendimiento de salida de datos, etc.) y ha de ser tenido en cuenta en implementaciones puntuales.

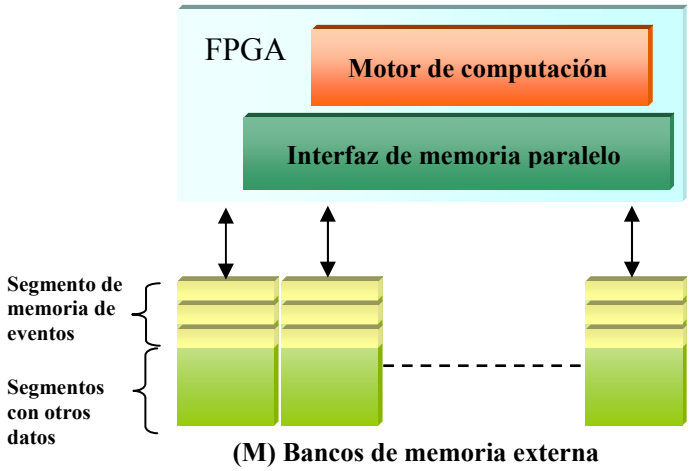


Ilustración 21: Lista de eventos almacenada en recursos de memoria externa. Utilizando M bancos de memoria externa con M puertos de acceso en paralelo.

4.2.5 Estrategia escalable de selección del siguiente evento: Eventos almacenado completamente en bancos de memoria embebida

Para permitir un alto grado de paralelismo en el proceso de búsqueda del evento de tiempo menor, se pueden almacenar los eventos en memoria embebida (bancos de memoria integrados en el interior del chip FPGA). Esta estrategia es la adoptada finalmente para el procesador neuronal que se ha desarrollado y puede verse en la Ilustración 22. En este caso el nivel de paralelismo depende del número de EMBs dedicados. Si se utilizan bancos de memoria de doble puerto y se distribuyen los eventos de forma balanceada, es posible acceder en paralelo a $P_a=2\cdot M_{EMB}$ (donde M_{EMB} es el número de EMBs utilizadas). Si cada evento está compuesto por d_f campos de datos de 32 bits y se almacenan todos en EMBs, en un chip FPGA Virtex II es posible almacenar $N_E=M_{EMB}\cdot 512/d_f$ eventos. Con

esta aproximación es posible conseguir un acceso en paralelo considerable a las etiquetas de tiempo. Por el contrario el consumo de recursos de EMB para almacenar otros datos es considerable. (Recuérdese que el resto de campos de los eventos no son necesarios durante el proceso de búsqueda).

$$N_{ciclos-búsqueda} = \frac{N_E}{2 \cdot M_{EMB}} + C + d_f$$

Ecuación 22

Esta estrategia está limitada en escalabilidad en términos de carga de trabajo (máximo tamaño de la lista de eventos) pero maximiza el acceso paralelo (P_a) durante el proceso de búsqueda. El problema radica en que el resto de campos de un evento han de ser leídos secuencialmente gastando más de un ciclo de reloj al final de proceso de búsqueda. El número de ciclos utilizados por esta estrategia de búsqueda se puede ver en la Ecuación 22.

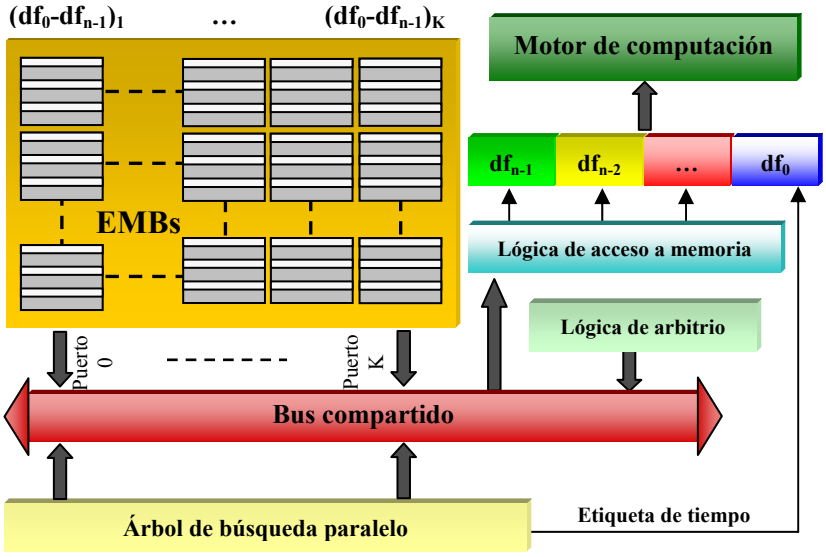


Ilustración 22: Todos los campos de los eventos almacenados en EMBs. Requiere la utilización de lógica de arbitrio para la gestión del acceso a los bancos de EMB.

4.2.6 Estrategia escalable de selección del siguiente evento: Etiquetas de tiempo en bancos de memoria embebida específicos

Es posible almacenar de forma separada los campos de datos de los eventos guardando las etiquetas de tiempo en bancos de EMBs dedicados (el resto de campos en otros EMBs) tal como se muestra en la Ilustración 23.

$$N_{ciclos-búsqueda} = \frac{N_E \cdot d_f}{2 \cdot M_{EMB}} + C + 1$$

Ecuación 23

Con esta configuración el esquema requiere puertos de acceso dedicados para las etiquetas de tiempo y otros para el resto de campos. En este caso el número de ciclos de reloj consumidos en proceso de búsqueda depende linealmente del número de eventos N_E y de los campos de datos d_f como se muestra en la Ecuación 23.

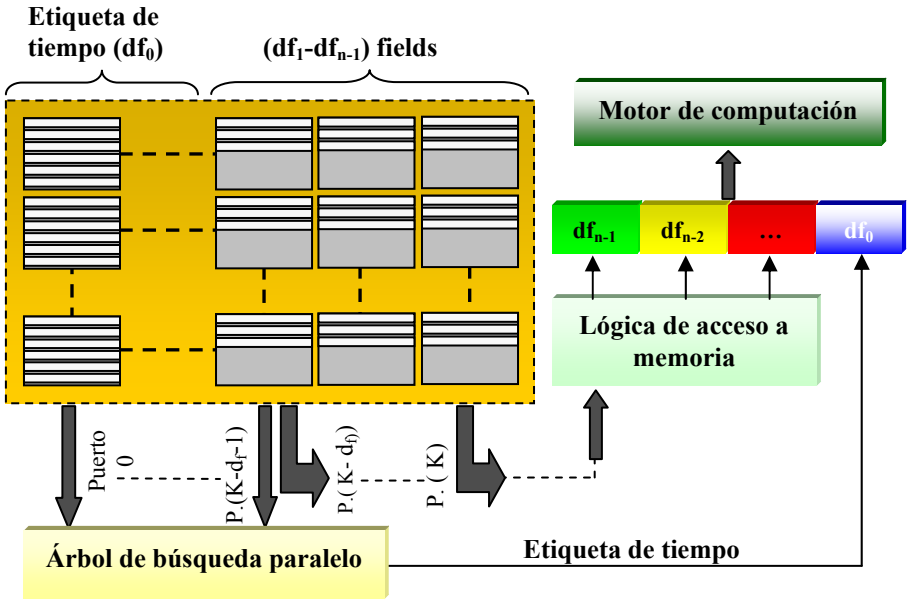


Ilustración 23: Etiquetas de tiempo almacenadas en bancos de memoria (EMBs) específicos.

Obsérvese que es preciso multiplicar por d_f debido a que parte del número de bancos de EMBs está dedicado a almacenar sólo las etiquetas de tiempo de cada evento y por tanto disminuye el nivel de paralelismo de este esquema.

4.2.7 Estrategia escalable de selección del siguiente evento: Campos de un evento almacenado de forma entrelazada en EMBs

Otra opción consiste en almacenar de forma entrelazada los eventos de diferentes EMBs de modo que durante el proceso de búsqueda están accesibles todas las etiquetas de tiempo. Pero una vez obtenido el evento con tiempo mínimo, se pueda recuperar el resto de campos del evento en un solo ciclo de reloj.

$$N_{ciclos-búsqueda} = \frac{N_E}{2 \cdot M_{EMB}} + C + 1$$

Ecuación 24

Esta aproximación maximiza el número de puertos paralelos para el proceso de búsqueda y también, optimiza el acceso al resto de los campos de datos. El número de ciclos consumidos se puede calcular siguiendo la Ecuación 24. En la Ilustración 24, se muestra un ejemplo de esquema de entrelazado para eventos con 4 campos. Utilizando este esquema, es posible acceder al mismo tiempo a las etiquetas (DS 0-0, DS 1-0…) y al resto de campos del evento en sólo un ciclo (DS 0-0, DS 0-1, DS 0-2…).

El esquema entrelazado de memoria es una técnica muy utilizada en los procesadores vectoriales para explotar el paralelismo como es ampliamente discutido en [28]. No obstante el consumo de recursos hardware es mayor (dada la lógica asociada al decodificador de direcciones) que la estrategia utilizada en este estudio, por lo que se ha optado por la segunda opción dado que el rendimiento obtenido no es muy superior.

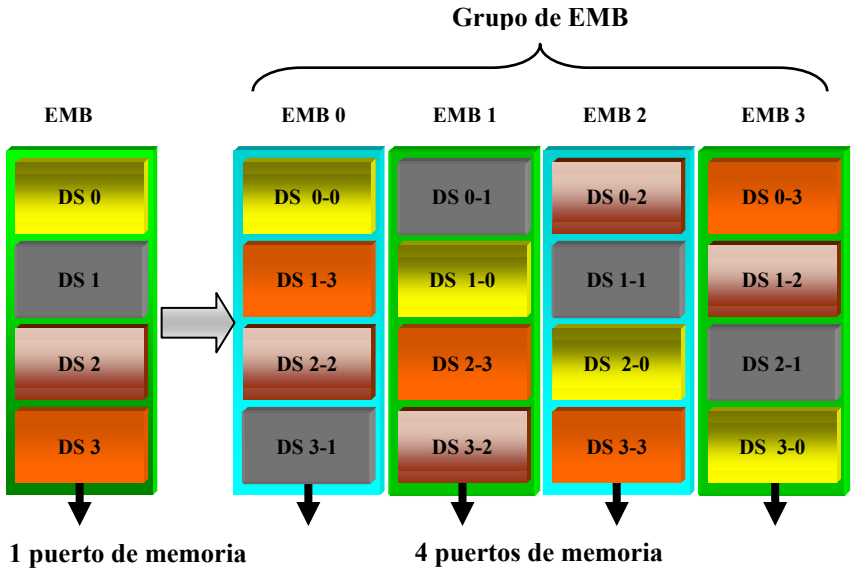


Ilustración 24: Ejemplo de entrelazado de 4 vías.

4.2.8 Estrategia de selección del siguiente evento escalable: Híbrida memoria externa y memoria embebida

En esta técnica, sólo las etiquetas de tiempo son almacenadas (de forma balanceada) en bancos de memoria embebida para maximizar el paralelismo ($P_a=2 \cdot M_{EMB}$), mientras que el resto de campos se almacenan en bancos de memoria externa. Esta característica no es muy limitante en cuanto a rendimiento global del sistema, debido a que solamente un evento (evento ganador de tiempo mínimo) será completamente recuperado durante cada iteración del cauce de datos de alto nivel. De este modo se maximiza el tamaño de la lista de eventos que pueden ser almacenados en EMBs.

Si se utilizan bancos de memoria externa con 2 ciclos de retardo en el acceso, el rendimiento se puede calcular siguiendo la Ecuación 24 pero añadiendo dos ciclos más debido a la latencia de las memorias externas.

Esta latencia puede ser disminuida a un ciclo por dato leído mediante la utilización de estructuras de micro-segmentadas en el acceso a memoria (estrategia utilizada en la implementación). No obstante la latencia inicial en el llenado del cauce sigue siendo de 2 ciclos.

4.3 Camino de datos segmentado de procesamiento de eventos

El procesamiento está dividido en nueve etapas con la peculiaridad de que se ha incluido la propia gestión del mismo como una etapa más. El conjunto de etapas son los siguientes:

SS: Etapa de desplazamiento de registros intermedios; los registros intermedios almacenan los resultados parciales del resultado del procesamiento de cada etapa del cauce en cada interacción.

S0: Búsqueda del evento de tiempo menor; esto se hace mediante el árbol de búsqueda paralelo comentado en las secciones anteriores.

S1:

- Acceso a memoria externa para recuperar las variables de estado de la neurona fuente
- Acceso a memoria externa para recuperar las variables de estado de la neurona objetivo
- Acceso a memoria externa para recuperar las variables de la conexión (topología)

S2: Desplazamiento del buffer de pre-salida para los eventos que salen del simulador

S3: Desplazamiento del buffer de pre-inserción para los eventos ínter-neuronales

S4: Aprendizaje

S5: Cálculo de la contribución sináptica

S6: Generación de eventos de salida cuello axónico o "*Axon Hillock*" y validación de cambios.

S7: Actualización del estado de la conexión ínter-neuronal (actualización del instante en que se activó la conexión)

S8: Actualización de las variables de estado de la neurona objetivo en memoria externa

En un cauce de datos segmentado sería ideal que todas las etapas tuviesen el mismo coste de cómputo en ciclos de reloj y que todas las etapas accediesen a recursos no compartidos. Desgraciadamente, en este caso práctico, esto no es así por lo que es preciso establecer políticas de gestión del cauce que solucionen eficientemente los problemas inherentes de la computación paralela. Estas políticas han sido resumidas en:

1. Política de sincronización de etapas del camino de datos.

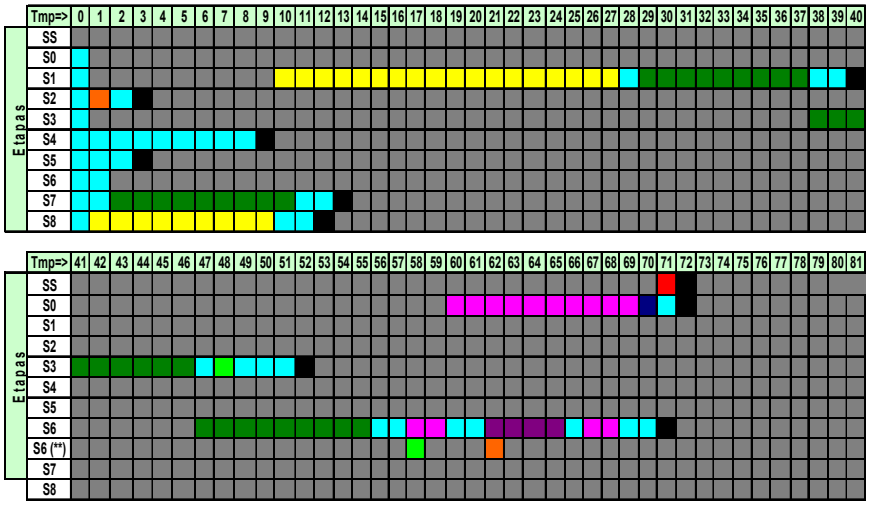
2. Resolución de riesgos (ínter bloqueos por acceso compartido a recursos, bloqueos por riesgos de dependencia de datos, vaciado del cauce).
3. Balanceo de carga (coste en ciclos de cómputo) entre las diferentes etapas del cauce.

Estos puntos son ampliamente discutidos en los apartados 4.3.1, 4.3.2 y 4.3.3.

4.3.1 Política de sincronización de etapas del camino de datos

Para sincronizar el procesamiento de las diferentes etapas del cauce, se utilizan barreras y banderas. Cada etapa tiene asociada una bandera (indica cuando acaba) y una barrera que bloquea la etapa al final del procesamiento. La etapa SS comprueba las banderas del resto de etapas (S0…S8). Cuando todas las etapas han terminado realiza el desplazamiento de los registros resultado. Es decir, copia en cada etapa el registro resultado de la etapa anterior. Esta acción ocupa únicamente un ciclo de reloj y puede realizarse al inicio de cada etapa, pero se ha hecho así (en una etapa diferente) por simplificar y clarificar la gestión del cauce ya que el rendimiento no se ve afectado. Una vez que se ha realizado el desplazamiento de los registros de resultados parciales, se desbloquean las etapas, permitiendo que se inicie un nuevo ciclo de computación.
La gestión de banderas se realiza de forma programada, mientras que la gestión de barreras se hace de forma automática. Ya que esta gestión es inherente a la estructura utilizada del propio lenguaje Handel-C [12].

La Ilustración 25 y la Ilustración 26 muestran cronológicamente el acceso y la gestión de los recursos disponibles por el sistema durante un ciclo completo de computación. En concreto, la Ilustración 25 muestra la sincronización en el acceso compartido a los recursos. Describe que recursos necesita cada etapa y cuándo están disponibles. Es importante aclarar que en ocasiones aunque esté disponible el recurso no puede ser utilizado debido al conjunto de dependencias lógicas que son discutidas en el apartado 4.3.2.



Recursos externos al chip FPGA

(*)	banco 0	Control
	banco 1	almacenamiento de variables neuronales
	banco 2	topología de la red
(*)	banco 3	estímulos de entrada al sistema
	Banco 4	estímulos de salida del sistema

Recursos internos al chip

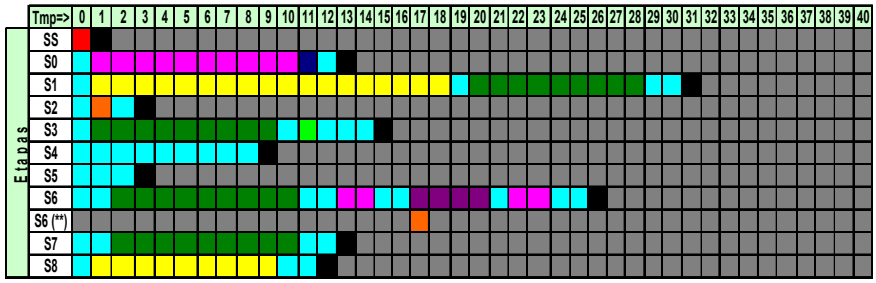
	EMB (memoria embebida)	sistema de búsqueda paralelo
	buffer de presalida	almacenamiento temporal de eventos de salida del sistema
	buffer de pre-inserción	almacenamiento temporal de eventos de entrada generados por el sistema
	registros intermedios	
	Buffer de eventos extraidos	almacenamiento temporal de eventos extraidos del sistema de búsqueda paralelo

Acción realizada

	En espera
	Porcesamiento local

(*) recursos no utilizados por el pipeline durante el cómputo de un paquete de eventos procedentes del bus PCI o del robot
(**) procesamiento paralelo en la misma etapa

Ilustración 25: Cronograma de utilización de recursos de las diferentes etapas del cauce durante un instante de la computación. El color negro indica la finalización de la etapa.



Recursos externos al chip FPGA

(*)	banco 0	Control
	banco 1	almacenamiento de variables neuronales
	banco 2	topología de la red
(*)	banco 3	estímulos de entrada al sistema
	Banco 4	estímulos de salida del sistema

Recursos internos al chip

	EMB (memoria embebida)	sistema de búsqueda paralelo
	buffer de presalida	almacenamiento temporal de eventos de salida del sistema
	buffer de pre-inserción	almacenamiento temporal de eventos de entrada generados por el sistema
	registros intermedios	
	Buffer de eventos extraidos	almacenamiento temporal de eventos extraidos del sistema de búsqueda paralel

Acción realizada

	En espera
	Porcesamiento local

(*) recursos no utilizados por el pipeline durante el cómputo de un paquete de eventos procedentes del bus PCI o del robot
(**) procesamiento paralelo en la misma etapa

	fin de ciclo de computación

Ilustración 26: Cronograma de colisiones en el acceso a recursos al inicio de una simulación. Obsérvese las colisiones que aparecen entre la etapa S1 y la etapa S8 ó entre la etapa S3 y las etapas S6 y S7.

4.3.2 Resolución de riesgos: Inter-bloqueos por acceso compartido a recursos, bloqueos por riesgos de dependencia de datos y vaciado del cauce

El análisis de la gestión del cauce muestra la posibilidad de inter- bloqueos por acceso compartido a los recursos por más de una etapa. Estos riesgos surgen sobre todo cuando se accede a memoria externa. Por ejemplo la etapa S1 rivaliza con la etapa S7 por el acceso a los bancos de memoria externa ya que en el mismo ciclo de computación la etapa S1 necesita recuperar el registro asociado a una neurona fuente mientras que la etapa S7 precisa escribir la contribución de un evento ya computado en etapas anteriores. La solución de esta interacción pasa por establecer una política de acceso a memoria por turnos de prioridad.

Además puede ocurrir que sea la misma neurona la que se está procesando al mismo tiempo en varias etapas del cauce (por ejemplo en el caso de una topología de una neurona realimentada consigo misma). En este caso se inducen ciclos de cómputo vacío en el cauce, de forma que no se computa el evento hasta que no se escribe un resultado previo en memoria.

Un riesgo de dependencia de datos muy importante ocurre cuando el evento que se está computando produce una excitación neuronal tal que provoca que la neurona excitada dispare. Esto, a priori, no es un problema siempre que el retardo sináptico asociado a la sinapsis de salida no tenga un retardo tal que produzca un evento de tiempo inferior al evento que acaba de entrar en el cauce para ser procesado. En este caso todos los cambios realizados en etapas anteriores del cauce han de ser deshechos, para mantener la coherencia temporal en el procesamiento de eventos.
Para solucionar este riesgo se ha utilizado un buffer de prealmacenamiento. Es decir, cada operación de escritura en cualquiera de los bancos de memoria (incluidos los dedicados para el sistema de ordenación paralelo) se realiza sobre registros de desplazamiento (funcionando como una cola) de longitud igual al tamaño de la estructura de datos de un evento. De tal forma que las escrituras se demoran hasta que se conoce si el procesamiento ha tenido o no éxito validando los cambios. La etapa que realiza la comprobación

y validación de cambios es la etapa S6 mientras que las etapas S2 y S3 se utilizan para desplazar y encolar los cambios.

La aparición de este riesgo supone una degradación en la eficiencia del sistema debido a la ruptura del cauce. La penalización es importante ya que supone insertar en el sistema de ordenación elementos ya extraídos previamente, invalidar los registros intermedios en las etapas del cauce, e iniciar un nuevo ciclo de espera por llenado del cauce.

La probabilidad de que ocurra esta condición (riesgo entre eventos) depende de la topología de la red, de la frecuencia de disparo media, del número de etapas del cauce y del retardo sináptico interneuronal. Dada la variabilidad de factores a tener en cuenta, es sumamente difícil describir una expresión válida para el cómputo de este tipo de riesgo, no obstante se ilustra a modo de ejemplo un cálculo cuantitativo concreto.
Por ejemplo: Supongamos una red de 100 neuronas densamente conectadas (todas con todas) con una latencia media entre conexiones de 3 ± 2ms (desviación estándar de una distribución Gausiana) donde cada neurona dispara con una frecuencia media de 10 Hz.
Como cada neurona dispara 10 veces por segundo y las neuronas están conectadas entre si, la actividad de la red será de $10 \cdot 100 \cdot 100 = 100000$ eventos por segundo. Si suponemos una distribución uniforme equiespaciada de los eventos, ocurrirán colisiones siempre que la diferencia temporal entre dos eventos multiplicada por el número de etapas del cauce previas a la detección del la colisión, sea mayor que el retardo mínimo sináptico.
Es decir, con un cauce de 8 etapas donde la etapa 6 detecta la colisión tendremos un espaciado temporal de 10 microsegundos · 6 etapas, por lo tanto aunque todos los eventos de entrada durante un segundo generen nuevos eventos, no se producirían colisiones, ya que la ventana temporal del cauce abarca únicamente un entorno de 60 microsegundos y el retardo sináptico como mínimo será de 3-2 = 1 milisegundo.

Con sinapsis normales (no eléctricas) esta característica no es un problema excesivamente grave porque existe un retardo mínimo distinto de cero. El problema surge con la simulación de sinapsis eléctricas ya que tienen un retardo prácticamente cero. En esta situación, y dependiendo de la frecuencia de disparo y del número de

ellas dependerá la eficiencia del sistema, ya que se producirá un porcentaje mayor de colisiones y por tanto mayor número de interrupciones en el cauce.

Etapa	Nº ciclos	Acción	Recurso
SS	Etapa de mayor retardo +1	Copia y desplazamiento de de registros intermedios	Registros intermedios
S0	$N_{ciclos} = \frac{256}{2 \cdot 16} + 8$ 18 ciclos	Obtener evento de tiempo menor (ejemplo para 256 eventos y 16 bancos de EMBs con 1 EMB = 256 palabras de 32 bits) - inicialización (1) - leer evento (8+8) - semáforo (1)	EMB Buffer de eventos extraídos
S1	1 + 9 + 9 + 9 + 2 = 31	Captura de variables neuronales y conexiones: - inicialización (1) - Leer neurona fuente (9) - Leer neurona objetivo (9) - Leer conexión (9) - semáforo (1) - semáforo (1)	Banco 1 Banco1 Banco 2
S2	1 + 1 + 1 = 3	Desplazamiento del buffer de pre-salida: - inicialización (1) - insertar y desplazar (1) - semáforo (1)	buffer de pre-salida (eventos de salida del sistema)
S3	1 + 9 +1+ +1+1+1 = 14	Desplazamiento del buffer de pre-inserción: - inicialización (1) - Leer conexión (9) - semáforo (1) - crear nuevo evento (1) - insertar y desplazar (1) - semáforo (1)	buffer de pre-inserción (eventos a insertar en EMB)
S4	1 + 1 + 1 + 4 + 1 = 8	Aprendizaje: - inicialización (1) - calculo influencia retardo sináptico (1) - calculo del peso sináptico (1) - decaimiento pasivo (4) - semáforo (1)	

S5	3	Contribución sináptica, cálculo del potencial de membrana: - inicialización (1) - calculo potencial de membrana (1) - semáforo (1)	
S6	2 + 9 + 1 + 1 + 4 + 1 + 6 + 1 + 4 +4 = 33	Generación de eventos de salida (*Axon Hillock*) y validación de de cambios: (*) se considera que dispara la neurona y no hay que vaciar el cauce por riesgos - inicialización (2) - leer conexión (9) - semáforo (1) - crear nuevo evento (1) - insertar evento en EMB (4) - semáforo (1) - escribir evento de salida en memoria (6) - procesamiento local (1) - insertar evento en EMB (4)	- Banco 2 - EMB - Banco 4 - buffer - pre-inserción
S7	2 + 9 + 2 = 13	Actualización del estado de la conexión ínter-neuronal - inicialización (2) - escribir conexión (9) - semáforo (1) - semáforo (1)	Banco 2
S8	1 + 9 + 2 = 12	Actualización de las variables de estado de la neurona objetivo en memoria externa - inicialización (2) - escribir neurona (9) - semáforo (1) - semáforo (1)	Banco 1

Tabla 6: Resumen de consumos de ciclos de computación de todas las etapas del cauce. Así como el número de ciclos de retención de cada recurso junto con la operación que se hace en cada etapa. Todas las etapas utilizan registros locales intermedios

A continuación se muestra la lista completa fruto del análisis de riesgos realizado sobre el cauce:

a) Riesgos por dependencia de recursos:

Acceso simultaneo a bancos o registros de memoria físicos:

- S1 (lee) con etapa S8 (escribe) por el banco 1
- S1 (lee) con etapa S3 (lee), S6 (escribe) y S7(escribe) por el banco 2
- S0 (extracción) con S6 (inserción) por EMB
- S0 (inserción + desplazamiento) con S6 (lectura) por acceso a buffer temporal de eventos extraídos
- S3 (escritura + desplazamiento) con S6 (lectura) de buffer temporal de evento de inserción
- S2 (escritura + desplazamiento) con S6 (lectura) de buffer temporal de eventos de salida

b) Causas de vaciado del cauce:

- En la etapa S6 se genera un evento de tiempo menor que el mayor de los eventos introducidos hasta el momento.

c) Riesgo por dependencia de datos

- Problemas de lecturas antes que escrituras: En el cauce se procesan ocho eventos al mismo tiempo. Es posible que la etapa S1 lea una neurona que está siendo actualizada por la etapa 8 cuyo evento asociado ha sido procesado en etapas anteriores. Luego el estado de las variables neuronales que se está capturando en S1 no corresponde temporalmente con el que tendría que tener si no se da prioridad de escritura a la etapa 8.

d) Optimizaciones

- La eficiencia del cauce baja notablemente cuando una neurona tiene un alto índice de conectividad de salida, es decir, está conectada con multitud de neuronas de la red. En esta situación cuando la neurona genera un evento habría que insertar tantos eventos secuencialmente como conexiones de salida tiene la neurona. Para solucionar eficientemente este problema, lo que

se hace es insertar únicamente el evento correspondiente a la primera conexión sináptica de salida. De este modo cuando se procese ese evento se insertará un nuevo evento correspondiente a la siguiente conexión pendiente. Esta tarea la realiza la etapa S3 y S6.

4.3.3 Balanceo de carga (coste en ciclos de cómputo) entre las diferentes etapas del cauce

La eficiencia del sistema de cómputo segmentado, entre otros factores tales como el acceso a recursos compartidos etc., está directamente relacionada con el número de etapas que acceden al mismo recurso compartido y con el número de ciclos consumidos por etapa. En este diseño se ha buscado un equilibrio entre el número de ciclos consumidos por cada etapa de modo que el rendimiento dependa de la etapa más larga y ésta sea lo mas corta posible en tiempo.

A nivel teórico la etapa que consume mayor número de ciclos es la S1 con 31 ciclos (véase la Tabla 6) con una lista de eventos de 2048 que trabajando con un reloj de 33Mhz obtiene un rendimiento del cauce de 1,06 Meps (millones de eventos por segundo) y entre 66 y 80 Mhz produce aproximadamente 2.5 Meps. No obstante estos límites no son en la práctica alcanzables debido al conjunto de riesgo e ínter bloqueos descritos en los apartados 4.3.1 y 4.3.2.
La etapa S0 descrita en la Tabla 6 muestra un consumo variable en el consumo de ciclos debido a que la eficiencia del sistema de búsqueda paralelo depende directamente del número de eventos de la lista y del número de bancos de memoria ocupados.

4.4 Modelo neuronal

La arquitectura de computación descrita, es válida para múltiples modelos neuronales, de hecho la computación del estado neuronal es una etapa del cauce que puede ser vista como una caja negra. La única restricción es que el modelo neuronal permita que las variables neuronales puedan ser actualizadas de forma discontinua.

En el diseño presentado, se ha elegido un modelo neuronal sencillo con sinapsis aditivas. De hecho es una simplificación del modelo presentado en el Capítulo 3 ya que el objetivo ha sido probar la funcionalidad y rendimiento del sistema más que un modelo neuronal complejo.
Se han particularizado cuatro células nerviosas (Purkinge, Granular, célula de Golgi e Interneuronal) pertenecientes a la estructura cerebelosa aunque el sistema puede soportar cualquier tipo de neurona con diferentes constantes de tiempo.

La Ilustración 27 muestra el modelo neuronal descrito en la Ecuación 25. La Ilustración 27 (a) representa la plasticidad sináptica dependiente de los tiempos entre pulsos de entrada y salida (STPD) "*Spike Time Dependent Plasticity"* expresada en la Ecuación 25. La Ilustración 26 (b) representa el termino de decaimiento pasivo indicando el valor que ha de ser restado al potencial de membrana en función del tiempo transcurrido entre el evento actual y el último evento recibido. La expresión que rige este comportamiento se muestra el la Ecuación 25.

La Ilustración 27 (b, trazo superior), muestra el término de decaimiento pasivo de una célula Granular con constante de tiempo τ y un decaimiento pasivo débil, mientras que la Ilustración 27 (b, trazo inferior) para una célula de Purkinje con un fuerte decaimiento pasivo.

$$\Delta W_i^{t_k} = 27 \cdot \left(t_k^i - t_j^i \right) \cdot e^{\frac{\left(t_k^i - t_j^i \right)}{10}}$$

$$V_x^{t_k} = V_x^{t_j} - \left[\frac{V_{\max}}{\tau} \cdot \left(t_k - t_j \right) \right]$$

$$V_x^{t_K} = V_x^{t_J} + W_J^{t_J}$$

$$Salida = \left\{ \begin{array}{l} 1 \; si V_x^{t_K} \geq Umbral \\ 0 \; si V_x^{t_K} < Umbral \end{array} \right\}$$

Ecuación 25: Conjunto de ecuaciones que definen el modelo neuronal. De arriba hacia abajo: Aprendizaje sináptico, decaimiento pasivo, contribución sináptica,

generación de eventos de salida (*Axon Hillock*). Vx denota el potencial de membrana y W el peso sináptico. Los pesos son calculados de acuerdo con la primera expresión y dependen del retardo entre un evento que llega en un tiempo t_K y un evento que llegó en t_J.

La arquitectura de procesamiento descrita en las secciones anteriores es válida para múltiples modelos neuronales. De hecho, el procesamiento del estado neuronal se puede ver como una caja negra pudiendo implementar diferentes modelos neuronales. La única restricción en el modelo neuronal es que los estados de las variables puedan ser actualizados de forma discontinua.

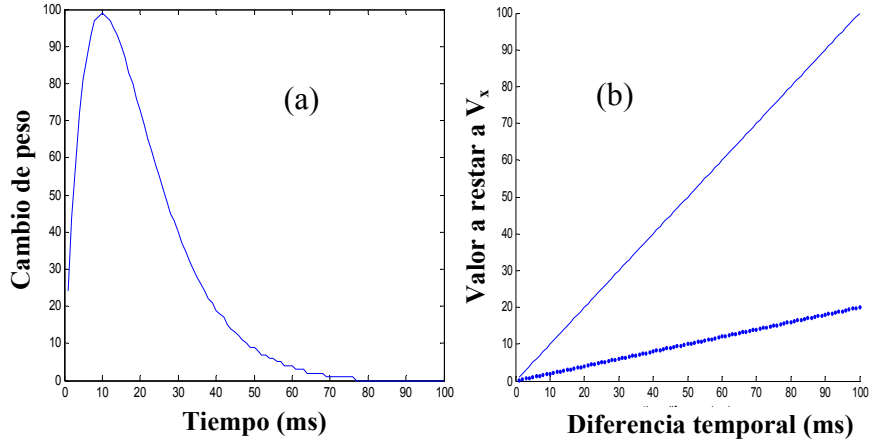


Ilustración 27: Modelo neuronal, (a) plasticidad temporal, (b) decaimiento pasivo del potencial de membrana para una célula Granular (gráfica superior b) y para una célula de Purkinje (gráfica inferior b).

4.4.1 Rendimiento de la simulación y recursos hardware consumidos

Es difícil comparar el rendimiento de éste sistema con otras aproximaciones que utilicen diferentes modelos de redes neuronales. Actualmente, uno de los más eficientes simuladores de neuronas de pulsos dirigido por eventos en software [109] es capaz de computar 0.8 millones de eventos por segundo utilizando un procesador AMD a 2.8 Ghz. Esto indica que a través de diseños de propósito especifico que trabajan con relojes de 2 órdenes de magnitud por debajo de los ordenadores convencionales, el sistema es capaz

de superar en rendimiento en más de un factor 2. Hay otros simuladores sencillos de neuronas de eventos [17] [18], pero incluyen modelos simplificados de neuronas, y topologías de red no compatibles con lo descrito en este trabajo.

La superficie mostrada en la Ilustración 28 ha sido obtenida utilizando una topología de red (todas con todas con retardos sinápticos pequeños). En este caso, el riesgo entre eventos no afecta significativamente al rendimiento del sistema. Tal y como puede verse el rendimiento no depende del tamaño de la red, sólo depende de la actividad global de la red. A modo de ejemplo daría lo mismo procesar una red de 10 millones de neuronas con una actividad de un 1 % que una red de 1 millón de neuronas con una actividad de un 10 %.

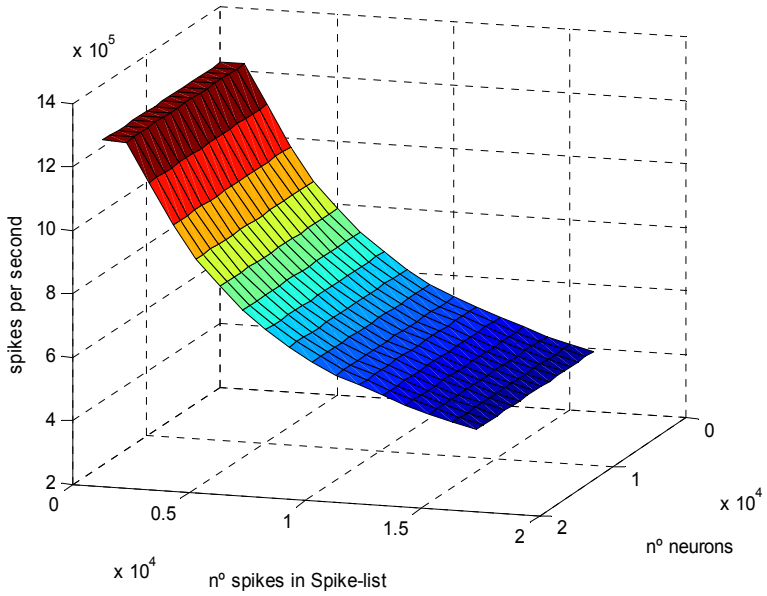


Ilustración 28: Rendimiento frente a actividad global y tamaño de la red. El rendimiento total (D_t) del sistema queda definido con la Ecuación 26, el cual es independiente del tamaño de la red e incluye un termino (A_{riesgo}) para tener en cuenta la degradación del sistema como consecuencia de los riesgos e interbloqueos [108].

$$D_t = \frac{f_{clk}}{A_{riesgo} + \max\left[27,\ N_{ciclos-búsqueda}\right]}$$

Ecuación 26

La Tabla 7 muestra el consumo de recursos hardware del módulo de búsqueda del evento de tiempo menor en función del número de bancos de EMBs utilizados. Este módulo posee un paralelismo masivo y por tanto un consumo importante de puertas lógicas del dispositivo reconfigurable. Es el elemento delimitante en el diseño. Obsérvese que el crecimiento es lineal con el número de bancos de memoria utilizados. El resto de módulos tienen un consumo mínimo en relación con éste módulo.

Nº de puertas de sistema	EMBs
148.402	2
577.758	8
2.295.005	32
4.823.495	64

Tabla 7 Consumo de recursos hardware para el árbol de búsqueda paralelo. Diseño sintetizado para una FPGA Virtex II 6000 [135].

4.5 Interfaz software

Volviendo sobre el diagrama de bloques de la Ilustración 18, donde hardware y software procesan e intercambian información durante todo el proceso de simulación, en este simulador destaca el papel de la parte software como un elemento de supervisión. En esta línea el software, a modo de herramienta, permite la monitorización a través de una interfaz gráfica que facilita la interacción entre el usuario y el simulador.

La definición de la topología de red, de los estímulos de entrada (en el caso de no utilizar un sistema físico externo) y de las características neuronales deseadas, se realizan mediante ficheros de configuración con una estructura específica. Estos ficheros se cargan durante el proceso de configuración del sistema y pueden ser utilizados por la aplicación software para mostrar, cómodamente, la topología y características de la red.

A modo de ejemplo, la Ilustración 29 muestra el aspecto de la aplicación durante un proceso de simulación. Por otro lado, la Ilustración 30 muestra la topología de una arquitectura neuronal

simplificada basada en el cerebelo biológico. Esta ilustración se obtiene a partir de los ficheros de configuración iniciales, lo que facilita en gran medida la comprensión y control de las simulaciones que se ejecutan en hardware. Para ello, la aplicación integra dentro del sistema, un software de visualización gráfica de libre distribución *"Graphviz Graph Visualization Software"* [40]. *Graphviz* permite la visualización de información de forma estructurada a través de diagramas gráficos. *Graphviz* representa la información a partir de un fichero de configuración que utiliza un lenguaje de descripción muy sencillo pero con una estructura y reglas predefinidas.
Para interconectar la aplicación software, que gestiona la interacción con el hardware y la *Graphviz,* se construyo un traductor que genera, a partir de los ficheros de definición topológica, los ficheros con el formato adecuado para la interpretación con *Graphviz.*

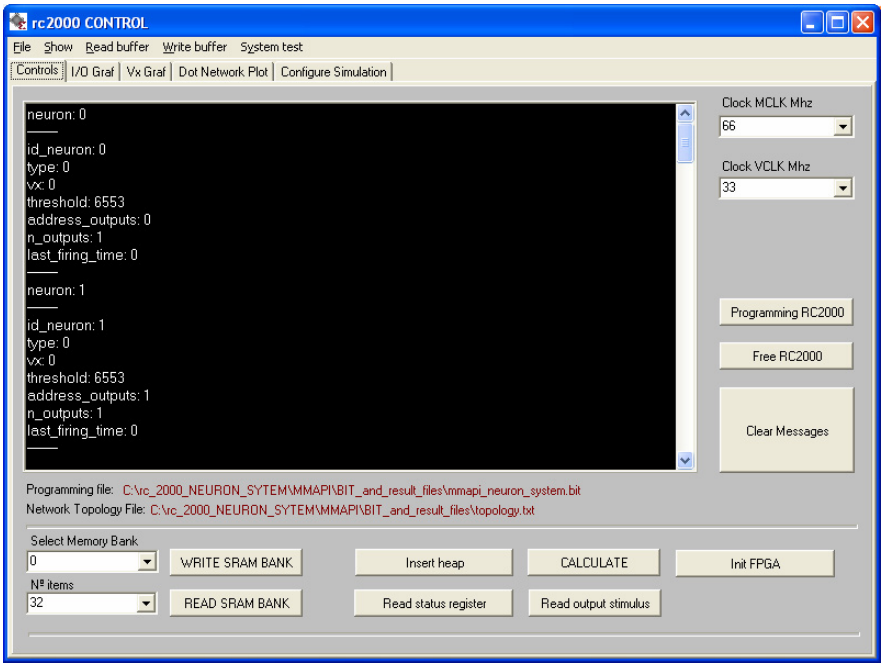


Ilustración 29: Aspecto de la aplicación para interactuar con el simulador hardware durante un instante de la ejecución donde se ha pedido el estado de las variables neuronales.

Los parámetros de configuración neuronales, tales como capacidad de membrana, constantes de tiempo, conductancias etc, han sido

predefinidos en hardware debido a que algunos cálculos computacionalmente costosos se precalculan a través de tablas. De este modo el tipo de neuronas que se pueden interconectar está limitado a nivel de software teniendo que modificar la parte hardware con la inclusión de nuevos modelos. Pero debido a que el objetivo es el estudio del cerebelo como mecanismo coordinador, resulta una opción cómoda de definición de red.

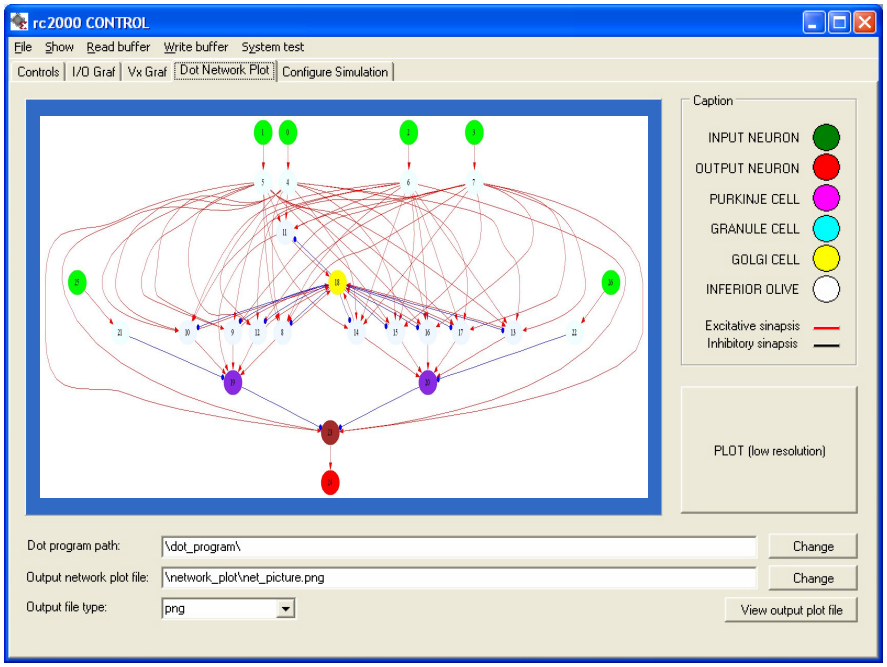


Ilustración 30: Visualización gráfica de la topología de red**.** La aplicación permite la visualización de la topología de la red a simular de forma gráfica.

4.6 Discusión

La principal innovación de éste sistema es el uso eficiente de recursos para procesamiento paralelo en un dispositivo FPGA para un simulador de neuronas de pulsos dirigido por eventos.

Se ha adoptado una estrategia para manejar de forma eficiente la lista de eventos, utilizando arquitecturas de micro-segmentado y recursos masivamente paralelos. Esta es una aproximación comple-

tamente nueva en el marco de los simuladores de redes de pulsos dirigidas por eventos.

Todo el sistema de computación neuronal está implementado con una estructura segmentada en 9 etapas donde es preciso tener en cuenta problemas de concurrencia, bloqueos y riesgos entre eventos.

La arquitectura de cauce segmentado puede ser subdividida, creando una arquitectura micro segmentada, no obstante dependiendo de la topología de la red puede que no crezca más la eficiencia debido al los riesgos entre eventos.

Aunque la comparación entre diferentes esquemas de simulación dirigida por eventos es difícil, la aproximación construida obtiene resultados similares en la práctica a otras aproximaciones software [109] pero con la peculiaridad de disponer de un consumo menor y mayor portabilidad.

El esquema de computación utilizado es muy general y puede ser adaptado para diferentes modelos neuronales, esto tiene interés en el marco de las simulaciones masivas de neuronas y los experimentos de tiempo real (como ejemplo en robótica, con experimentos de aprendizaje con esquemas que integran ciclos cerrados sensor-motor).

Capítulo 5
Sistemas empotrados de control de robots

A lo largo de este capítulo se mostrarán diferentes arquitecturas robóticas definidas como plataformas de pruebas, para el estudio y evaluación in-situ de diferentes circuitos neuronales y sistemas bio-inspirados.
En la sección 5.1 se aborda la descripción de un brazo robot bio-inspirado junto con un estudio de la viabilidad de los algoritmos genéticos como mecanismo de ajuste de las constantes de control.
En 5.2 se describe un robot humanoide comercial modificado como plataforma de test para la evaluación de aprendizaje con circuitos neuronales de eventos.
Todos ellos mantienen en común la posibilidad de ser controlados mediante circuitos neuronales implementados en hardware reconfigurable y son un mecanismo eficaz para el estudio de las interacciones con el medio en ciclo cerrado percepción-acción.

5.1 Brazo robot bio-inspirado

5.1.1 Introducción

El estudio de esquemas de control en bucle cerrado, que permiten a un sistema mantenerse estable frente a perturbaciones externas, ha dado como resultado la creación de reguladores más o menos sofisticados. Entre ellos, el más común es el controlador continuo PID (Proporcional Integral Derivativo). Ciertamente el funcionamiento de los reguladores PID, una vez ajustadas sus constantes proporcional, integral y derivativa, es bastante bueno, sin embargo el ajuste de estas constantes no es una tarea sencilla en la práctica [138] [86]. Sobre todo cuando el elemento actuador es un elemento compuesto. Piénsese en el caso de un brazo robot con cinco o seis articulaciones controladas independientemente por un regulador local para cada una. Las acciones de control de una articulación influyen irremediablemente sobre las otras produciendo sobre-oscilaciones y en general, inestabilidad del sistema. El problema se agrava aún más cuando se modifica la dinámica del robot. Por ejemplo; suponer que el brazo de seis articulaciones manipula un objeto de un determinado peso. En ese instante todos los ajustes de constantes que definían la dinámica de brazo y por consiguiente las acciones de control quedan desfasadas.

Una solución práctica para minimizar los efectos de la variación de la dinámica interna pasa por la utilización de motores sobredimensionados con respecto al par de giro que han de ejercer. De este modo cualquier perturbación del sistema es absorbida permitiendo que el brazo llegue a las posiciones de consigna. Esta solución tiene dos inconvenientes que la hacen inviable para su aplicación en sistemas autónomos generalmente propulsados por baterías.

1. Gran consumo de energía. Sistema autónomo poco eficiente energéticamente.

2. Sistemas pesados, rígidos poco ágiles y peligrosos en la interacción con humanos

Lo deseable pasa por el desarrollo de un sistema de control con una filosofía totalmente diferente. Un sistema adaptativo que aproveche la inercia y en general la dinámica interna de la estructura del robot. Una vez más, volviendo la mirada hacia la biología se observa que, en lo referente a la parte mecánica, los músculos son sistemas flexibles y que gobiernan articulaciones no rígidas. Mientras que en la parte de control se establecen sistemas adaptativos con aprendizaje de modelos. No se aplican esquemas de minimización de error en tiempo real (como esquemas PID) debido a los retardos neuronales.

En robótica existen multitud de esquemas de control adaptativo. Sin embargo, la mayoría de ellos precalculan las ganancias o parámetros de ajuste en una fase previa de diseño con modelos simulados. Los parámetros quedan fijados a intervalos de forma que se utilicen unos u otros (los más apropiados) en función de un criterio de selección, por ejemplo: el punto del espacio de trabajo donde se mueva el robot.
La implementación más convencional es la de una tabla en la cual las entradas son intervalos discretizados de las variables, que determinan la pertenencia a una región u otra del espacio de trabajo, mientras que las salidas son los valores de las ganancias en estos intervalos. Las variables más empleadas para ajustar las ganancias en el control de robots manipuladores son las variables articulares y la carga. Nótese que el procedimiento para ajustar los parámetros de los controladores PID de las articulaciones puede ser muy laborioso puesto que, en general, se necesitan ajustar *3n* parámetros K_p, K_i, K_d, siendo *n* el número de articulaciones. Por consiguiente, el número de ensayos que es necesario realizar depende de la resolución que se desee. En cualquier caso si no se dispone del modelo dinámico del sistema robótico por su enorme complejidad, es necesario realizar una gran cantidad de pruebas. Una posible solución consiste en ajustar las ganancias (parámetros del PID) de acuerdo a una ley de adaptación que intenta minimizar (normalmente el error cometido) de acuerdo con un modelo de referencia [107] El modelo de referencia es una aproximación que describe cómo se desea que se comporte el sistema mientras que la técnica de ajuste modela las constantes del controlador PID de acuerdo a un objetivo (función de optimización). Véase la Ilustración 31.

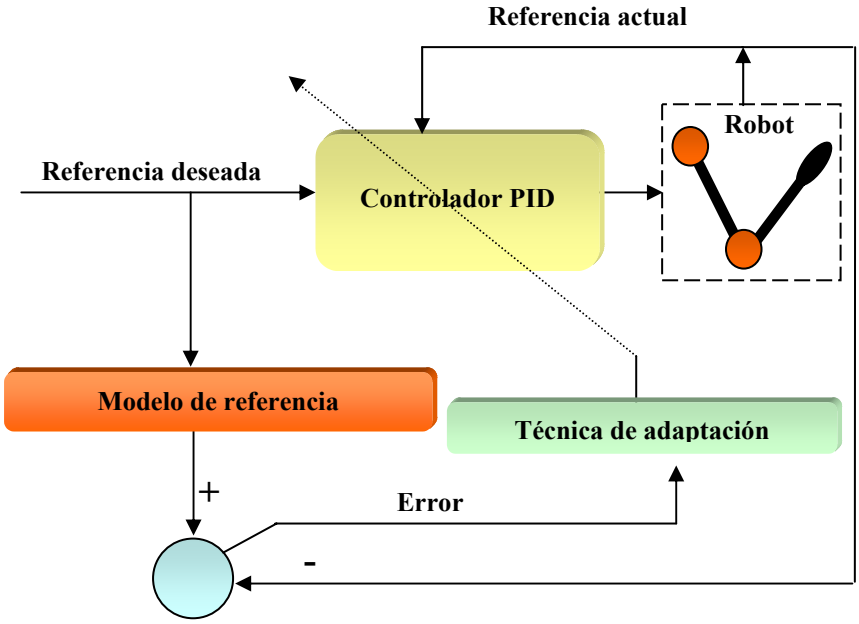


Ilustración 31: Control por modelo de referencia

En resumen dos son elementos esenciales que facilitan la versatilidad y eficiencia de un robot.

1. Una arquitectura mecánica con articulaciones flexibles

2. Un sistema de control con aprendizaje que optimice la dinámica del sistema.

Los resultados que aportan el estudio realizado con esta plataforma justifican estos dos factores. Por un lado el robot construido, posee articulaciones no rígidas (el par aplicado a cada articulación se realiza prácticamente de forma directa). Por el otro, el sistema de control analizado permite el aprendizaje de constantes dinámicas.

Una primera aproximación para conseguir un sistema de control adaptativo consiste en ajustar las constantes del controlador PID para que sean óptimas para una trayectoria predefinida, haciendo que el error cometido y el consumo de energía sean los mínimos posibles (control adaptativo dinámico).

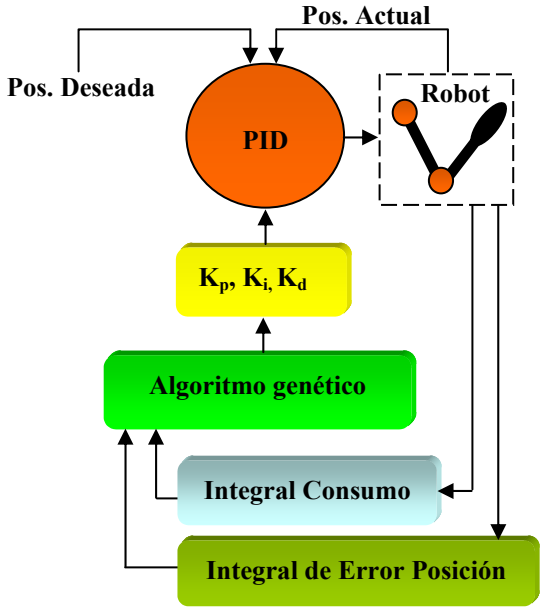


Ilustración 32: Optimización de constantes dinámicas mediante algoritmos genéticos.

Para el ajuste de estas constantes se ha utilizado un algoritmo genético simple, el cual codifica mediante individuos las constantes para los controladores PID (véase la Ilustración 32). Normalmente, los robots convencionales sólo utilizan modelos cinemáticos ya que pueden realizar tareas de control en ciclo cerrado (sensor-actuador). Pero esto significa un consumo enorme de energía que hace esta aproximación poco viable para robots autónomos. Además, si las constantes inerciales de los elementos a manipular son significativas con respecto a la fuerza que son capaces de realizar, el control será muy deficiente.
Esta característica pone de manifiesto la necesidad de estudio de otros mecanismos de control más eficientes, como es el caso de los sistemas biológicos.

5.1.2 Descripción de la plataforma de experimentación

La plataforma está compuesta por los siguientes elementos:

1. Una librería de funciones de alto nivel software de control para PC.
2. Tarjeta de estándar de hardware reconfigurable (RC200).
3. Tarjeta interfaz de control de potencia de motores.
4. Mini cámara digital a color.
5. Sistema electromecánico bio-inspirado de dos grados de libertad (véase la Ilustración 33 y la Ilustración 34).

La Tabla 8 muestra, a modo de avance, el volumen de ocupación y la velocidad de cada unidad integrada en el chip FPGA.

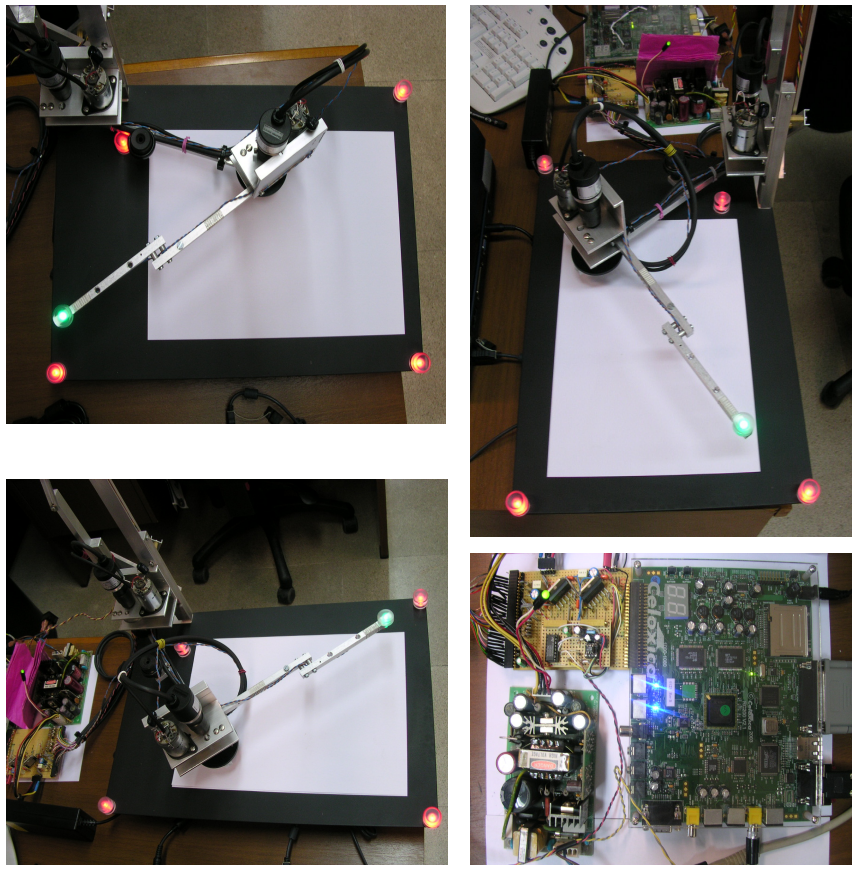


Ilustración 33: Aspecto de la plataforma robótica formada por un brazo robot de dos grados de libertad y articulaciones no rígidas. Obsérvese (parte inferior derecha) la tarjeta interfaz desarrollada a medida para interconectar el dispositivo reconfigurable con el robot. Los cilindros luminosos son para localizar la posición del brazo en el caso de utilizar el sistema de visión [3].

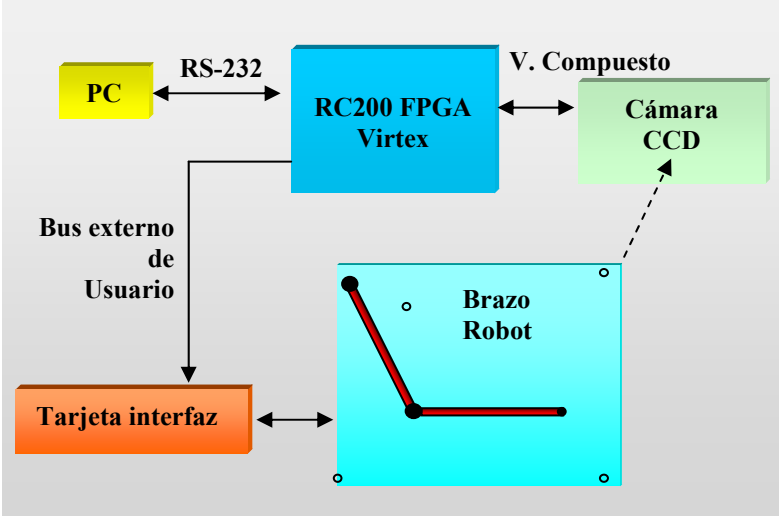


Ilustración 34: Arquitectura general de la plataforma robotizada. Se observa, a nivel modular, los diferentes bloques físicos que lo componen. La cámara emite imágenes en video compuesto.

Módulos	% Consumo Slices	Frecuencia Mhz
Comunicaciones	**5.64**	**139.25**
Lector de posición	**0.97**	**127.68**
Controlador PID	**0.99**	**132.22**
Ajuste de cero	**0.97**	**127.68**
Generador PWM	**3.78**	**88.37**
Lógica de apoyo (*)	**1.85**	**121.62**
TOTAL	**37.9**	**33.82**

Tabla 8: Ocupación del diseño utilizando un chip FPGA (Spartan II). Datos obtenidos realizando compilaciones parciales. Total de Slices del dispositivo reconfigurable: 5120. (*) Lógica de test para mostrar estados (Led, display etc).

5.1.3 Tarjeta interfaz

La tarjeta interfaz realiza la tarea de interconexión física entre el brazo robot (sensores y actuadores) con la plataforma RC200. Esta tarjeta está realizada a medida y consta de dos controladores de potencia LMD18200 de *National Semiconductors* [84] para dirigir los dos motores del brazo (hombro y codo), además realiza la conversión de niveles lógicos TTL proporcionados por los sensores de posición ópticos a niveles de 3.3 V compatibles con la FPGA.

Estos circuitos disponen de tres entradas de control (Dirección, Freno y señal PWM) y han sido especialmente desarrollados por el fabricante para el manejo de motores mediante PWM *(Pulse Width Modulation)* en modo unipolar (signo-magnitud) o en modo antifase *(Locked Antiphase PWM)*. Las tres entradas son controladas directamente por patillas auxiliares de la FPGA puesto que los niveles lógicos admisibles por este componente permiten trabajar a 3.3 V. La entrada de frenado tiene una característica especial y es que permite un frenado regenerativo si se combina con las otras dos señales [14]. Esto es, en el caso de hacer funcionar el sistema con baterías la energía producida por el motor durante su frenado pasaría a cargar las baterías permitiendo un ahorro en el consumo.

5.1.4 Sensores de posición

Los sensores de posición son de tipo incremental de 1024 pasos por revolución y señal de ajuste de cero. Estos sensores disponen de cinco salidas de las cuales se utilizan solamente tres porque dos de ellas son señales complementadas. Estas señales son: Señal de paso por cero (CHN), señal cuadrada de paso (CHA), señal cuadrada de paso desfasada 90 grados con respecto de la anterior (CHB) (Véase la Ilustración 35 y la Ilustración 36). Todas estas señales son cuadradas y producen un cambio de nivel de tensión por cada paso.

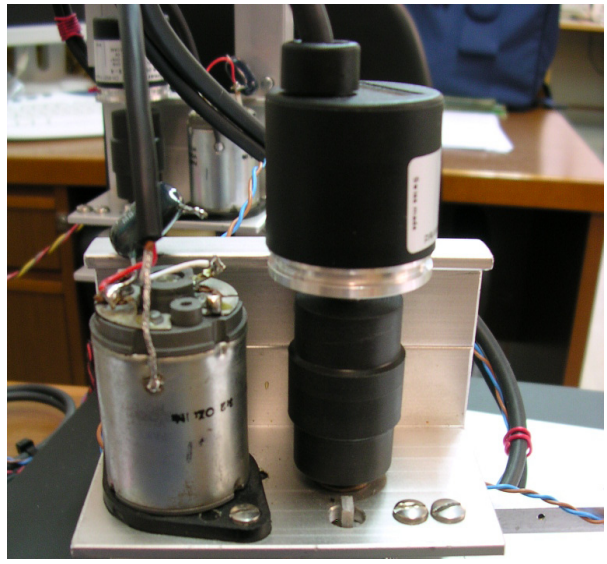


Ilustración 35: Conjunto motor y sensor de posición de una de las articulaciones del brazo robot.

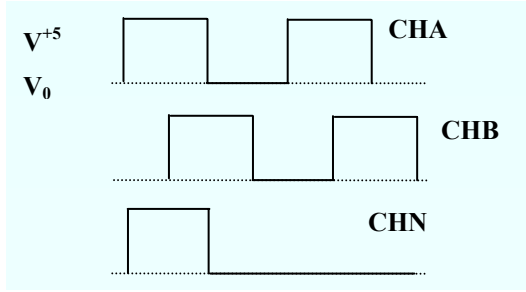


Ilustración 36: Señales de posición del codificador. CHA (canal A), CHB (canal B) igual que la señal (A) pero desfasada 90 grados, CHN (señal de paso por cero que avanza el eje del codificador. Puesto que la señal (CHB) entregada esta desfasada 90 grados con respecto a (CHA) es posible multiplicar la resolución del dispositivo por cuatro si se utilizan los flancos de las señales. De este modo se puede alcanzar una resolución de 4096 pasos por revolución en vez de los 1024. Además es posible conocer el sentido de giro del eje, simplemente viendo la secuencia de cambios de estado en la señal.

5.1.5 Plataforma reconfigurable RC200

El sistema reconfigurable en el quel se han implementado todos los controladores ha sido la RC200 de Celóxica [12] que in-

corpora una FPGA Virtex-II [135]. Esta plataforma de desarrollo cuenta con diversos periféricos de entrada/salida. Entre ellos se distinguen E/S de vídeo compuesto, salida RGB, puerto serie, puerto paralelo, Leds, visor de siete segmentos, etc. Además de un conector de expansión con pines conectados a patillas de la FPGA disponibles para el usuario y que han sido utilizados para la conexión con el robot.

5.1.6 Cámara para visión

Se trata de un elemento auxiliar, incorporado en el sistema para hacerlo compatible con otros experimentos relacionados con visión fuera de la línea de estudio de este capítulo.
La cámara digital CCD en color tiene una resolución máxima de 628 x 582 puntos a la cual se le ha añadido un filtro polarizador ajustable que permite disminuir la saturación en presencia de un grado alto de intensidad de luz ambiente.

5.1.7 Librería de control

En el PC se ha desarrollado una librería de funciones específica para la comunicación a través de puerto serie con la placa de control RC200. Esta librería permite activar y desactivar los controladores locales, mandar y recibir posiciones de las articulaciones del brazo, mover independientemente cada motor y ajustar las constantes de los controladores. La Tabla 9 resume el conjunto y funcionalidad de la librería implementada.

Funciones	Utilidad
hardware_init()	Inicia la comunicación con el hardware estableciendo una configuración inicial segura
hardware_end()	Desconecta la plataforma liberando todos los recursos requeridos previamente
get_pos()	Recibe la posición actual de la articulación seleccionada previamente con una dirección.
set_dir()	Establece la dirección de la articulación con la que se va a interactuar.
set_pos()	Envía una posición de consigna a la articulación seleccionada previamente con una dirección.
calibrate()	Mueve el brazo hacia una posición segura de calibración.
act_pid()	Conecta el controlador PID asociado con la dirección de la articulación previamente establecida.
des_pid()	Desconecta el controlador PID
set_kp()	Configura la constante proporcional del controlador PID (dirección de la articulación previamente establecida).
set_ki()	Configura la constante integral.
set_kd()	Configura la constante derivativa.
set_frec_pid()	Configura la frecuencia de muestreo del controlador PID correspondiente.

Tabla 9: Conjunto de funciones de control desarrolladas para la plataforma robótica incluidas en la librería de control software.

5.1.8 Implementación Hardware

El diseño presentado ha sido definido mediante un lenguaje de descripción hardware (HDL) de alto nivel (Handel-C). La arquitectura interna del diseño consta de seis módulos (véase la Ilustración 37). Todos los módulos funcionan en paralelo y se sincronizan mediante canales (canales Handel-C de Celoxica) [12] para obtener un rendimiento y sincronización óptima. De este modo se consigue procesamiento y respuesta del sistema en tiempo real.

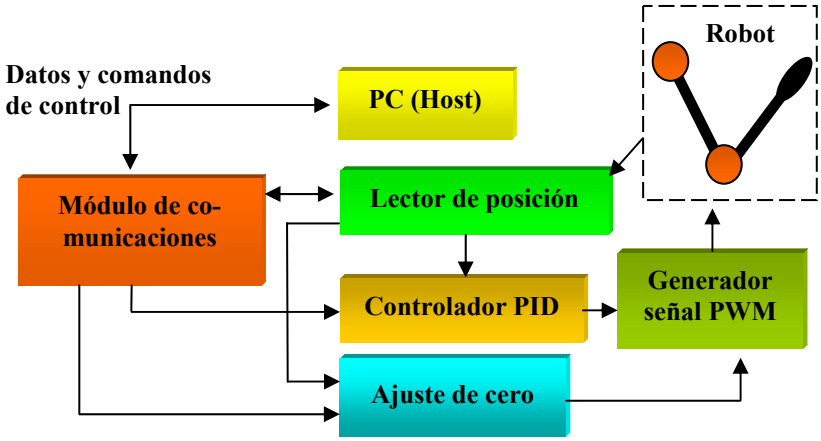


Ilustración 37: Arquitectura interna del sistema. Se observan seis módulos funcionales trabajando en paralelo sincronizados mediante canales para procesamiento en tiempo real.

5.1.9 Módulo de comunicaciones

El módulo de comunicaciones transmite y recibe comandos predefinidos desde el PC. El conjunto de comandos codifican acciones de la FPGA, tales como, indicar una posición destino al controlador PID para una articulación, realizar una calibración de sensores etc.
Para la comunicación se utiliza el puerto serie RS-232 a una velocidad 9600 bps. Velocidad suficiente dado que la información que se transmite es mínima, dado que todo el control está empotrado en la FPGA. El envío y recepción de datos utiliza un protocolo de comunicaciones de diseño propio permitiendo recibir y enviar tramas de datos y tramas de control libres de errores. Para conseguir esto se numeran las tramas, se establecen comandos de confirmación ACK y control de paridad a nivel físico.

5.1.10 Lector de posición

El módulo lector de posición calcula de forma continua las posiciones de las articulaciones mediante la comprobación del estado de los canales de los sensores de posición. El proceso es el si-

guiente: Primeramente se detecta un cambio en cualquiera de los canales CHA o CHB obteniéndose un patrón (00, 10, 11 ó 01). Posteriormente el valor obtenido se compara con el patrón almacenado de una lectura anterior permitiendo conocer el sentido de giro de la articulación. Si el sentido es horario, el sistema incrementa una variable que almacena la cuenta de pasos del eje. Por el contrario, si el sentido de giro es anti-horario se decrementa dicha variable. El resultado es un valor que codifica la posición de cada eje de giro. Cada sensor de posición tiene una resolución de 1024 ppr (pasos por revolución), como el mecanismo se basa en la detección de flancos (CHA, CHB), la resolución total del dispositivo es de 4096 ppr lo que supone aproximadamente 0.087 grados de precisión.

5.1.11 Controlador PID

Este módulo se encarga de mantener continuamente la posición deseada de las articulaciones del brazo. El controlador PID se ha empotrado en hardware en el chip FPGA en vez de en el PC. De este modo se minimiza el tráfico de información. El PC solamente realiza tareas de alto nivel (enviar posiciones de consigna, enviar parámetros de ajuste, etc) mientras que la FPGA realiza todas las tareas que requieren procesamiento en tiempo real (generación de señal PWM, lectura continua de sensores, etc).

El controlador digital se implementa físicamente como un algoritmo programado en la FPGA, que obtiene la secuencia de valores *u(k)* (secuencias de valores k-ésimos de acción para el motor) a partir de la secuencia de valores *e(k)* (errores de posición k-ésimos de la articulación). Una posibilidad para obtener la representación discreta del controlador PID programable [86] [63]consiste en partir de la ecuación diferencial continua (véase la Ecuación 27) y obtener una ecuación en diferencias discreta entre la señal *e(k)* y la señal *u(k)*. La forma discreta cosiste en aproximar la ecuación diferencial continua calculando *u(kT)* a partir de los valores *e(t)* (error de posición: diferencia entre la posición deseada y la actual del eje en el instante t) en los instantes de muestreo [*e(T), e(2T)*…].

$$u(t) = K_p\left(e(t) + T_d \frac{de}{dt} + \frac{1}{T_i}\int_0^t e(\tau)d\tau \right)$$

Ecuación 27

Operando y sustituyendo los términos integral y derivativo en la Ecuación 27 por la aproximación más sencilla para la integral y la derivada (Ecuación 28) resulta la expresión discreta del controlador (Ecuación 29).

$$\int_0^{kT} e(t)dt \approx \sum_{j=0}^{k-1} e(jT)\cdot T \qquad \frac{de(kT)}{dt} \approx \frac{e(kT) - e((k-1)T)}{T}$$

Ecuación 28

Obsérvese que en la Ecuación 29 aparecen las constantes q_0, q_1 y q_2 cuyo valor depende de los términos proporcional, integral y derivativo (Ecuación 30) que definen la dinámica de funcionamiento del controlador.

$$q_0 = k_p\left(1 + \frac{T_d}{T}\right) \qquad q_1 = k_p\left(-1 + \frac{T}{T_i} - 2\frac{T_d}{T}\right) \qquad q_p = k_p \frac{T_d}{T}$$

Ecuación 29

$$u_k = u_{k-1} + q_0 e_k + q_1 e_{k-1} + q_2 e_{k-2}$$

Ecuación 30

El ajuste de estos términos no es nada trivial porque dependen de las características mecánicas del dispositivo a controlar (peso, inercias, etc) [138] [107]. El ajuste de estos parámetros se estudia más adelante mediante una aproximación genética del problema.

La implementación del controlador digital en la FPGA, donde la disponibilidad de unidades de punto flotante es un recurso costoso, pasa por un escalado previo de valores que sitúe la respuesta del controlador en un rango de valores enteros suficientemente grande para que no se sature.

El proceso de control de la posición de una articulación consiste en ejecutar periódicamente el código del controlador y realizar las siguientes operaciones: Leer el valor de la entrada (posición del brazo), Calcular la acción de control según la ecuación del controlador, actualizar la salida (activar los motores del brazo). Generalmente este proceso se asocia a una interrupción de un temporizador que en el contexto de hardware reconfigurable será un registro contador incrementado cada ciclo de reloj. Un punto importante en la implementación es la comparación de la acción de control con los valores máximo y mínimo permitidos. Esta comparación es necesaria para evitar el efecto de anti-reset "*wind-up"* del integrador. Si la acción de control se sale del rango admisible (el controlador se satura) y no se limita su valor en el programa, ésta puede crecer mucho sin verse reflejado en una acción sobre el proceso (por la saturación del actuador), lo que se traduce en una sobre-oscilación.

5.1.12 Ajuste de cero

Dado que los sensores de posición son de tipo incremental, es preciso conocer a partir de qué posición de cada eje se comienzan a contar los pasos para que la medida sea correcta. Para ello los sensores disponen de una señal (CHN) que se pone en alta cuando están alineados. El módulo de ajuste activa los motores hasta la detección de la señal de cero y hace un reset sobre los registros de posición. A partir de ese instante el sistema está calibrado y puede empezar a trabajar.

5.1.13 Generador de señal PWM

El objetivo de esté módulo consiste en mantener una señal cuadrada de frecuencia constante a 20Khz y tiempos en estado alto y bajo variables. La arquitectura de este módulo es básicamente un conjunto de registros funcionando a modo de contadores. Cada contador definirá un periodo de tiempo en el que la señal de salida permanecerá en un estado (alto o bajo). Conociendo la frecuencia de reloj principal (50 Mhz en esta aplicación) se establecen los valores de dichos contadores para la generación de la señal PWM a la frecuencia deseada. La estrategia de generación de señal PWM elegida

para el robot ha sido la de anti-fase. En este modo de funcionamiento la polaridad aplicada al motor cambia con el tiempo a una frecuencia de 20Khz (véase la Ilustración 38). Para producir este comportamiento se envía al controlador de potencia una señal cuadrada de frecuencia constante pero de tiempo en alta y baja variable. Si el tiempo que está la señal en estado alto es superior al tiempo en estado bajo el motor girará en un sentido (con una velocidad y par proporcional al tiempo en alta diferencial), mientras que si es al contrario girará en sentido opuesto. En el caso de ser iguales el motor permanecerá inmóvil.

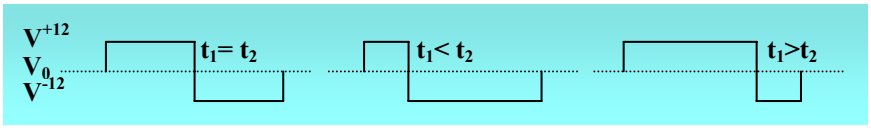


Ilustración 38 Señal PWM en anti-fase. En cada ciclo de control se cambia la polaridad de alimentación del motor permitiendo parar, establecer el sentido de giro y la velocidad del mismo mediante la modificación del tiempo (t) de la señal.

5.1.14 Algoritmo genético

La funcionalidad del algoritmo genético consiste en generar las constantes Kp, Ki y Kd del controlador PID y mejorarlas, en cada iteración, intentando minimizar el consumo y el error de posición integrado de una trayectoria. Por tanto, las funciones objetivo a minimizar son el *error de posición,* y el *consumo*. Esto trae consigo un problema a la hora de asignar una valoración concreta a un individuo "*fitness*" porque una solución con consumo próximo a cero y error medio pequeño puede ser etiquetada como óptima y sin embargo el robot no se mueve. La solución pasa por establecer una política multiobjetivo para el algoritmo o bien definir la funcion objetivo filtrando estas inconveniencias. En este estudio se ha adoptado esta última opción.

La población está formada por un conjunto de individuos costituidos por los tres parámetros Kp, Ki, Kd, que consforman los cromosomas. Los individuos se cruzan entre sí en función de sus cualidades *"fitness"* e intercambian su material genético. Para ello se ha utilizado un algoritmo genético sencillo [48].

Al tener que realizar "movimientos reales" en cada iteración para cada individuo, la evolución del sistema es lenta. El cruce se realiza generando un valor aleatorio que indicará qué trozo de material genético se intercambia. Por ejemplo; Supongamos que un individuo es el 0101 y el siguiente mejor es el 00010. Ahora suponegamos que el valor aleatorio de cruce es el 0100; entonces daría como resultado dos nuevos indivíduos, que son 0001 y el 0110. Utilizando este esquema de cruce es obvio que solamente los n/2 individuos mejores se cruzan dando lugar a otros n/2 nuevos individuos, y los n/2 peores de la lista inicial, se descartan. Para comenzar la competición se genera una población de individuos (en este caso 8) con valores de cromosomas aleatorios pero dentro de un rango máximo admisible. Esto es; max para Kp =1024, max para Ki 50, max para Kd = 1024. Posteriormente se siguen los siguientes pasos que se repiten un número de veces predefinido (generaciones):

1. Evaluar las cualidades de cada individuo (selección natural). Esto consiste en configurar el controlador PID con los cromosomas de cada individuo y evaluar la trayectoria previamente definida midiendo el consumo y error totales. Esta evaluación supone la activación del robot para realizar la trayectoria.

2. Ordenar los individuos según la puntuación obtenida (función de minimización de error de posición y consumo). Esto consiste en ordenar los individuos de mayor a menor "fitness" utilizando el algoritmo *"Quick Short"* [45] y permitir reproducirse solamente a los primeros de la lista.

3. Permitir a cada uno de los individuos reproducirse (intercambien material genético) de acuerdo con su puntuación *"fitness"*. Además se permite con una probabilidad (P) que alguno de los bits de un gen se vea alterado espontáneamente.

5.1.15 Proceso de evaluación de individuos

El proceso de evaluación parte con una trayectoria predefinida formada por un conjunto de posiciones deseadas obtenidas mediante programación gestual. Seguidamente se ajustan

las constantes del PID con los valores del individuo a evaluar y se comienza el envio de coordenadas objetivo hacia el robot a intervalos de 10 ms. En cada envio se lee del robot el consumo y el error de posición de la coordenada enviada en el instante anterior (inicialmente 0).

Para calcular el *error*, Ecuación 31, y el *consumo*, Ecuación 32, se calculan las integrales discretas a lo largo de toda la trayectoria.

$$Err = \int_0^{kT} c(t)\,dt \approx \sum_{j=1}^{k-1} \frac{c(j)\cdot c(j-1)}{2}$$

Ecuación 31

$$Con = \int_0^{kT} e(t)\,dt \approx \sum_{j=1}^{k-1} \frac{|e(j)|\cdot|e(j-1)|}{2}$$

Ecuación 32

Con estas dos magnitudes, el algoritmo genético evalua la calidad de la solución y establece la clasificación entre individuos peores y mejores de acuerdo a la función de minimización.

5.1.16 Funciones de evaluación de error y consumo

Para minimizar el error y el consumo se han evaluado tres funciones distintas (A, B, C) que se ilustran seguidamente.

Función de evaluación "fitness" (A)

La primera función de evaluación consta de dos partes ponderadas (Ecuación 33): La integral del error *Err* y la integral del consumo *Con*. Estas dos partes se ponderan y suman para dar una valoración final a un individuo, que el algoritmo intentará hacer máxima (en este caso un individuo es mejor cuanto menor es el valor del error y el consumo). Mediante el ajuste de los valores de ponderación (W) se puede dar mayor o menor importancia al error o al consumo de la trayectoria.

$$F = 1 - (w \cdot Con + (1 - w) \cdot Err)$$

Ecuación 33

La utilización de esta función de evaluación conlleva implícitamente la posibilidad de soluciones óptimas en lo referente al proceso de búsqueda del algoritmo genético, pero inútiles en la práctica. Porque es posible valorar como buena una solución con un consumo muy próximo a cero y un error aceptable (suponer que el brazo se encuentra en un punto medio de una trayectoria circular). Esta solución haría que el robot no se moviese porque no hay un consumo suficiente. Por consiguiente este tipo de soluciones no son buenas ya que interesa que la valoración de la solución quede comprendida lo mas cerca posible del máximo absoluto pero compensada entre ambos parámetros (error y consumo). Es decir, la solución óptima estará en un intervalo que establezca un compromiso entre un consumo y un error admisible del plano de soluciones posibles.

Función de evaluación "fitness" (B)

En esta función de evaluación se intenta solucionar el problema comentado anteriormente, mediante la definición de un escalón en la frontera de valores admisibles. Este escalón establece un umbral mínimo para el consumo. Todas las soluciones que tengan un consumo inferior a cierto umbral serán penalizadas. La elección del umbral conlleva, nuevamente, un compromiso puesto que si la trayectoria predefinida se puede realizar con un consumo menor el algoritmo intentará alcanzar dicho consumo y penalizará una solución que a priori era mejor. El valor óptimo del umbral de consumo depende por tanto de la trayectoria predefinida y de la dinámica del robot (supóngase que coge algún peso). Como estos parámetros son a priori desconocidos, una solución práctica consiste en ajustar el umbral de consumo a un valor lo suficientemente pequeño como para permitir que el robot se mueva sin carga (sin coger pesos) y, de este modo, no tener el problema descrito anteriormente.

A continuación se muestra una posible implementación sencilla del algoritmo de evaluación:

si consumo < 250 mA
entonces fitness = 0.0;
si no
entonces fitness =F(Con)+F(Err);

Las funciones "*fitness*" descritas en la Ecuación 34 y la Ecuación 35 son de tipo exponencial. De este modo se obtiene una mayor velocidad de convergencia hacia la solución.

$$F(Con) = e^{\left(\frac{-6*Con(n)}{2*pos.size} + \log(0.5) + 0.75\right)}$$

Ecuación 34

$$F(Err) = e^{\left(\frac{-6*Err(n)}{4096*pos.size} + \log(0.5)\right)}$$

Ecuación 35

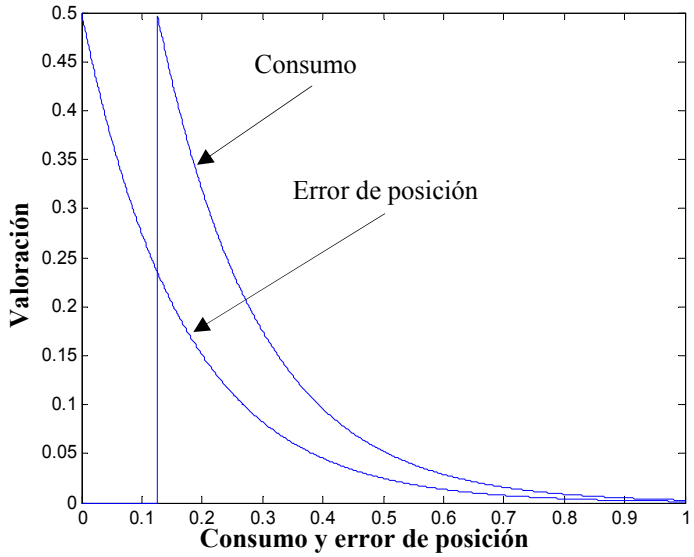


Ilustración 39: Funciones de evaluación; consumo y error de posición normalizado [0, 1]

Esto es crítico ya que el tiempo de evaluación es largo cuando es necesario ejecutar movimientos reales con robots. Además la forma exponencial permite una mayor flexibilidad en la valoración de los

individuos en etapas iniciales del algoritmo. De esta forma se puede elegir secuencias de soluciones a priori poco prometedoras pero buenas a largo plazo. En etapas finales, cuando el conjunto de las soluciones son más o menos parecidos, la discriminación entre individuos es mucho mayor (gracias a la pendiente exponencial del plano de soluciones) permitiendo una tendencia más elitista del algoritmo.
Obsérvese que tanto en *F(Err)* como en *F(Con)* (Ecuación 34 y Ecuación 35), se realiza una operación de escalado para que los valores introducidos en la función exponencial estén comprendidos entre 0 y 1. Este escalado se hace para el caso de la integral del consumo dividiendo por *2 * nº de puntos* de la trayectoria. El valor 2 es porque el consumo máximo es de 2 amperios.

Para el caso del error se divide por *4096 * nº de puntos* porque el error máximo posible es de 4096ppr (Pasos Por Revolución). La ponderación para el consumo y el error esta integrada implícitamente en las propias funciones de valoración. Ambas funciones dan como resultado valores comprendidos en el intervalo [0, 0.5] de forma que al sumarlas darán valoraciones entre 0 y 1. Véase Ilustración 39.

Función de evaluación "fitness" (C)

La función "fitnes" para este caso es igual a la función (B) solamente que ahora la valoración de un individuo para el consumo se hace igual a 0.5 si está comprendido entre 250 y 500mA. Esta característica da mayor flexibilidad permitiendo mantener buenos individuos que producen trayectorias con consumos medios dejando la elección final del mejor individuo a la otra componente de la función, el error de posición.

La modificación del algoritmo se ilustra en las siguientes líneas de código:

```
Si 250 mA <= consumo <= 500 mA
 entonces fitness = 0.5;
Si no
 entonces fitness =(Con)+F(Err);
```

5.1.17 Resultados experimentales

Los experimentos realizados con las diferentes funciones de evaluación fueron tres y se corresponden con las funciones mostradas en la Ilustración 40 y en la Ilustración 41.
Los parámetros de configuración utilizados para todos los experimentos fueron:

- Nº individuos: 8
- Nº de generaciones: 80
- Trayectoria: “dibujar un 9” de 149 puntos
- Periodo de muestreo: 10ms

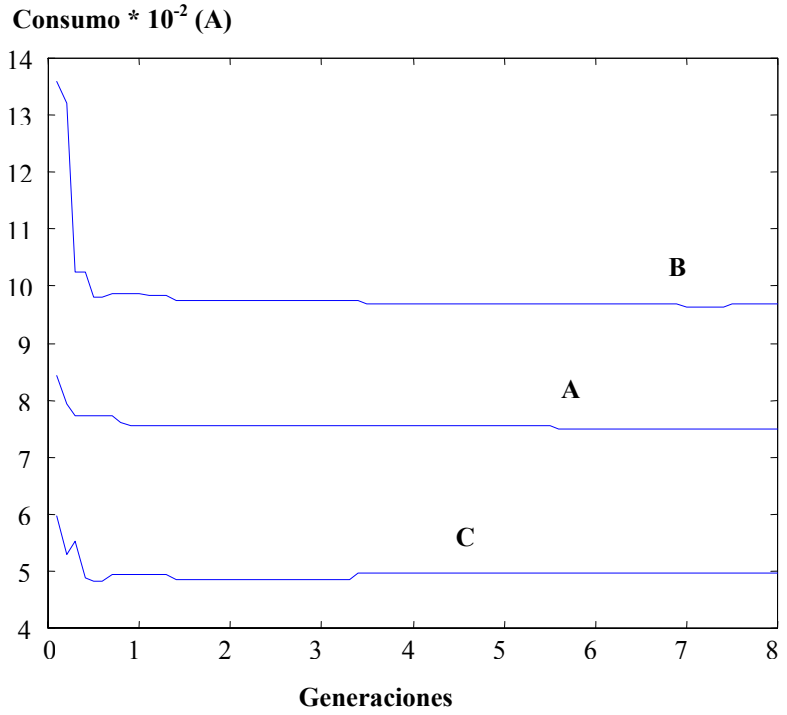


Ilustración 40: Curvas de consumo obtenidas con las funciones de evaluación A, B, C.

Obsérvese que el número de individuos e iteraciones es muy reducido ya que el proceso de evolución es muy lento. Las curvas de consumo acumulado para la trayectoria definida se ilustran en la Ilustración 40. En ella se observa la evolución del algoritmo genéti-

co a lo largo de 80 generaciones para las tres funciones de evaluación. El consumo mínimo acumulado en un ajuste manual de constantes [138] fue de $43{,}1 \cdot 10^{-2}$ Amperios mientras que el consumo mínimo obtenido por el algoritmo fue aproximadamente de $50 \cdot 10^{-2}$ Amperios. Esta diferencia fue compensada por el error de posición.

La Ilustración 41, muestra la evolución del error de posición acumulado obtenido con las tres funciones de evaluación comentadas. El error mínimo acumulado obtenido mediante ajuste manual de constantes fue de 3450 ppr (pasos por revolución) siguiendo la técnica de ajuste de [138], mientras que el error acumulado obtenido por el algoritmo fue de 200 ppr.

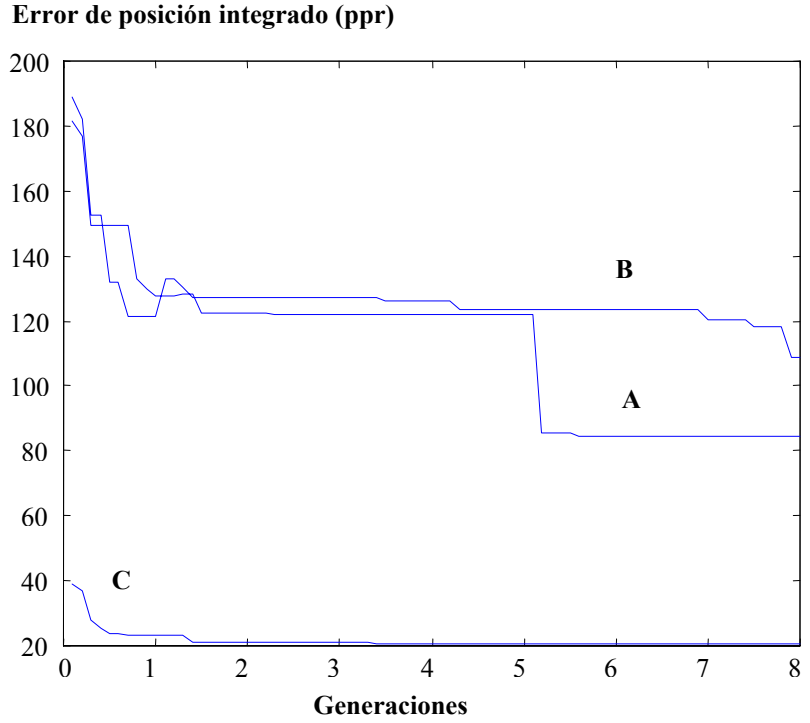


Ilustración 41: Curvas de error obtenidas con las funciones de evaluación A, B, C.

5.1.18 Discusión

El objetivo inicial de esta experiencia ha consistido en el estudio de un sistema de control con aprendizaje dinámico mediante

un proceso evolutivo, que sirve de base de estudio, para sistemas de control adaptativo futuros. Además el brazo robótico construido ha servido hasta ahora para ensayar múltiples sistemas de control bio-inspirados que incluyen aprendizaje [7] [103] [109].

Sobre este estudio se pueden resumir las siguientes conclusiones:

1. El proceso evolutivo es lento y costoso para sistemas físicos porque los motores se calientan al tener que evaluar trayectorias de individuos malos. Quizás un sistema simulado pueda ser útil para la obtención de constantes que más tarde puedan ser afinadas en el modelo real. Además la velocidad de convergencia de la función de evaluación es importante.

2. La definición de la función de evaluación es un punto crítico en cuanto a los umbrales de consumo y elección de pesos para los distintos objetivos. Esto es debido a que el robot tiene multitud de parámetros electromecánicos difíciles reflejar explícitamente en la función de evaluación.

3. Los experimentos realizados con el robot a priori muestran que solamente con una magnitud, como por ejemplo el error de posición, no se obtienen soluciones buenas para las constantes del controlador. Por ejemplo la constante integral está íntimamente relacionada con el consumo y es preciso disponer de algún parámetro en la función de evaluación que implique esta magnitud.

4. Variación en las mediciones de los parámetros de error y consumo con la temperatura. Un mismo individuo puede ser evaluado de forma diferente como consecuencia de modificaciones de rendimiento ocasionadas por variaciones térmicas en los motores. Este tipo de factores son difíciles de simular y justifican el trabajo experimental con un robot real.

5. Se ha observado que entre 20 y 80 generaciones (dependiendo de la función de evaluación) para una población inicial de 8 individuos se obtiene un ajuste más fino que el realizado manualmente.

5.2 Robot Humanoide

En esta sección se resume la arquitectura y características de funcionamiento de un robot humanoide que ha sido adaptado para ser controlado mediante el simulador de redes neuronales de pulsos descrito en el Capítulo 4.

5.2.1 Introducción

La utilización de robots Humanoides y más concretamente el desplazamiento en bipedestación dinámica, supone un reto en la utilización de sistemas de control tradicionales para robots, debido a la alta complejidad dinámica del sistema.
Recordemos que se habla de *equilibrio estático* en bípedos cuando todas las fuerzas que actúan sobre el sistema están equilibradas, de tal forma que se mantiene la posición deseada sea cual sea. Por el contrario se habla de *equilibrio dinámico* cuando el robot es capaz de avanzar según un movimiento manteniendo el conjunto de fuerzas que actúan sobre él en un intervalo controlado.

Entre otras dificultades del diseño del sistema de control, hay que tener en cuenta las siguientes características:

- El elemento a desplazar es el propio robot que se transporta así mismo y por tanto no existe un sistema de referencia fijo con el entorno.
- El estado de reposo (erguido) es una posición parcialmente inestable.
- El único contacto con el suelo se realiza en dos puntos como máximo.
- El torso tiene mayor inercia y masa que las piernas.

Estas características unidas a la interacción con el entorno exige el cálculo en tiempo real de los parámetros de movimiento (centro de masas, tensor de inercia etc.). El cálculo de estos parámetros para la obtención de un modelo dinámico del robot no es una tarea sencilla en general. Sin embargo, la biología ha solucionado este problema implementando un sistema de control con aprendizaje. Aunque si bien es cierto que la estructura muscular y las características mecá-

nicas de los músculos influyen notablemente en la simplificación de esta tarea.
Los sistemas biológicos entrenan y adaptan la dinámica corporal desde la infancia, comenzando con movimientos aparentemente caóticos para la extracción de modelos "extracción emergente de conocimiento" (pensemos por ejemplo en los movimientos de los brazos y piernas de un bebe en etapas tempranas de su existencia), hasta conseguir movimientos coordinados. La biología no codifica un único modelo dinámico corporal sino un número muy elevado de modelos "seleccionables" en función de los requerimientos del entorno que se precisen en cada instante. Pensemos por ejemplo en la actitud inherente e instintiva de una persona que coge una botella de vidrio transparente. Si la botella está llena automáticamente se elige un modelo dinámico diferente a si la botella está bacía, dado que las fuerzas y aceleraciones aplicadas van a ser diferentes en cada caso.

El aprendizaje de modelos no sólo es inherente a la dinámica del movimiento autónomo sino que puede ser extrapolado y utilizado a otros niveles de más alto grado cognitivo. Pensemos por ejemplo en un simple juego de ordenador utilizado comúnmente en la realización de exámenes psicotécnicos para conductores. En este juego el examinado ha de mantener un coche, generalmente representado por un trazo vertical, sobre dos líneas sinuosas paralelas a modo de carretera. La velocidad de desplazamiento vertical es cambiante así como la sinuosidad de las líneas. En esta situación el examinado aprende el control de la dinámica del sistema sin necesidad de sentir físicamente las perturbaciones (cambios en la sinuosidad de la carretera, cambio de velocidad etc.) del modelo real. Probablemente, y a modo de hipótesis, el examinado asocia inicialmente algún modelo dinámico aprendido previamente con características parecidas al cual se le va dando forma según se desarrolla el proceso de aprendizaje, utilizando la visión como elemento de realimentación.

Uno de los centros nerviosos dedicados y sin duda con mayor relevancia en realizar esta tarea, es el cerebelo. El cerebelo cuenta con la mitad de todas las células nerviosas que componen un cerebro humano que es de aproximadamente 100.000 millones de neuronas (se alcanzan cifras de 220.000 millones en un cerebro completo incluyendo cerebelo de un recién nacido). En él se codifican y almacenan multitud de modelos dinámicos o puede que únicamente

unos pocos básicos, aprendidos durante la infancia, con miles de parámetros de configuración que los hacen adaptables en cada situación. El sistema siempre está aprendiendo y adaptando modelos durante la vida del individuo, desde la infancia donde la respuesta motora es rápida y enérgica hasta la vejez donde la respuesta motora es lenta y pesada adaptándose de forma automática a cada nueva situación.
Girando la mirada hacia la tecnología, sin olvidar que nuestro objetivo es imitar a la biología en mayor o menor medida en tareas que ha resuelto eficientemente, resulta que la construcción de un cerebelo sintético de estas dimensiones no es posible con los medios tecnológicos actuales. Esto es así debido a limitaciones tanto a nivel espacial (conectividad inter-neuronal) como a nivel de respuesta temporal, debido a la capacidad de cómputo paralelo masivo. No obstante, tal y como se ha demostrado a lo largo del desarrollo de la presente Tesis, es posible construir una aproximación escalada de su funcionamiento.

Con esta idea en mente, el objetivo que se ha perseguido con la utilización de un robot humanoide ha consistido en la extracción de modelos dinámicos a partir del movimiento (extracción emergente de conocimiento) utilizando como arquitectura de procesamiento un cerebelo sintético. En esta línea se persigue conseguir un sistema universal de aprendizaje de modelos dinámicos, válido para cualquier robot tras un periodo deseablemente pequeño de aprendizaje, en vez de obtener un modelo dinámico exacto y fijo de un robot concreto.

5.2.2 Descripción física del sistema

La plataforma robótica utilizada, es una plataforma comercial ROBONOVA-I a la que se le han añadido sensores para adaptarla a las necesidades exigidas por los circuitos neuronales (véase la Ilustración 42). El robot esta fabricado en aluminio por la empresa Alemana (Multiplex) [80] y distribuido por (HI-TEC) [106] dedicada a la construcción de accesorios para modelismo. Se ha utilizado esta plataforma por ser una primera aproximación sencilla y económica frente a los humanoides de alto presupuesto fabricados por importantes empresas como Sony, Fujitsu [31], etc.

El robot dispone de 16 servo-motores, tres acelerómetros y dos sensores de contacto. Todo el sistema ha sido conectado a la plataforma de prototipado estándar de hardware reconfigurable RC2000 de Celoxica lo que permite cómodamente probar la funcionalidad entre el sistema neuronal y el robot desde un PC a través del bus PCI.
Aunque la plataforma dispone de su propio circuito de control denominado MR-C3024 basado en un microcontrolador AVR Risc con arquitectura Harvard Atmega-128L, se ha anulado dicho sistema pasando todo el control a la plataforma RC2000 a través de un pequeño circuito de adaptación de tensiones. De este modo, todo queda integrado en una misma tarjeta de control, que simplifica el diseño y se eliminan retardos en las señales como consecuencia de la inserción de dispositivos intermedios en el camino de las señales de control.

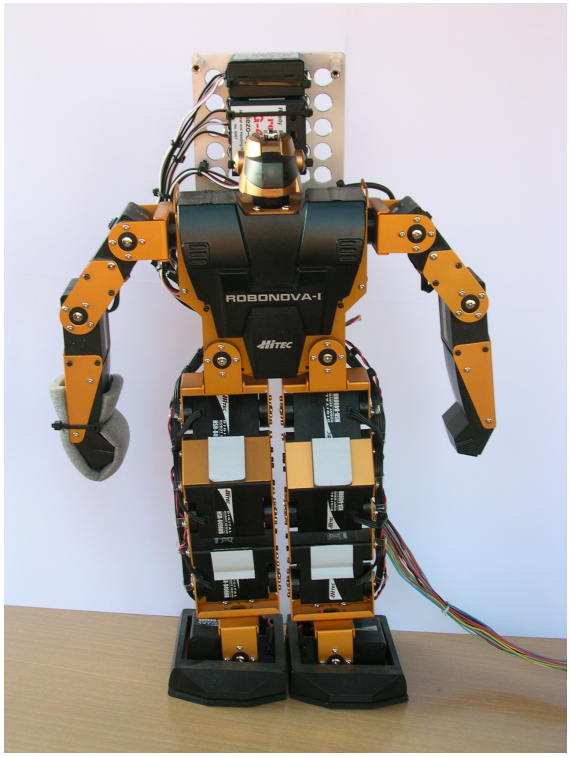


Ilustración 42: Robot humanoide Robonova I

La conexión entre los servomotores y la tarjeta de control se realiza de forma directa utilizando un bus de expansión. Esta conexión es factible debido a que los servos motores incluyen la electrónica de control de potencia compatible con niveles de 3.3V de la FPGA.
Para el caso de los acelerómetros se han utilizado unos circuitos adaptadores de tensión ya que estos circuitos tienen salida en niveles TTL.

5.2.3 Servo sistema actuador

Como se ha mencionado en las secciones anteriores, el robot utiliza servomotores digitales como elementos actuadores. Cada servomotor dispone de un circuito basado en un microcontrolador AVR Atmega-8L del fabricante Atmel [5], con las características descritas en la Tabla 10.

Tensión de alimentación	2.7V a 5.5V
Flash	8K auto reprogramable
RAM	1024 bytes
EEPROM	512 bytes
UART serie	Sí
ADC	Sí, de 10 bit
Reloj	De 0 a 8 MHz

Tabla 10: Características básicas del microcontrolador Atmel8-L

Estos servomotores codifican la posición deseada mediante señales de pulso cuadrado modulado en anchura PWM (véase la Ilustración 43) con valores de tiempo en alta entre 0.4 y 2.6ms, en rango ampliado y entre 1 y 2 ms para rango normal con una frecuencia en entre 50 y 80 Hz. El ángulo de giro máximo es de 190º y la precisión es de aproximadamente de 1º.

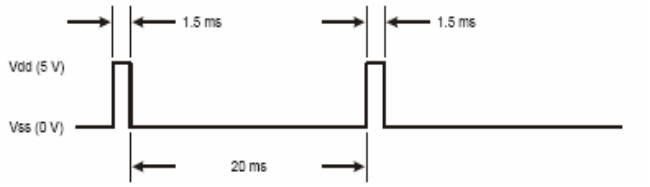


Ilustración 43: Señal PWM de control de posición clásico para servomotores.

La electrónica interna del servomotor trabaja comparando la anchura del pulso de la señal recibida en la entrada con la anchura del pulso de una señal interna dependiente del potenciómetro acoplado al eje del servo. La diferencia entre la anchura del pulso de entrada y la anchura del pulso interno se utiliza como señal de error.
La lógica del servo se encarga de determinar la dirección de giro del motor para minimizar dicho error. Para ello activa los circuitos de potencia del motor modificando la posición del eje y con él la del potenciómetro acoplado de realimentación. La acción de control a aplicar sigue una curva que depende de la distancia entre la posición origen y objetivo (esto se amplía en el apartado 5.2.6 y en la Ilustración 60).
El bucle de realimentación así definido es negativo siendo la precisión del servomotor dependiente tanto de la precisión del potenciómetro como de la precisión de la anchura de los pulsos.
La resolución genérica que se consigue está en torno a un grado pero es posible conseguir resoluciones mayores utilizando servos digitales. Los servos digitales de gama alta (Véase la Ilustración 44) [116] no poseen potenciómetro interno, sino un codificador de posición óptico digital acoplado al eje de giro. Además la inclusión de un circuito microcontrolador permite la implementación de un control local más completo (PID, PD, PI, etc.) que ajusta al máximo la respuesta del motor en función del error entre la posición deseada y la actual. Aunque esta aparente ventaja no resulta una buena opción para diseñar sistemas bioinspirados como veremos más adelante.

Al tratarse de servomotores digitales diseñados específicamente para este tipo de robot disponen de otras funciones, a parte de la gestión de la posición, tales como medida de consumo de potencia y media de la posición actual del eje de giro. Estas funciones son accesibles a través de un protocolo específico denominado (HMI)

Hitec Multi-protocol Interface [96] y se utiliza como sistema de retroalimentación del robot.

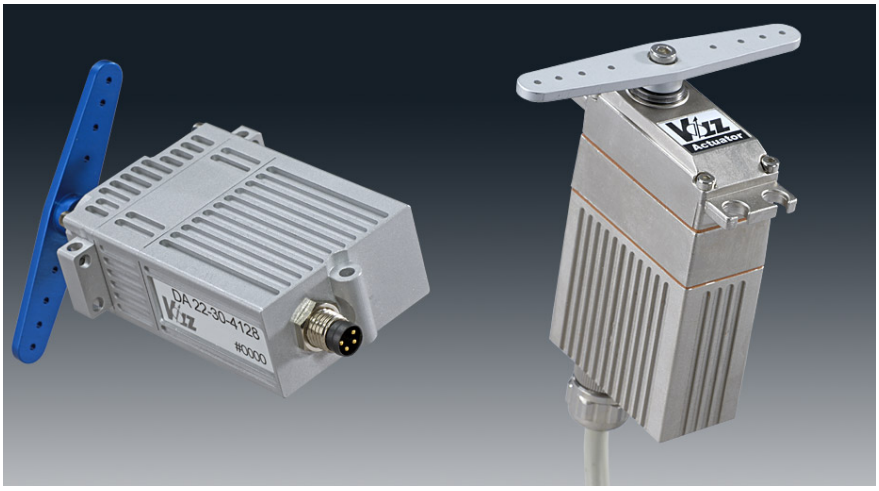


Ilustración 44: Conjunto de servo-actuadores industriales, muy útiles para su aplicación en robótica y aeronáutica. Ilustración adaptada de [116].

Ilustración 45: Señal de control del protocolo HMI para la medida de la posición del eje de giro del servo HSR-8498HB. (A) El controlador genera un pulso de 50 microsegundos. (B) Controlador mantiene la señal de 2 a 20 microsegundos y pasa a alta impedancia. (C) El servo reconoce el pulso y continúa poniendo la señal en baja durante 125 microsegundos. (D) El servo envía la posición manteniendo la señal en alta entre 250 y 2550 microsegundos en función de la posición. (E) El servo pone la señal en baja durante 250 microsegundos. (F) El controlador reconoce el final de la secuencia y coloca la señal en baja.

El protocolo HMI codifica los comandos de control mediante pulsos de duración específica durante la generación de la señal PWM en alta. Por ejemplo, si se genera un pulso de 50 microsegundos (véase

cronograma de la Ilustración 45) de ancho durante el periodo de generación de la señal de posición en alta, el controlador del servo entiende que se está pidiendo la posición actual del eje.
Con otros valores diferentes de ancho de pulsos es posible leer el consumo del motor o programar parámetros del controlador interno. Sin embargo, HMI es un protocolo cerrado y por tanto hay comandos que no son conocidos.

Hasta aquí todo parecen ventajas, sin embargo hay un problema intrínseco al servomotor y es la desconexión temporal del sistema de potencia durante la ejecución de los comando del control. Esto provoca que (en el peor caso; supuesto el eje se encuentra en la posición más alejada del origen) durante 2.5ms el servo no ejerza fuerza sobre la extremidad produciendo dos reacciones:

A) Baja frecuencia de muestreo; Suponiendo que se realiza una secuencia de lectura de posición (2.5ms) seguida de una actualización de posición (2.5ms + 20ms), resulta que la frecuencia de muestreo máxima es aproximadamente de 40 Hz.

B) Disminución del par de fuerza instantáneo; Si el se realizan secuencias de muestreo de posición a 40 Hz, resulta que el motor finalmente se comporta como si estuviese siendo controlado por un controlador PWM de alto nivel aparte del propio interno del servo. La disminución pulsante del par de fuerza trae consigo la aparición de vibraciones indeseadas y un consumo de energía mayor. Una forma de solucionar este problema es disminuir la frecuencia de muestreo, pero entonces el sistema de control (supuesto un controlador clásico como un PID) proporcionará respuestas lentas en la respuesta de control.

Para solucionar estos inconvenientes, nuevamente volvemos la mirada hacia la biología, observando que en los sistemas biológicos el retardo asociado con la acción neuro-motora o una estimulación neuro-sensora es del orden de milisegundos [61] Entonces ¿Cómo se soluciona este retraso en la captación de las señales de posición derivado de una baja frecuencia de muestreo? Los sistemas biológicos han aprendido a solucionar esta aparente desventaja mediante sistemas predictivos con aprendizaje. Los sistemas predicativos con

aprendizaje "predicen" la velocidad y fuerzas en base a la información sensomotora previa a la aplicación de un modelo. Si la predicción resulta ser errónea el sistema aprende a actualizar el modelo y corregir el error. Esta peculiaridad, a parte de solucionar el problema del retardo en las señales, soluciona el modelado de la dinámica interna del sistema ya que la predicción del retardo de la acción motora forma parte intrínsecamente del modelo dinámico.
Por ejemplo: supóngase un brazo robotizado que se mueve siguiendo una secuencia de puntos definidos. Durante las trayectorias, el sistema predictor va proporcionando de forma anticipada las velocidades, aceleraciones y fuerzas que se van a generar en a posición $(n+1)$ desde la posición (n) sin que dependa del conocimiento de la dinámica del brazo, ya que esta información va siendo aprendida poco a poco. Si en la posición $(n+1)$ resulta una predicción errónea, el sistema aplicará una ley de aprendizaje para ajustar los parámetros para la siguiente iteración. Tras un número razonable de iteraciones, el robot dispone de un modelo dinámico genérico del brazo. Si la operación se repite cambiando la dinámica del sistema, por ejemplo ahora el brazo transporta una carga, resulta que partiendo del modelo inicialmente aprendido se puede construir un nuevo modelo que será más sencillo de adaptar que si se partiera de cero ya que al menos incluye los parámetros dinámicos del brazo sin carga. Finalmente, durante la evolución del sistema, se dispone de un grupo de modelos seleccionables en función de diferentes características de entrada dependientes del entorno.
Esta solución es válida para sistemas biológicos ya que el proceso de aprendizaje se realiza de forma continuada durante la etapa de crecimiento pero es difícil de aplicar en robots reales debido al gran número de iteraciones necesarias hasta alcanzar un sistema preciso.

5.2.4 Acelerómetros y el sistema vestibular humano

En esta sección se vuelve la mirada hacia la biología para establecer similitud entre el sistema vestibular humano y el sistema de equilibrio utilizado en el robot Humanoide de experimentación. El objetivo consiste en observar qué soluciones aporta la biología en este campo y poder construir una aproximación electromecánica lo más fiel y funcional posible.

Mantener el equilibrio en robots Humanoides ya sea en bipedestación estática o dinámica, es una tarea compleja debido a varios factores que pueden ser resumidos en:

- Desplazamiento del centro de gravedad prediciendo de forma correcta los vectores de fuerza gravitatoria e inercia.

- Interacción con un entorno cambiante de un sistema físicamente inestable uno o dos puntos de apoyo en una superficie que puede ser irregular.

- Necesidad de introducir acciones de desestabilización para producir desplazamiento: El sistema de control ha de introducir una desestabilización controlada (desplazamiento del centro de gravedad) para producir un desplazamiento de toda la estructura.

- Necesidad de ejecutar acciones de control complejas que involucren varios motores, como resultado de medidas de aceleraciones lineales y angulares en un espacio tridimensional.

A modo de resumen tres son los retos que ha de superar el sistema de control:

1. Mantener una posición constante (mantener el equilibrio)
2. Generar respuestas que se anticipen a los movimientos (predicción)
3. Sistema adaptativo (aprendizaje)

La biología resuelve este problema mediante un sistema sensorial y un sistema de control coordinado que involucra diferentes centros nerviosos (tronco encefálico, cerebelo, corteza motora etc.) así como sensores de velocidad y aceleración tanto lineal como angular.
En el cuerpo humano, el sistema sensorial de equilibrio o sistema vestibular se localiza en el oído interno. La aceleración lineal y angular se obtiene mediante un grupo de cinco órganos sensoriales gemelos (cinco en cada oído) que conforman el laberinto membranoso o vestibular. Estos cinco órganos proporcionan información acerca de hacia dónde se mueve el cuerpo e influye en la información de percepción del espacio que nos rodea.

La aceleración lineal gravitatoria y la resultante producida por el cuerpo son detectadas por el *utrículo y el sáculo* (Ilustración 46 e Ilustración 48) mientras que las aceleraciones angulares producidas por la rotación de la cabeza o el cuerpo son detectadas por los *conductos semicirculares* (vertical anterior, vertical posterior y horizontal) Ilustración 47.

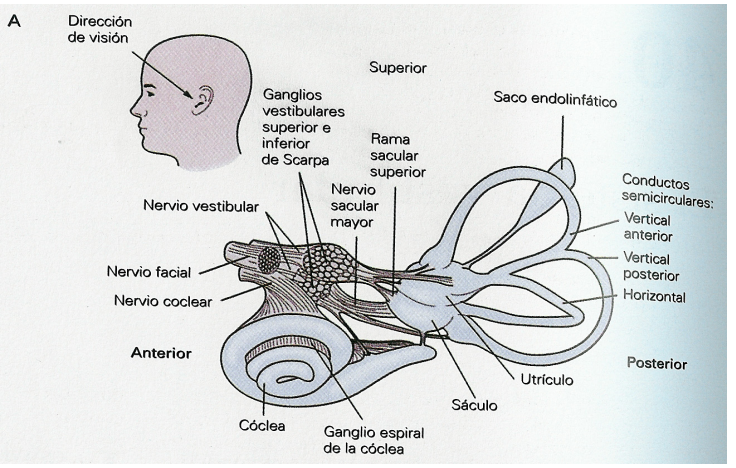


Ilustración 46 Estructura del sistema vestibular localizado en el oído interno donde se observan el *utrículo* y *el sáculo*. Ilustración adaptada de [58].

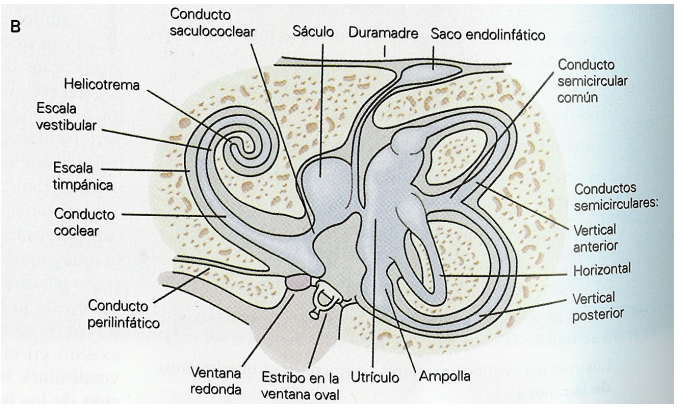


Ilustración 47: Detalle del sistema vestibular y oido: Obsérvese los *conductos semicirculares* (vertical anterior, vertical posterior y horizontal). Ilustración adaptada de [58].

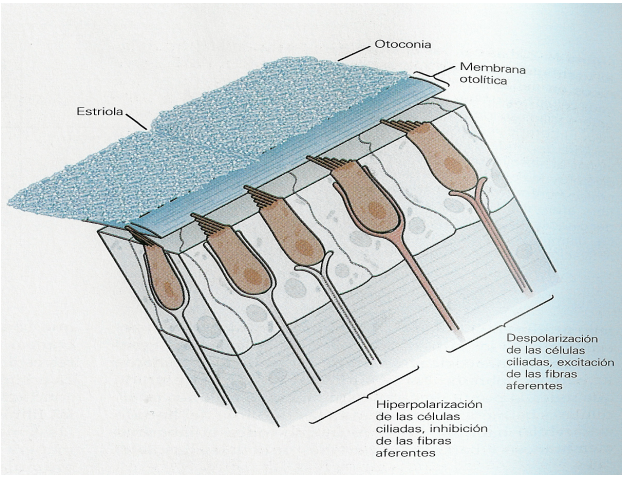


Ilustración 48: Vista interior del *utrículo*. Las células ciliadas detectan la inclinación de la cabeza gracias al desplazamiento de la membrana Otolítica. Ilustración adaptada de [58].

Reproducir mecánicamente la arquitectura interna del sistema vestibular humano no es una tarea sencilla. Además no resultaría inteligente, dado que la base tecnológica de que disponemos es totalmente diferente. Sin embargo sí que podemos construir sistemas electromecánicos que se comporten de forma similar a como lo hace su equivalente biológico.

De este modo, la funcionalidad del *utrículo* y el *sáculo* puede ser reproducida utilizando acelerómetros lineales piezo-eléctricos, mientras que a modo de *canales semicirculares*, es posible utilizar Giróscopos.
El Giróscopos está basado en un fenómeno físico conocido como *"principio de conservación del momento angular"*. Dicho de forma intuitiva se puede interpretar diciendo que todo cuerpo que describe un movimiento circular tiene una cantidad determinada de energía de rotación que conserva constante aunque varíen los parámetros que afectan al mismo.
Por ejemplo, (véase la Ilustración 50) si una masa se encuentra girando en el sentido de las agujas del reloj y se ejerce una fuerza (F) perpendicular al eje de giro se observará que el sistema se inclina con dirección perpendicular al eje de giro y a la fuerza aplicada (flechas rojas de la Ilustración 50).

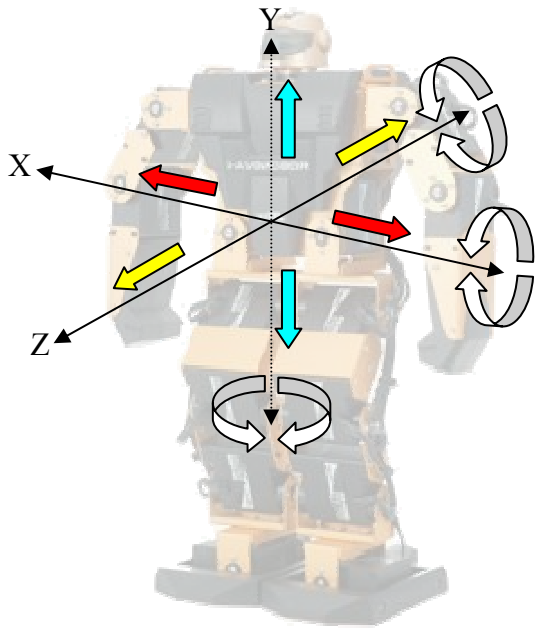


Ilustración 49: Movimiento tridimensional; se observan 6 posibles aceleraciones lineales y 6 posibles aceleraciones angulares.

Este tipo de giróscopos, con un volante giratorio, se utilizan en aviones para fijar el rumbo y en satélites como mecanismo de orientación espacial. No obstante, existen versiones miniaturizadas, así como giroscopios construidos con cerámicas piezo-eléctricas (véase Ilustración 51).
Su detección se basa en la distorsión que sufre una corriente eléctrica pulsante que atraviesa una pieza cerámica como consecuencia de las fuerzas que aparecen cuando se producen cambios en la velocidad angular del sistema donde se encuentra.

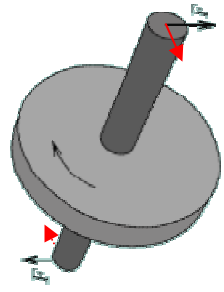


Ilustración 50: Efecto giroscópico; si se aplica un par de fuerzas perpendiculares al eje de giro (flechas negras en la figura), el sistema responde generando un

nuevo par de fuerzas de nuevo perpendiculares al eje y opuesto a las fuerzas aplicadas (flechas rojas en la figura) [29].

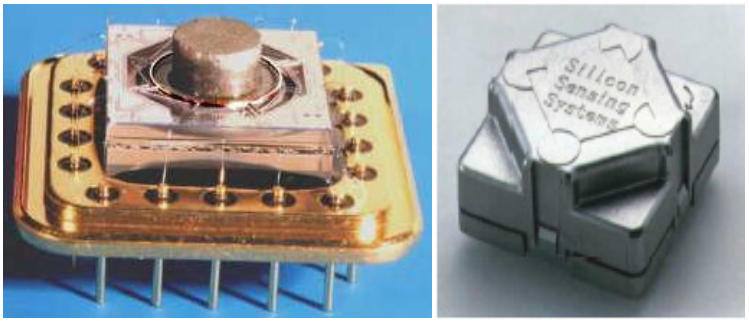


Ilustración 51: Sensor piezo-resistivo modelo CRS03 cortesía de *Silicon Sensing Systems Japan*

Estos sensores son pequeños, del tamaño de un chip de montaje superficial. A modo de ejemplo se pueden encontrar los sensores Gyrostar (ENC-03J) fabricado por *Murata* o el ADXRS150 de *Analog Devices* [4].

En el robot se han instalado tres giróscopos modelo G-400 fabricado por la empresa Alemana Robbe con ajuste digital de ganancia. Estos giróscopos son del tipo piezo-eléctrico y aunque en esencia miden aceleraciones angulares, tras un proceso de integración digital interno, se comportan como giróscopos dando una medida de velocidad angular. El proceso de integración tiene cierto error acumulado debido a la precisión de los acelerómetros piezoeléctricos internos por lo que el microcontrolador interno realiza un ajuste de compensación teniendo en cuenta dos parámetros:

1. La temperatura ambiente que modifica la respuesta de las cerámicas piozo-eléctrica
2. La historia pasada de conjunto de muestras desde el instante (t) hasta el actual.

Estos giróscopos están pensados para controlar directamente servomotores, proporcionando una acción de control directa. Sin embargo, utilizando esa señal de salida como entrada sensorial del sistema es posible utilizar el acelerómetro como elemento sensor en vez de cómo elemento senso-actuador.

Ilustración 52: Giroscopio Piezo eléctrico modelo G-400, cortesía de *Robbe Modellsport*

Todas las señales de entrada y salida del giroscopio (véase Ilustración 52) van codificadas en PWM "*Pulse Width Modulation"* definiéndose una codificación exactamente igual que un servomotor como los utilizados para las articulaciones del robot.

Estas señales son tres (véase Ilustración 52): Señal de Ganancia (Aux), Cero (Rx) y Salida (Servo).

- Señal de Ganancia (Aux): Modificando la anchura del pulso entre 1 y 2ms se ajusta la ganancia o respuesta (en ambos sentidos de giro) del giróscopo frente a un cambio en la aceleración angular.

- Cero (RX): Es una señal de entrada al giróscopo. Con esta señal se ajusta la posición del servomotor que se conectaría al giróscopo. Puesto que no se va a conectar ningún servomotor, esta señal trabajará como señal de ajuste de cero. Es decir, configura un tiempo entre 1 y 2ms a partir del cual se considera que la aceleración medida se produce en un sentido u en otro de giro. En el sistema se ha dejado fija a 1.5ms para mantener el mismo rango de medida hacia un sentido y otro.

- Respuesta (Servo): Es la respuesta del giroscopio. La señal está codifica en anchura, al igual que la señal que se genera para controlar un servo motor, y da una medida de la dirección de giro y velocidad angular instantánea del sistema tomando como referencia el instante configurado con la señal Cero (Rx).

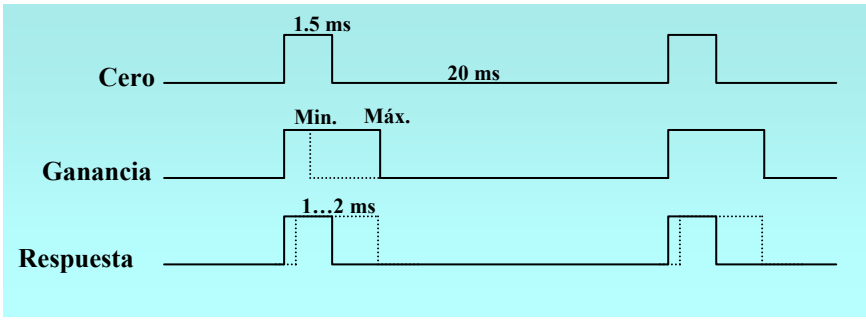


Ilustración 53: Cronograma de señales de salida (Cero, Ganancia y Respuesta) entre cada giróscopo y la plataforma FPGA RC2000. Obsérvese que la señal "Cero" se mantiene fija a 1.5ms para mantener el mismo rango de medida de aceleración en un sentido y otro de giro.

La frecuencia de la señal de respuesta de cada giróscopo está limitada y varía entre 45 y 70 Hz debido a que el sistema está pensado para controlar servomotores de forma directa. Estos servomotores, como se ha mencionado, trabajan con señales PWM con las frecuencias descritas anteriormente. No obstante esta frecuencia de muestreo es suficiente como para detectar las aceleraciones y velocidades producidas por los movimientos del robot.

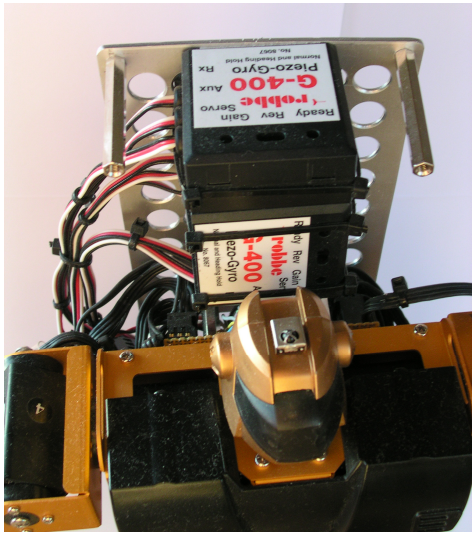


Ilustración 54: Colocación de los tres giróscopos en el robot. Obsérvese la disposición de cada giróscopo para medir la velocidad angular en cada eje posible de giro.

Los giroscopios han sido colocados en la parte superior del robot (véase Ilustración 54), a nivel de la cabeza de igual modo que el biología. De este modo el sistema sensorial es sensible a cualquier desplazamiento del centro de gravedad.

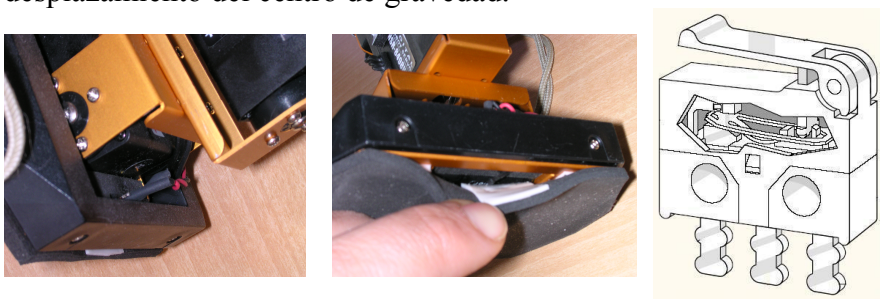


Ilustración 55: Detalle del micro conmutador colocado en cada pié del robot humanoide.

El sistema sensorial se complementa con la incorporación de dos sencillos interruptores de contacto dispuestos bajo cada pié del robot. Estos interruptores no proporcionan una medida analógica de la presión ejercida, pero conjuntamente con la información de la posición de cada articulación, proporciona un método para identificar en qué estado se encuentra el robot correspondiente a un patrón de movimiento predefinido.

5.2.5 Interfaz Simulador-Robot-Computador

Previo a al integración del sistema humanoide con el simulador neuronal, se ha desarrollado una interfaz de alto nivel (véase la Ilustración 58) con el robot. Mediante esta interfaz se puede manejar manualmente el robot permitiendo comprobar la correcta funcionalidad de todo el sistema. Además es posible generar y probar patrones de movimiento dando la posibilidad de ver el comportamiento del sistema senso-motor (véase la Ilustración 57).

La conexión de bloques de alto nivel se observa en la Ilustración 56. La conexión entre el software del PC y la tarjeta se realiza físicamente a través del bus PCI y la memoria de acceso compartido situados en la tarjeta, mientras que la transferencia de datos se realiza a través de canales DMA.

Existe un repertorio de comandos que interpreta el hardware programado en la FPGA diseñado como una máquina de estados. El conjunto de comandos permite leer o escribir posiciones de forma independiente para cada motor, configurar la velocidad o realizar peticiones de parámetros capturados por el hardware.

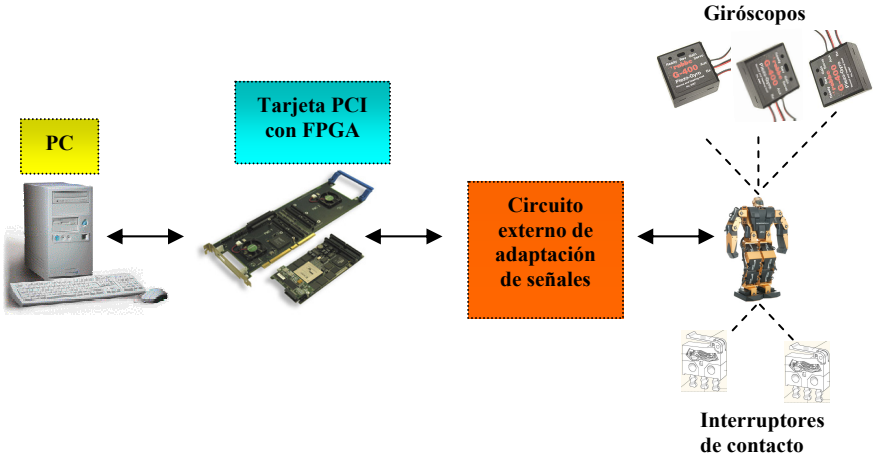


Ilustración 56 Interconexión entre el robot la plataforma FPGA y el computador

El mapa de memoria es muy sencillo. Cada controlador de motor tiene asociado una dirección de memoria donde se encuentran los parámetros de configuración (posición deseada, velocidad deseada, posición actual etc). Cada vez que la aplicación realiza un cambio sobre la memoria de configuración, la unidad hardware correspondiente integrada en el chip se configura con el nuevo valor produciendo el cambio deseado sobre el robot. De igual modo cuando el software realiza una petición de datos, como por ejemplo leer la posición de un motor, el circuito correspondiente escribe en la memoria de configuración el resultado capturado.
A modo de resumen, la Tabla 11 muestra el conjunto de funciones software que componen la librería de control de alto nivel del robot. No obstante hay que aclarar que esta librería es sólo válida para sistemas de control que funcionen en el PC. La integración del sistema de control del robot con el simulador neuronal se realiza completamente en hardware mediante el acceso directo a los registros de configuración de las diferentes unidades de control. En este caso, el PC pasa a un según plano realizando únicamente tareas de monitorización de alto nivel sobre el funcionamiento del sistema neuronal.

Funcionalidad	Denominación de la función
Entre otras tareas, posiciona los motores en una aposición segura	void biped_init(void)
Libera recursos y desactiva el robot en una posición segura	void biped_end(void)
Obtiene la posición enviada a un motor	unsigned int get_send_position(unsigned int motor)
Ajusta la posición deseada de un motor	void set_motor_position(unsigned int motor, DWORD position)
Obtiene la posición real, actual de un motor	DWORD get_motor_position(unsigned int motor)
Ajusta la velocidad de un motor individual	void set_speed(unsigned int motor, DWORD speed)
Ajusta la velocidad de todos los motores al mismo tiempo	void set_all_motor_speed(DWORD speed)
Ajusta o lee la posición de todos los motores al mismo tiempo	void set_all_motor_positions(DWORD *positions) DWORD * get_all_motor_positions(void)
Lee todos los valores sensados del robot al mismo tiempo	void get_all_sense_data(void)
Recupera un valor sensado indicado específicamente	DWORD get_index_sense_data(unsigned int index)
Ajusta la ganancia del giroscopio seccionado	void set_gain_servo(unsigned int index, unsigned int gain)
Ajusta la señal de cero del giroscopio seleccionado	void set_gyro_zero(unsigned int index, unsigned int position)
Ejecuta patrones de movimiento predefinidos para realizar medidas de aceleraciones durante su ejecución.	void stand_up(void) void sit_down(void) void forward_walk(void) void back_walk(void) void hands_up(void) void hands_down(void) void hands_before(void) void hands_wave(void) void tell_hallo(void)
Desconecta los generadores de PWM	void biped_off(void)
Conecta los generadores de señales PWM	void biped_on(void)

Tabla 11: Conjunto de funciones de control para el robot implementadas a modo de librería.

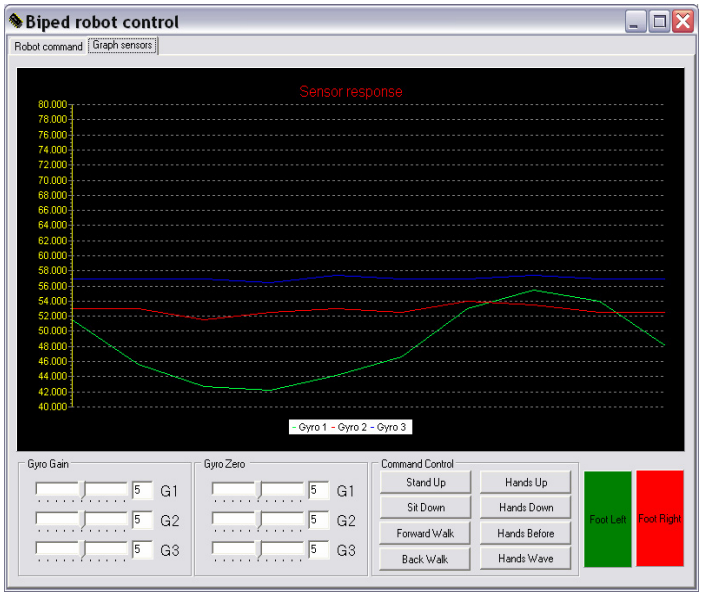


Ilustración 57: Representación gráfica instantánea obtenida en tiempo real por la aplicación, de los tres acelerómetros instalados en el robot Humanoide Robonova. Obsérvese el estado de los sensores de contacto en la parte inferior derecha.

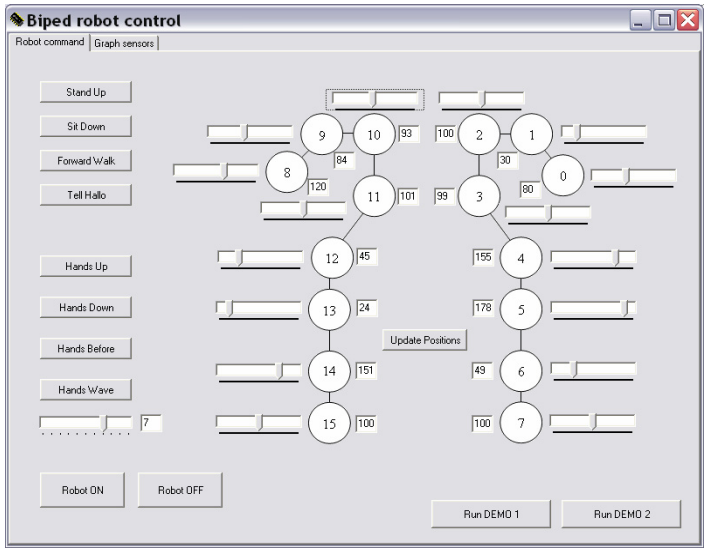


Ilustración 58: Detalle de la ventana de control del robot. En ella se ofrece la posibilidad de mover de forma independiente cada articulación (cada circunferencia representa un motor), conocer su posición y ajustar la velocidad de giro de cada motor, entre otras posibilidades.

5.2.6 Servomotores y la unidad motora de acción muscular humana

En las siguientes líneas se establece una asociación, lo más fiel posible, entre el funcionamiento de los músculos y el sistema de funcionamiento de los servomotores utilizados en el robot Humanoide objeto del presente estudio. Como veremos, se enmascara el funcionamiento codificado intrínsico de cada servo para establecer un puente de conexión con el sistema neuronal de eventos descrito en el Capítulo 4. De este modo cada servo emula, dentro del límite que la tecnología a este nivel puede ofrecer, a un sistema muscular que funciona con pulsos.

Una de las principales funciones del sistema nervioso es la generación de las señales de control para los músculos del cuerpo. El sistema nervioso dice cuándo y cómo moverse para alcanzar un objetivo. Los músculos, tendones y articulaciones tienen un conjunto de propiedades mecánicas que han de ser identificadas e incluidas en el problema de control que resuelve el cerebro.
Los robots actuales, robots industriales, siguen teniendo problemas de control, sobre todo, cuando el sistema está formado por multitud de segmentos dependientes entre si, así como multitud de grados de libertad. En este tipo de arquitecturas una perturbación en una de las articulaciones afecta a todas las demás y, de igual modo, una acción de control sobre una articulación tiene influencia sobre el conjunto. En la biología este problema se resuelve en gran medida debido a las características intrínsecas de los músculos. Nuestros músculos están dotados de propiedades mecánicas que contribuyen de forma muy importante a la simplificación de la dinámica del sistema de control.

Un músculo típico está formado por miles de fibras musculares trabajando en paralelo. Un conjunto de fibras musculares agrupadas bajo el control de una *neurona motora* constituyen una *unidad motora* (las neuronas motoras se localizan en la médula espinal y abarcan entre 1 y 4 segmentos vertebrales por músculo). De este modo, el sistema nervioso puede controlar diferentes partes de un mismo músculo. Puede hacerlas trabajar al unísono, para obtener una respuesta rápida y enérgica o de forma consecutiva, para obtener una respuesta proporcional y elástica o incluso, y dado que las fibras

musculares necesitan cierto tiempo para volver al estado de relajación tras una contracción, alternar el funcionamiento de diferentes unidades motoras para obtener una respuesta muscular sincronizada de mayor rendimiento. Todo esto unido a la posibilidad de almacenamiento de energía mecánica en los tendones y las *aponeurosis* (elementos de unión entre las fibras musculares y los tendones), dotan al músculo de unas propiedades elásticas y mecánicas difícilmente superables por los sistemas electromecánicos actuales.

La tensión activa de un músculo y por tanto la fuerza que ejerce, depende de la frecuencia de los pulsos de estimulación, del nivel de actividad que recibe cada fibra muscular, de su longitud y de la velocidad con que se está moviendo [60]. Por tanto, dado que la forma de onda de cada pulso es aproximadamente constante, se puede concluir que la fuerza ejercida por un músculo en contracción se codifica con el tiempo en baja de la señal (véase la Ilustración 59).
Con esta idea en mente es posible hacer trabajar a un servomotor de forma parcialmente parecida a como lo hace un músculo o al menos conseguir que la información de control sea coherente con la respuesta (basada en pulsos) que proporciona el circuito neuronal del simulador descrito en el Capítulo 4.
En este nivel, el controlador implementado en la FPGA para cada servo, recibe como entrada la velocidad y la posición donde se desea que gire el servo. Sin embargo la salida del sistema neuronal es una salida de pulsos. Por otro lado, la biología cuenta con sistemas musculares gemelos trabajando en modo agonista y antagonista, sin embargo, el robot utiliza un único motor por eje de giro. Por todo ello, ha sido necesario el desarrollo de un circuito intermedio que realiza dos tareas esenciales:

1. Decodificar la señal codificada en frecuencia de pulsos a velocidad, posición y pares de fuerza.
2. Detección del sentido de giro (paso de diferencial de pulsos a sentido de giro).

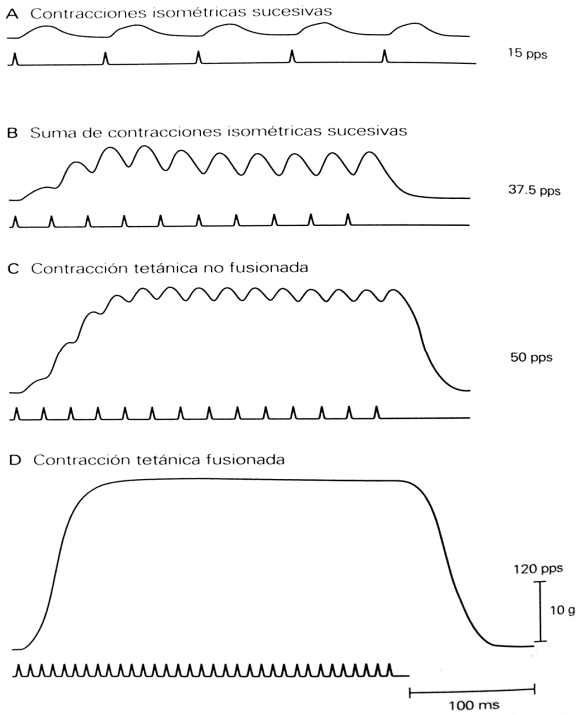


Ilustración 59: Contracción muscular en función de la frecuencia de pulsos (pps: pulsos por segundo) de estimulación en 4 casos diferentes (A, B, C y D) diferenciados por la frecuencia de estimulación. Obsérvese como el músculo es capaz de seguir la señal de pulsos de entrada a frecuencias bajas (A y B) comportándose como un elemento integrador para frecuencias más elevadas (C y D). El segmento vertical en la parte derecha ilustra la magnitud de una fuerza equivalente a 10g de peso. El músculo en el que se ha centrado el estudio es el *caudofemoral* de un gato (Ilustración adaptada de [61]).

Antes de dar una solución al circuito que realice las tareas 1 y 2 mencionadas anteriormente, es importante analizar las características de la señal que va a recibir el sistema motor.

Por simplicidad se considera que la señal de entrada al controlador procedente del sistema neuronal es un tren de pulsos formados por unos y ceros, donde cada pulso tiene una duración fija mínima y una amplitud constante. Dado que la señal codifica la fuerza ejercida por el músculo en frecuencia y la anchura del pulso en estado alto es fija, la codificación se realiza modificando el tiempo en baja de la señal. Una forma sencilla de hacer esto consiste en medir el tiempo en baja y asignarle una curva de respuesta para la velocidad de giro del servo. El problema de esta técnica es que el motor generará la respuesta al menos con un tiempo de retraso proporcional a

la diferencia temporal entre dos eventos y no justamente cuando llega el primer pulso excitativo como haría un músculo biológico. Sin embargo, en la práctica, no resulta un problema ya que el pequeño retraso en la acción de control será absorbido por el sistema anticipador de control.
Otra posibilidad sería tener en cuenta la historia pasada de la señal de entrada y asignar un valor de tiempo en baja promediado en cuanto llegue el primer pulso de control. Sin embargo esta técnica filtraría en cierto modo la señal de pulsos de entrada con los siguientes pulsos y complica el hardware dedicado.

Una observación importante, es que la señal de entrada de pulsos no trae implícitamente una codificación de la posición ya que la posición se retroalimenta hacia la red neuronal con otra señal. Sin embargo sí es preciso generar de forma artificial una posición objetivo para el servomotor.

El par aplicado al motor depende de la diferencia entre la posición actual del eje del servo y la posición objetivo, de tal modo que si la posición es muy alejada el par aplicado será mayor que si la posición destino es cercana siguiendo la curva descrita en la Ilustración 60 . Si esto no fuese así la propia inercia del eje de giro unida a la respuesta sobredimensionada del motor para disminuir el error de posición, cuando la posición del eje está próxima a la posición objetivo, produciría un efecto de sobre-oscilación.

Siguiendo la Ilustración 60 se observa que la velocidad de rotación libre y por tanto el par aplicado (la velocidad de giro es proporcional al par aplicado) en función de error entre la posición actual y la posición objetivo, sigue una curva característica lineal en un intervalo inicial y logarítmico en el intervalo final. Esta curva define la respuesta del servomotor y por tanto una parte de la dinámica de control sobre el sistema. Dado que no se conoce a priori qué masa y a qué inercias estará sometido el eje del servo, éstas se calculan suponiendo que el motor mueve una masa límite proporcional al par máximo aplicable por el motor suponiendo que con una masa inferior el exceso de par aplicado producirá un comportamiento deseable sin sobreoscilaciones.

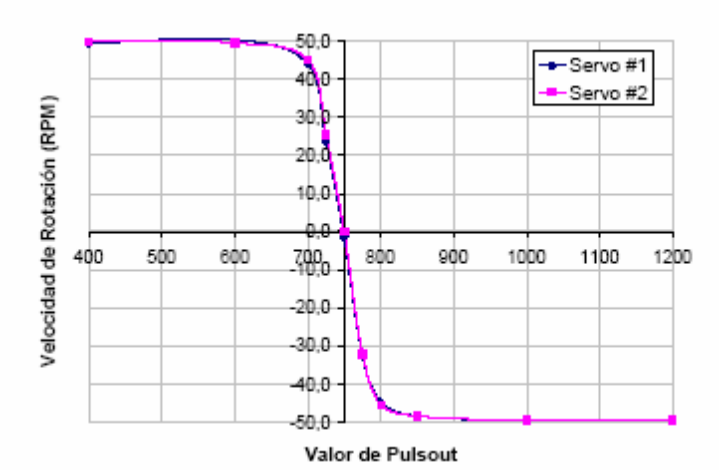


Ilustración 60: Respuesta en RPM (revoluciones por minuto) de dos servomotores (servo #1 y servo #2) en función de la anchura de pulso aplicado. Obsérvese que a pesar de ser iguales existen diferencias apreciables en la respuesta (línea morada y línea azul) probablemente producidas por rozamientos o ajuste del sistema reductor interno. El tiempo (eje horizontal) se representa en microsegundos. (Ilustración adaptada del manual de usuario del robot Stamp de Parallax.inc [88]).

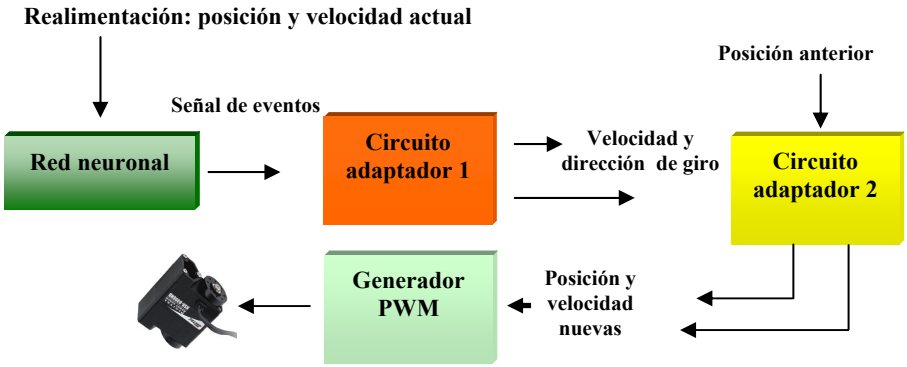


Ilustración 61: Diagrama de interconexión genérico de la red neuronal servomotor con circuitos adaptadores 1 y 2 (véase texto para mayor detalle).

Como se ha mencionado al inicio de esta sección, la biología cuenta con sistemas musculares gemelos trabajando en modo agonista y antagonista (véase la Ilustración 62). Probablemente, y a modo de hipótesis, la evolución biológica decidió no desarrollar músculos no contráctiles o extensores por no ser una ventaja evolutiva (entiéndase por músculos que pudieran ejercen fuerza al estirarse) sin embargo, al disponer de al menos dos músculos (agonista y antagonista)

es posible controlar la rigidez de la articulación. Característica ventajosa, por ejemplo, para el mantenimiento de la postura bípeda (los músculos de las piernas han de generar cierta rigidez para mantener al individuo erguido).

El robot dispone de un único motor por eje de giro y por tanto, la rigidez de la articulación no puede ser ajustada del mismo modo. Entiéndase por rigidez la fuerza o momento que se opone al movimiento de la articulación producida por la reducción de engranajes (rigidez pasiva o *momento de retención*) conectada al motor, o por la respuesta del controlador interno (rigidez activa durante el giro o fuerza que se opone a un cambio en la velocidad de giro). La rigidez articular influye en el consumo energético del sistema, en el mantenimiento de la postura y en la dinámica del robot.

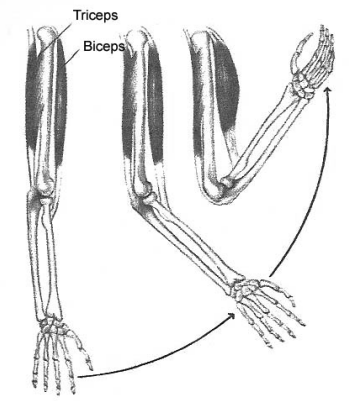


Ilustración 62: Músculo agonista (contraído) y antagonista (relajado) de un brazo humano.

Los servomotores utilizados, desgraciadamente tienen una alta rigidez pasiva debido a que están diseñados con cajas de engranajes con una relación de reducción elevada entre 200 y 300 a 1 (véase la Ilustración 64) lo que produce una alta rigidez articular. Esta rigidez articular sobrecarga mecánicamente las articulaciones frente a perturbaciones externas y consume energía. Por tanto imposibilita el aprovechamiento de la inercia, dentro del ámbito de control bioinspirado de la dinámica, como mecanismo de ahorro energético.

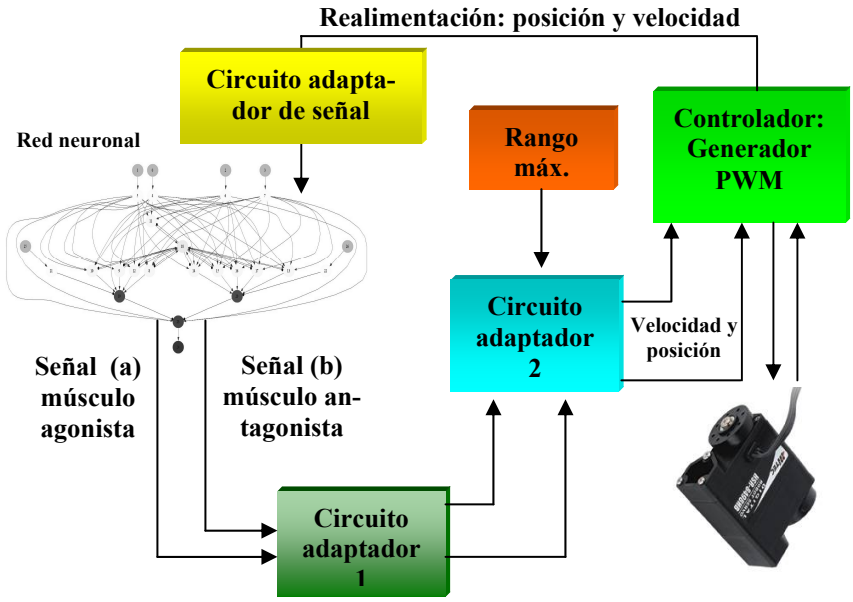
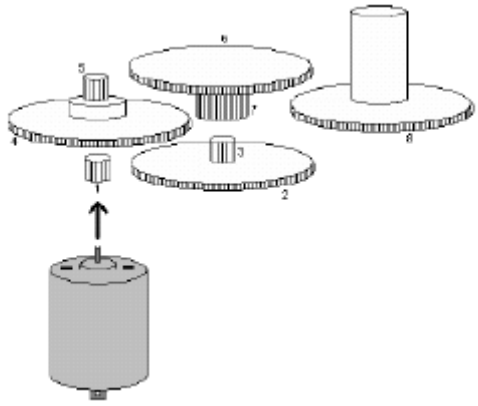


Ilustración 63: Esquema modular de control en ciclo cerrado entre la red neuronal y un servomotor con la inclusión de señales agonista y antagonista.

Engranaje	n1	n2	n3	n4	n5	n6	n7	n8
Nº de dientes	10	62	10	50	10	35	16	41



$$R = \frac{n2}{n1}\cdot\frac{n4}{n3}\cdot\frac{n6}{n5}\cdot\frac{n8}{n7} = \frac{62}{10}\cdot\frac{50}{10}\cdot\frac{35}{10}\cdot\frac{41}{16} \approx 278:1$$

Ilustración 64: Sistema reductor de un servo estándar modelo S3003 de Futaba junto con el número de dientes de cada engranaje y el cálculo de la relación de reducción [27] [26].

Una solución al circuito de adaptación puede verse en la Ilustración 61 y en la Ilustración 63, donde la señal de control procedente de la Red Neuronal es decodificada para proporcionar un valor de velocidad y sentido de giro.
La aproximación bio-inspirada para emular el funcionamiento agonista y antagonista pasa por utilizar un circuito acumulador con dos entradas de pulsos (a) y (b) (véase la Ilustración 63). El funcionamiento es muy sencillo: Si la frecuencia de pulsos de la entrada (a) es mayor que la frecuencia de pulsos de la entrada (b) entonces el sentido de rotación del eje de giro es horario y anti horario si se da el caso contrario. De esta tarea junto con el escalado de tiempo en baja para proporcionar la velocidad de giro, se encarga el circuito *adaptador 1*.
La frecuencia de la señal de entrada que recibe el circuito *adaptador 1* procedente de la red neuronal está entre 5 y 60Hz correspondiente con la respuesta obtenida en pruebas con músculos reales. En estas pruebas se observa que la frecuencia de estimulación de un músculo para ejercer una fuerza entre 0 y el 100 % de la capacidad total, depende del tamaño del músculo, pero en general, el rango para un músculo humano está comprendido entre 5 y 60Hz.
La señal de pulsos agonista y antagonista tendrá una representación discretizada del potencial de acción de una neurona motora (véase la Ilustración 65). Este potencial de acción está representado por una señal cuadrada de tiempo en baja variable y tiempo en alta fijado a 1.5ms correspondiente aproximadamente con el periodo del potencial de acción real [10].

El circuito *adaptador 2* realiza la tarea de transformar la velocidad y sentido de giro a posiciones y velocidades para el generador PWM. El circuito trabaja generando un rango de posiciones para el servo con la dirección y velocidad indicada por el circuito *adaptador 1*. (El circuito generador de PWM junto con el controlador de velocidad se analiza en profundidad en el apartado 5.3).

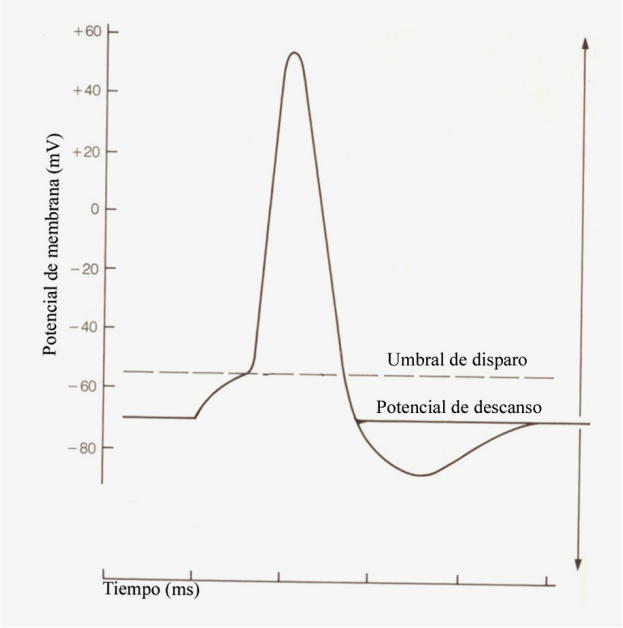


Ilustración 65: Representación gráfica del potencial de activación de una neurona real. Obsérvese cómo el pulso, por encima del umbral de reposos, tiene una duración aproximada entre 1.5 y 2ms. Imagen adaptada de [56].

5.3 Diseño del hardware de control

El sistema de control ha de manejar un total de 27 señales (16 servos + 3 giroscopios x 3 señales cada uno + 2 sensores de contacto). En el hardware de control se ha adoptado un esquema de control distribuido implementando un circuito controlador para cada motor. De este modo se aprovecha el paralelismo inherente que proporciona la tecnología basada en FPGA obteniendo un sistema mucho más rápido y, teóricamente, más tolerante a fallos.

El circuito controlador para los motores está compuesto por un generador de señal PWM, un controlador de velocidad y un generador de protocolo para la lectura de la posición y consumo (véase la Ilustración 66). El circuito generador de PWM se implementa con cuatro registros, un circuito sumador y un comparador. Dos de los cuatro registros almacenan el tiempo actual en baja y el tiempo en alta de la señal contado en ciclos de reloj mientras que los otros dos registros almacenan los límites en baja y alta respectivamente. De

este modo, modificando de forma sincronizada y concurrente el contenido de estos últimos registros es posible modificar la forma de la señal. La sincronización entre el circuito sumador, que incrementa los registros de cuenta, el circuito comparador que pone a cero los contadores cuando la cuenta ha llegado al límite, y el circuito externo que fija la posición deseada, es muy importante ya que las actualizaciones han de hacerse al inicio (flanco de subida) de la generación del tiempo en alta de la señal PWM. Si la modificación ocurriese a la mitad el motor correspondiente perdería la posición durante al menos 20ms con lo que aparecerían vibraciones indeseadas en la estructura del robot.

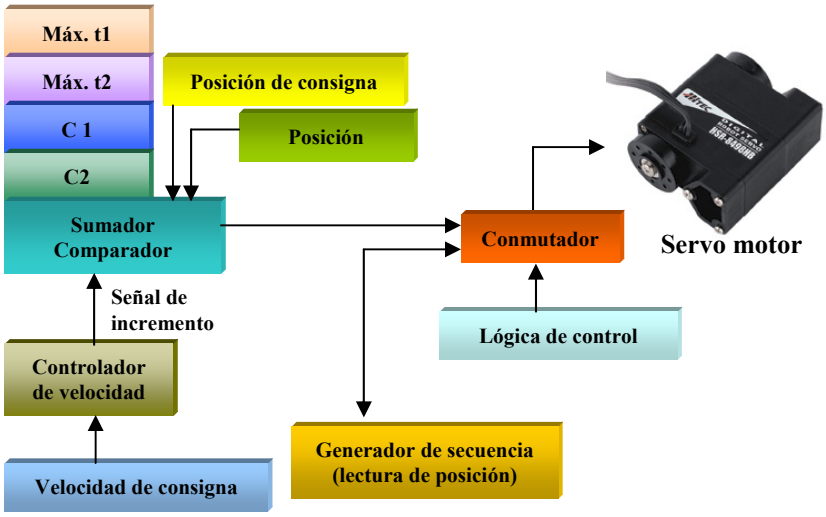


Ilustración 66: Esquema de controlador individual de cada servomotor. El sistema precisa de un dispositivo de sincronización que mantenga la concurrencia en el acceso a los registros de control.

El control de velocidad se implementa incrementando o decrementando el registro asociado con el periodo en estado alto de la señal PWM, con una diferencia temporal entre cada incremento o decremento que depende de la velocidad deseada (observación: decrementar o incrementar el registro implica girar en un sentido o en otro). Por ejemplo; suponiendo que el periodo actual de la señal en estado alto es de 1.5ms, la posición de consigna está a 1.8ms y la velocidad se fija a un valor de 5 (el intervalo de velocidad entre la velocidad máxima y la mínima posible se ha dividido en 10 subintervalos por simplicidad), el controlador de velocidad calcula un

número de ciclos de reloj de espera que generarán un retardo de tiempo entre incremento e incremento desde 1.5 hasta 1.8ms tal que la velocidad media durante el recorrido sea la deseada (para este ejemplo justo la mitad de la velocidad máxima).

Dado que el par generado por el motor está limitado por las características intrínsecas del mismo, la velocidad final de giro dependerá de la carga aplicada al motor y por ello se ha establecido un intervalo de velocidades posibles entre 0 y 10 girando el motor en vació sin carga. Esta resolución puede ser aumentada teniendo como límite la resolución en la generación de la señal que depende directamente de la frecuencia de reloj del sistema. La Ecuación 36 y la Ecuación 37 muestran la transformación directa e inversa en la velocidad de giro y el número de ciclos de cuenta.

$$N_{ciclos} = Min_{vel} - \left(Vel \cdot \frac{\left(Min_{vel} - Max_{vel} \right)}{10} \right)$$

Ecuación 36: Cálculo del número de ciclos de espera entre incremento o decremento, en función de la velocidad deseada comprendida entre 0 y 10. Obsérvese que el número de ciclos es siempre mayor que cero ya que la velocidad mínima (Min_{vel}) se corresponde con el retardo máximo.

$$Vel = \frac{10 \cdot \left(Min_{vel} - N_{ciclos} \right)}{Min_{vel} - Max_{vel}}$$

Ecuación 37: Cálculo de la velocidad, en rango entre 0 y 10, a partir del número de ciclos de espera entre incremento o decremento para el ajuste de velocidad.

Las constantes Max_{vel} y Min_{vel} en la Ecuación 36 y en la Ecuación 37dependen de la frecuencia de trabajo del circuito, siendo estos valores calculados siguiendo la Ecuación 38. Obsérvese que el cambio en la frecuencia de trabajo del sistema no influye en el cálculo para el ajuste de velocidad (en la Ecuación 38 el retardo asociado para Max_{vel} es de 0.01ms y para Min_{vel} es de 0.1 ms) ya que al modificar la frecuencia también se modifican el número de ciclos de espera y por tanto el tiempo permanece constante. Lo que cambia al modificar la frecuencia de trabajo es el número de veces que hay que incrementar o decrementos el registro de posición desde la posición origen hasta la posición destino. Por tanto, con frecuencias bajas se tendrá un número menor de pasos entre la posición inicial y

la posición destino y con frecuencias más altas este número será mayor.
Esta característica, se traduce en la existencia de una frecuencia mínima de trabajo por debajo de la cual el controlador de velocidad calcularía un retardo de tiempo mayor que el intervalo de tiempo de la señal PWM en alta para llegar desde una posición origen hasta una posición objetivo.

$$\text{Max}_{\text{vel}} = \frac{\text{Frec}}{10^5}$$

$$\text{Min}_{\text{vel}} = \frac{\text{Frec.}}{10^4}$$

Ecuación 38: Relación entre las constantes Max_{vel} y Min_{vel} y la frecuencia de trabajo del sistema.

Como se analizó el apartado 5.2.4 los giróscopos entregan una señal codificada en anchura PWM. El circuito diseñado para la captación de esta señal está formado por un detector de flanco y un circuito contador. El funcionamiento es muy sencillo y consiste en medir, contados en ciclos de reloj del sistema, la anchura del pulso en estado alto de la señal PWM de entrada.

Se pueden distinguir dos frecuencias de muestreo de la señal:

- La primera para medir la anchura del pulso en estado alto. En este caso la frecuencia de muestreo está limitada por la frecuencia del reloj del sistema, ya que se tomará una muestra por ciclo siendo el periodo de esta frecuencia la precisión de la medida.

- La segunda corresponde con el número de medidas que se pueden realizar por segundo y está limitada por la frecuencia base de la señal PWM enviada por el giroscopio. Esta señal está diseñada para manejar directamente un servo y por tanto su frecuencia está entre 50 y 80 Hz.

5.4 Resultados experimentales

El estudio experimental tiene dos etapas. En la primera se analiza la generación de patrones de movimiento mediante redes neuronales de pulsos. Mientras que el segundo caso constituye una aplicación practica con un robot humanoide.

5.4.1 Experimento de generación de patrones de movimiento con redes neuronales de pulsos para el control servomotor.

El experimento ha consistido en la simulación de una red neuronal de pulsos para la generación de un patrón de movimiento que controla un grupo muscular formado por dos músculos (agonista y antagonista). El grupo muscular siempre ejerce fuerzas contrapuestas entre sí produciendo un par de giro en una única articulación (véase la Ilustración 68). Cada músculo, a modo de integrador, ejerce una fuerza proporcional a la frecuencia de la señal recibida. Esta señal proviene de la red neuronal (véase la Ilustración 67) y se alterna en el tiempo, para producir un movimiento de giro repetitivo hacia un lado y hacia el otro.

En este experimento se analizan y estudian los siguientes apartados:

a) Las consecuencias en el movimiento de la articulación en función de las características (frecuencia, correlación temporal, etc.) de la señal introducida en cada músculo.

b) Modelo neuronal. ¿Qué características biológicas son necesarias simular en el modelo de neurona (inyección gradual de carga, decaimiento pasivo, etc.) para que la red pueda generar las señales adecuadas?

c) ¿Qué parámetros de la red producen la modificación de las características de la señal generada y por tanto cómo influyen en el movimiento final?

d) Aprendizaje, modificación de pesos y ajuste automático de las señales generadas.

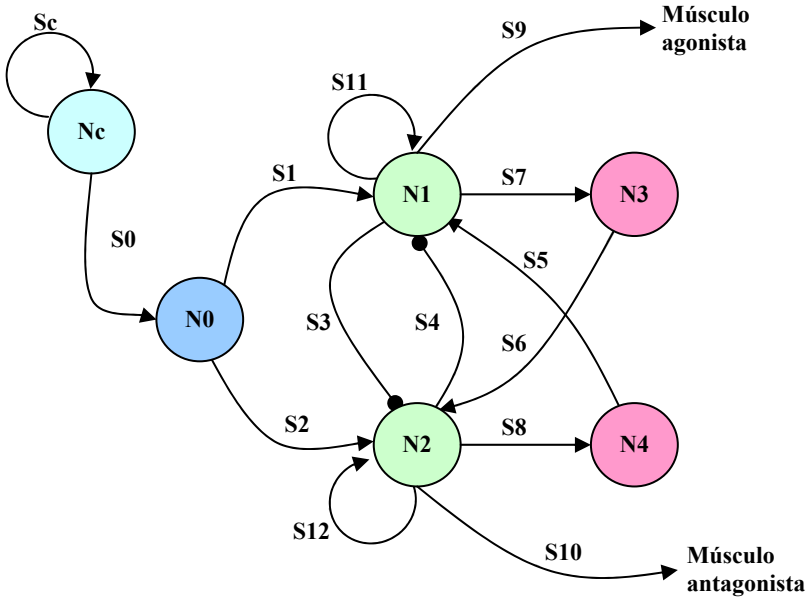


Ilustración 67: Modelo de red neuronal objeto de estudio. La flecha terminada en punta indica sinapsis excitatoria, mientras que acabada en círculo inhibitoria.

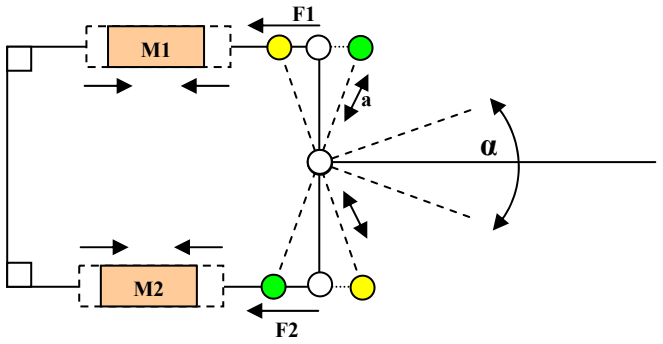


Ilustración 68: Músculo agonista y antagonista (M1 y M2) producen las fuerzas F1 y F2, permitiendo un giro α. Los segmentos a y b pueden cambiar de tamaño durante el giro para mantener las fuerzas F1 y F2 contrapuestas entre sí.

a) Las consecuencias en el movimiento de la articulación en función de las características (frecuencia, correlación temporal, etc.) de la señal introducida en cada músculo.

Como se ha mencionado inicialmente, la fuerza ejercida por cada músculo depende de la frecuencia de pulsos de la señal recibida. Nótese que como el tiempo en estado alto es constante, ya que se trata de un tren de pulsos, la fuerza ejercida por el músculo depende del tiempo en estado bajo de la señal de entrada.

La definición de un modelo muscular no es sencilla. Debido al alto grado de parámetros biofísicos implicados, tales como cambio en la fuerza ejercida en función de la deformación, fatiga, flexibilidad etc. Existen diferentes aproximaciones, métodos lineales que consideran que la fuerza ejercida es proporcional al grado de excitación muscular y no cambia con la longitud del mismo y modelos exponenciales o de colinas *"Hill-base muscle models"* que consideran la respuesta muscular (fuerza ejercida) frente a un estímulo, como un proceso de integración gradual dependiente del tiempo [55][137][133][78].

Una aproximación práctica definida para este experimento se describe en la Ecuación 39. Esta aproximación está basada en modelos de colinas más realistas, pero no modela la fuerza ejercida en función del grado de contracción debido a que la aplicación experimental practica, utiliza servomores en lo cuales se ha utilizado un modelo aproximado pero sin esta característica (véase apartados 5.2.6 y 5.4.3 para mayor detalle).

$$F(t) = F(t\text{-}t_k) + \frac{s(t)\cdot\left[F_{\max}\text{-}F(t\text{-}t_k)\right]}{F_{\max}} + \frac{F_{\max}\text{-}F(t\text{-}t_k)}{F_{\max}\cdot(t\text{-}t_l)} - \frac{F(t\text{-}t_k)}{\tau}$$

Ecuación 39: Modelo muscular para el cálculo del módulo de la fuerza.

En la Ecuación 39, *F(t)* es la fuerza ejercida por el músculo en el instante *t* . *s(t)* es la señal de pulsos de entrada. *Fmax* es la fuerza máxima que es capaz de ejercer el músculo. *tk* es el instante anterior del paso de simulación y τ es la constante de tiempo de decaimiento de la fuerza en ausencia de pulsos de entrada ("constante de relajación"). Por último *tl* es el instante de tiempo último en que se recibió un pulso en la señal de control *s*.

Cada músculo recibe su propia señal individual *s(t)*, por lo que la articulación girará en un sentido u otro en función del diferencial de fuerzas aplicado. Es importante destacar que el solapamiento de las señales de control en el tiempo provocará rigidez articular tanto mayo cuanto mayor coincidencia temporal y frecuencia exista entre las señales de control. Entiéndase por solapamiento, como la diferencia temporal t entre un evento de la señal *S1* con otro evento más cercano de la señal *S2*. Donde *t* ha de ser menor que la constante de tiempo del músculo y por tanto es dependiente de sus características intrínsecas (si no fuese así, la llegada de estímulos próximos en el tiempo en *S1* y *S2* no tendrían repercusión uno sobre el otro).

A la vista de las características de funcionamiento mencionadas, la frecuencia de la señal de pulsos de cada señal de control y el grado de solapamiento entre ambas, son los parámetros esenciales que han de poder ser modificados en la red neuronal de control.

b) Modelo neuronal. ¿Qué características son necesarias simular en el modelo (inyección gradual de carga, decaimiento pasivo, etc.) para que la red pueda generar las señales adecuadas?

El modelo neuronal utilizado, es un modelo práctico que incorpora inyección gradual de carga, contribución sináptica dependiente del potencial de membrana y decaimiento pasivo.

La descripción del modelo puede verse siguiendo la Ecuación 40. *V(t)* representa el potencial de membrana, *s(t)* y *wk(t)* es la contribución sináptica dependiente del potencial de membrana en el instante *t*. *tk* es el instante anterior de la simulación mientras que *tl* es el instante en que llegó el último evento. Por último, *Vmax* y *Vmin* son los potenciales máximos y mínimos admisibles.

$$V(t) = V(t\text{-}t_k) + \sum_1^j s(t)\cdot w_k(t)\cdot\frac{\left[V_{\max}\text{-}V(t\text{-}t_k)\right]}{V_{\max}} + \frac{V_{\max}\text{-}V(t\text{-}t_k)}{V_{\max}\cdot(t\text{-}t_l)} - \frac{V(t\text{-}t_k)\text{–}V_{\min}}{\tau}$$

Ecuación 40: Cálculo del potencial de membrana para el comportamiento subumbral.

La inyección gradual de carga (modelada con el tercer término de suma en la Ecuación 40) resulta importante en la sincronización de eventos. Esta característica se observa en las neuronas biológicas y permite que las consecuencias de la llegada de un conjunto de eventos tengan cierta tolerancia u "holgura temporal". Por ejemplo, supóngase una neurona con un potencial de membrana próximo al umbral de disparo. En el instante t llega un evento el cual sin inyección gradual de carga provocaría el disparo de la neurona en el mismo tiempo t, debido a que toda la carga se inyecta de forma instantánea. En el caso de disponer de inyección gradual de carga, la carga se inyecta a lo largo de un intervalo $t+k$ siendo k el instante de tiempo en que se supera el umbral de disparo en ausencia de nuevos eventos. Obsérvese que existe un intervalo de tiempo desde t hasta k donde la llegada de nuevos eventos provocaran alteraciones en el potencial de membrana llegando incluso a anular el disparo (en el caso de ser sinapsis inhibitorias). Por tanto la inyección gradual de carga permite tener en cuenta la sincronización entre eventos.

La contribución sináptica dependiente del potencial de membrana (modelada con el segundo término de suma en la Ecuación 40) ajusta la inyección de carga en el instante t en función del potencial del instante anterior $t-k$. Esto permite un aumento rápido del potencial cuando éste es bajo (justo por encima del umbral de reposo) y ajusta el potencial entre los valores máximos y mínimos. Esta característica permite que la neurona mantenga un nivel de potencial de membrana centrado entre el máximo y el mínimo durante su actividad (es una componente de normalización continua).

c) ¿Qué parámetros de la red producen la modificación de las características de la señal generada y por tanto como influyen en el movimiento final?

Las redes neuronales analógicas tipo Perceptrón y sus variantes, en general, trabajan como aproximadores funcionales universales. A priori una red de estas características podría generar señales analógicas que codificaran la fuerza de cada músculo. Sin embargo, en este tipo de redes es difícil simular aspectos de correlación temporal de señales. Esta característica es sumamente importante en experimentos con agentes reales (robots), debido a que influye directamente en el control de la dinámica del sistema. En el control diná-

mico, no sólo es importante el valor de la fuerza ejercida, sino en qué instante tiempo es aplicada.

Dentro de las redes neuronales basadas en pulsos, las características de las señales de salida dependen principalmente de:

- Características individuales de cada tipo de neurona (constante de tiempo, umbrales de disparo, etc.).
- Topología de la red.
- Relevancia o pesos de las conexiones (Ley de aprendizaje utilizada).
- Retardos sinápticos en las conexiones.

De entre todo el conjunto de características, sin duda, la topología de la red juega un papel fundamental para el establecimiento de la funcionalidad final. A modo de ejemplo, en el cerebro humano existen áreas con topologías muy definidas y especializadas, como es el caso del Cerebelo, involucradas principalmente en tareas de bajo nivel básicas o primarias (locomoción, coordinación de movimientos, etc) en vez de una "topología universal" con neuronas con características iguales capaz de aproximar cualquier funcionalidad.

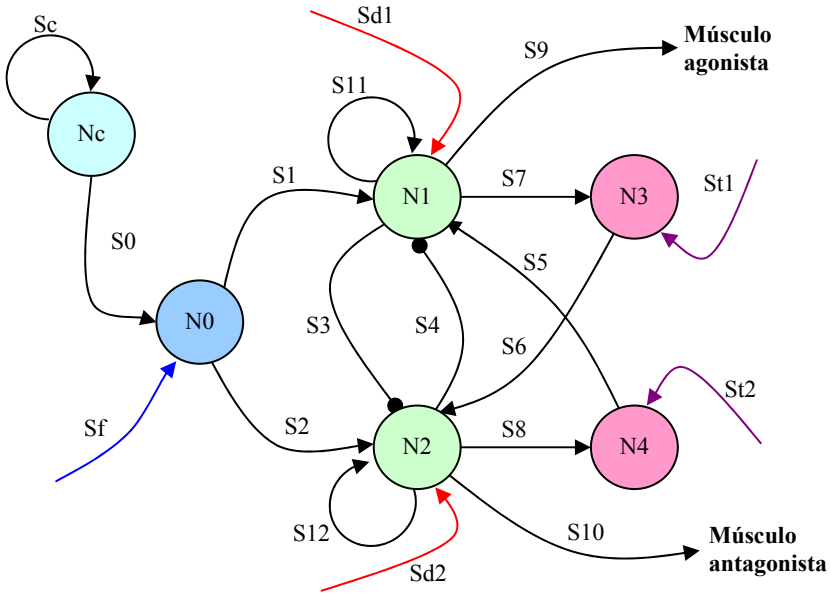


Ilustración 69: Modelo neuronal ampliado para soportar ajuste dinámico de la señales de salida

La topología básica de la red neuronal se muestra en la Ilustración 67 y en la Ilustración 69. Es una red que imita de forma práctica la funcionalidad de un generador central de patrones "*GPG General Pattern Generator*" responsable de producir de forma rítmica las señales que permiten a los seres vivos realizar tareas locomotrices (andar, correr, etc.) y de soporte vital (ritmo cardíaco, respiración, etc.).

La red está formada por seis neuronas (Nc, N0,…, N4). La neurona Nc funciona a modo de señal reloj, proporcionando un tren de pulsos a una frecuencia base. Esta frecuencia es dependiente de la constante de tiempo neuronal y del peso de la conexión de realimentación Sc.

La neurona N0, proporciona una salida de pulsos para las neuronas N1 y N2. Esta neurona funciona como divisor de la frecuencia base de la "señal de reloj" generada por N0.

N1 y N2 producen la señal de pulsos de salida para cada músculo (agonista y antagonista) y están involucradas en la alternancia del sentido de giro de la articulación. Cuando una de las neuronas N1 o N2 dispara, automáticamente se produce una inhibición en la neurona gemela, provocando que ésta disminuya su actividad. El tiempo que la neurona N1 o la neurona N2 permanecen disparando y por tanto el tiempo durante el cual la articulación girará en un sentido, queda ajustado por las constantes de tiempo de las neuronas N3 y N4 así como de los pesos de las sinapsis S5 y S6.

Inicialmente el potencial de membrana de la neurona N1 es ligeramente superior a la neurona N2 para producir el "arranque" inicial de la alternancia en el funcionamiento del circuito (en el caso de no recibir ninguna otra señal de actividad por las sinapsis sd1 o sd2). El potencial de membrana de N1 y N2 al inicio de la simulación define, por tanto, "el turno" en el sentido de giro de la articulación.

A modo de resumen las características de la señal de salida pueden ser modificadas del siguiente modo:

- **Aumentando la actividad en la sinapsis Sf:** Si se inyectan pulsos a través de esta sinapsis se aumenta la actividad de toda

la red y por tanto, aumenta la frecuencia las señales de salida (sinapsis s9 y s10). Por el contrario también disminuye el tiempo en la alternancia entre los estados activos de las neuronas N1 y N2. Factor a compensar actuando sobre N3 y N4, si se desea mantener el tiempo de alternancia entre N1 y N2 constantes.

- **Modificando la actividad en St1 y St2:** Aumentando la actividad en estas sinapsis, disminuirá el tiempo de alternancia entre N1 y N2, pero se mantendrá la frecuencia de disparo de las neuronas N1 y N2.

- **Alterando la actividad en las sinapsis Sd1 y Sd2:** Estas sinapsis permiten aumentar la frecuencia de forma directa de las señales de salida S9 o S10 y por tanto la fuerza ejercida por cada músculo. No obstante esta acción ha de ser compensada actuando en las sinapsis St1 o St2 en modo negativo para no alterar el tiempo de alternancia en la actividad entre las neuronas N1 y N2. Obsérvese que el grado de solapamiento (grado de rigidez articular) entre las señales puede ser alterado actuando sobre las sinapsis Sd1 o Sd2 contrarias a la neurona que está activa en ese intervalo de tiempo.

d) Aprendizaje, modificación de pesos y ajuste automático de las señales generadas.

La configuración de inicio preestablecida (configuración de pesos manual) en las sinapsis produce, el funcionamiento autónomo de la red. Es decir la red genera un patrón de movimiento "base" cuya frecuencia de salida y tiempos de alternancia quedan fijados inicialmente. Este patrón base puede ser alterado o "escalado" aumentando o disminuyendo la actividad de determinadas neuronas mediante sinapsis externas (véase las sinapsis externas Sf, Sd1, Sd2… de la Ilustración 69) o bien modificando los pesos de las conexiones.

En la biología, se observa que el aprendizaje, entendido como la modificación de pesos sinápticos ya sea a corto o largo plazo (STP, STD, LTD, LTP), está regido por la concordancia repetitiva entre el estímulo y la producción de un potencial de activación (existen distintas leyes de aprendizaje biológicamente plausibles como la ley de

Hebb [43]. Luego partiendo de unos pesos de la red que producen un patrón base es posible realizar un ajuste de los mismos para producir un patrón de movimiento mejor adaptado, modificando la actividad de determinadas neuronas de forma externa. Esa modificación externa trae implícitamente las características del aprendizaje y provendrá de otros circuitos nerviosos o bien directamente como señales sensoriales reactivas. Por tanto y a modo de hipótesis, directamente el entorno a través de señales sensoriales directas y señales sensoriales adaptadas mediante otros centros nerviosos modela y rige el aprendizaje.

A modo de ejemplo, la neurona N1 o N2 podría recibir una señal procedente de un circuito nervioso que se active cuando se intenta llegar más allá del límite máximo de giro de la articulación de tal modo que produzca una disminución en la actividad de la neurona y por tanto de la fuerza ejercida.

5.4.2 Resultados de simulación.

Se han simulado 15 segundos de tiempo real con una resolución de 1ms utilizando Matlab funcionando en un Pc con un procesador Pentium-4 a 1.6Ghz. Todo el procesamiento consume 12 segundos con lo cual el sistema permite funcionar en tiempo real. No obstante hay que tener en cuenta que solamente se simula un grupo muscular, en el caso de simular 16 grupos musculares, como en el robot descrito en el apartado 5.2, no podría funcionar en tiempo real. Esta característica ha motivado el desarrollo de los simuladores descritos en el Capítulo 3 y Capítulo 4.

La simulación se ha dividido en tres etapas (a, b y c), más una cuarta (d) con un análisis aclaratorio sobre criterios de escalado, magnitud de fuerzas y posición. Estas etapas son los siguientes:

- **a)** Simulación del modelo muscular (Escalado, magnitud de fuerzas, velocidad y posición).
- **b)** Simulación de la red neuronal propuesta.
- **c)** Simulación conjunta (red neuronal/sistema muscular).
- **d)** Escalado, magnitud de fuerzas, velocidad y posición.

a) Simulación muscular:

El resultado de la simulación del modelo muscular puede verse siguiendo la Ilustración 70. En ella se observan las señales de control agonista y antagonista (gráfica 1 y 2). El patrón inicial se ha generado de forma artificial para observar el comportamiento del modelo. De este modo, la señal para el músculo agonista comienza a una frecuencia de 12 Hz, posteriormente baja a 5Hz y finalmente a 2Hz. Por otro lado, la señal antagonista se mantiene a una frecuencia de 12 Hz durante un intervalo de 2.5sg. De esta forma se configura un patrón de señales genéricas representativas del comportamiento.

Las gráficas 3 y 4 (siguiendo con la Ilustración 70) muestran la fuerza ejercida por cada músculo tras el proceso de integración de la señal de pulsos. La gráfica 5 representa el módulo de la fuerza equivalente (diferencial de fuerzas: nótese que al ser las fuerzas paralelas entre sí, el módulo de la fuerza equivalente coincide con la resta de los módulos). El signo indica el sentido de la fuerza equivalente.

La gráfica 6 muestra la velocidad durante todo el movimiento, mientras que la gráfica 7 indica la posición. Obsérvese que debido a la forma del patrón de las señales de control, en este caso, la posición origen y destino no son las mismas.

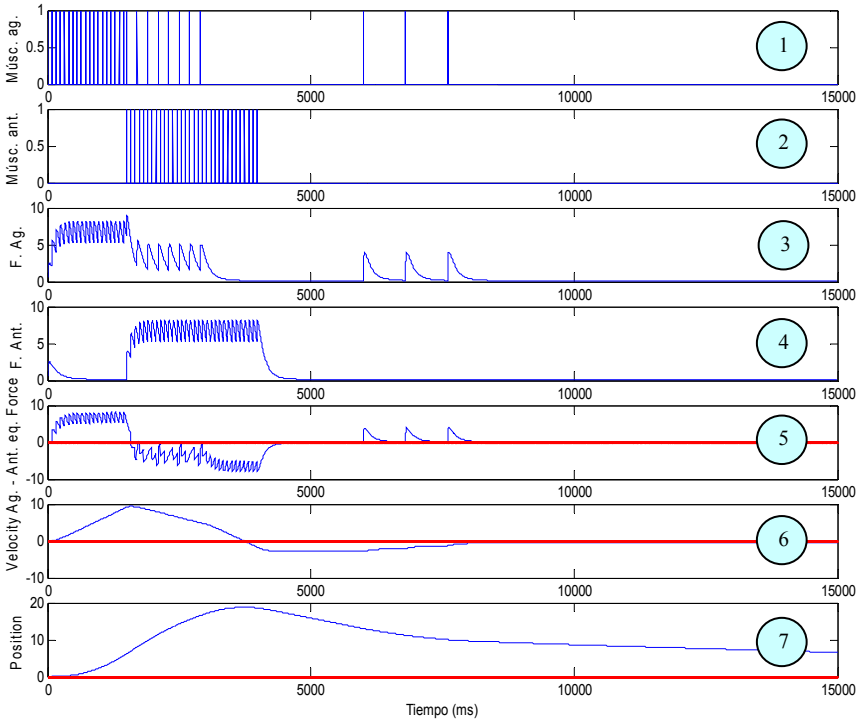


Ilustración 70: Resultado de la simulación del modelo muscular donde se observa; las señales de pulsos de control (1 y 2), la fuerza ejercida en cada músculo (3 y 4), la fuerza diferencial resultante (5), así como la velocidad y la posición durante el movimiento (6 y 7). Obsérvese el cambio de velocidad justo en el solapamiento de las señales de control (véase intervalo de velocidad entre el segundo 2 y el 5). Nótese que si la fuerza vale cero la velocidad permanece constante debido a que no se ha simulado rozamientos.

b) Simulación de la red neuronal propuesta:

El resultado de la simulación para la generación de un patrón de movimiento base se puede ver en la Ilustración 71 mientras que la evolución de los potenciales de membrana de las neuronas Nc…N4 se muestra en la Ilustración 72. Se ha establecido una frecuencia de disparo de la neurona SC a 1Khz y se ha simulado un total de 15 segundos con una resolución temporal de 1 ms. Las gráficas 1...10 de la Ilustración 71 muestran la actividad en las sinapsis Sc…S8. Obsérvese la alternancia en la actividad en las gráficas 9 y 10, regida por la actividad en las gráficas 7 y 8.

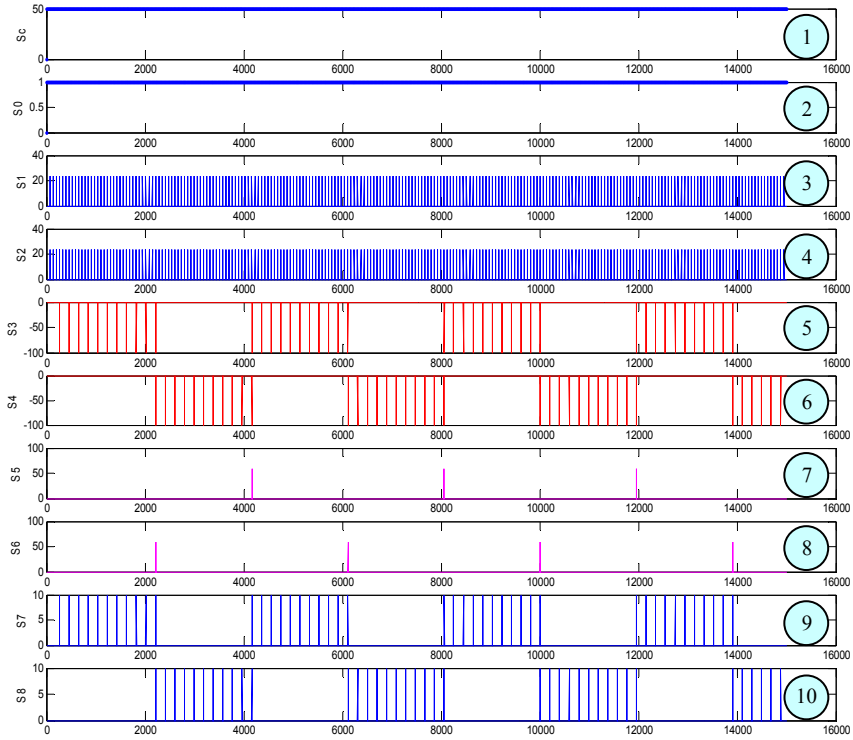


Ilustración 71: Representación de la actividad en las sinapsis de la red neuronal propuesta en la Ilustración 69.Obsérvese la alternancia en la secuencia de disparo en las sinapsis S7 y S8.

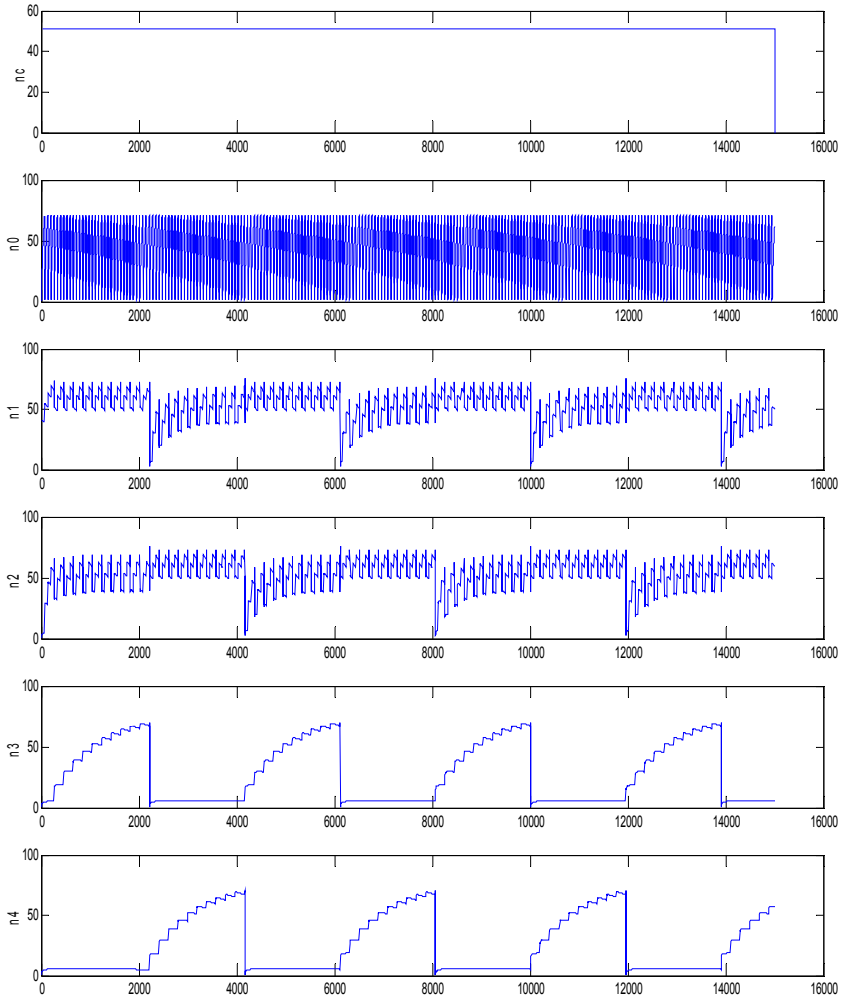


Ilustración 72: Conjunto de valores de potenciales de membrana para cada una de las neuronas *(nc…n4)* de la red mostrada en la Ilustración 67 durante la simulación. Nótese que debido a que la neurona *nc* dispara con una frecuencia de 1Khz y la resolución ser de 1ms se observa un nivel de actividad continua en toda la gráfica.

c) Simulación conjunta (red neuronal/sistema muscular):

La Ilustración 73 y la Ilustración 74 muestran el resultado de la utilización del patrón de movimiento generado por la red neuronal y el modelo muscular propuesto. En el caso de la Ilustración 73 el patrón generado es simétrico. Es decir los tiempos de espera configurados mediante las neuronas N3 y N4 son los mismos. Sin em-

bargo en la Ilustración 74 se ha incrementado el tiempo de espera configurado en N4 lo que produce un patrón asimétrico en las fuerzas ejercidas.

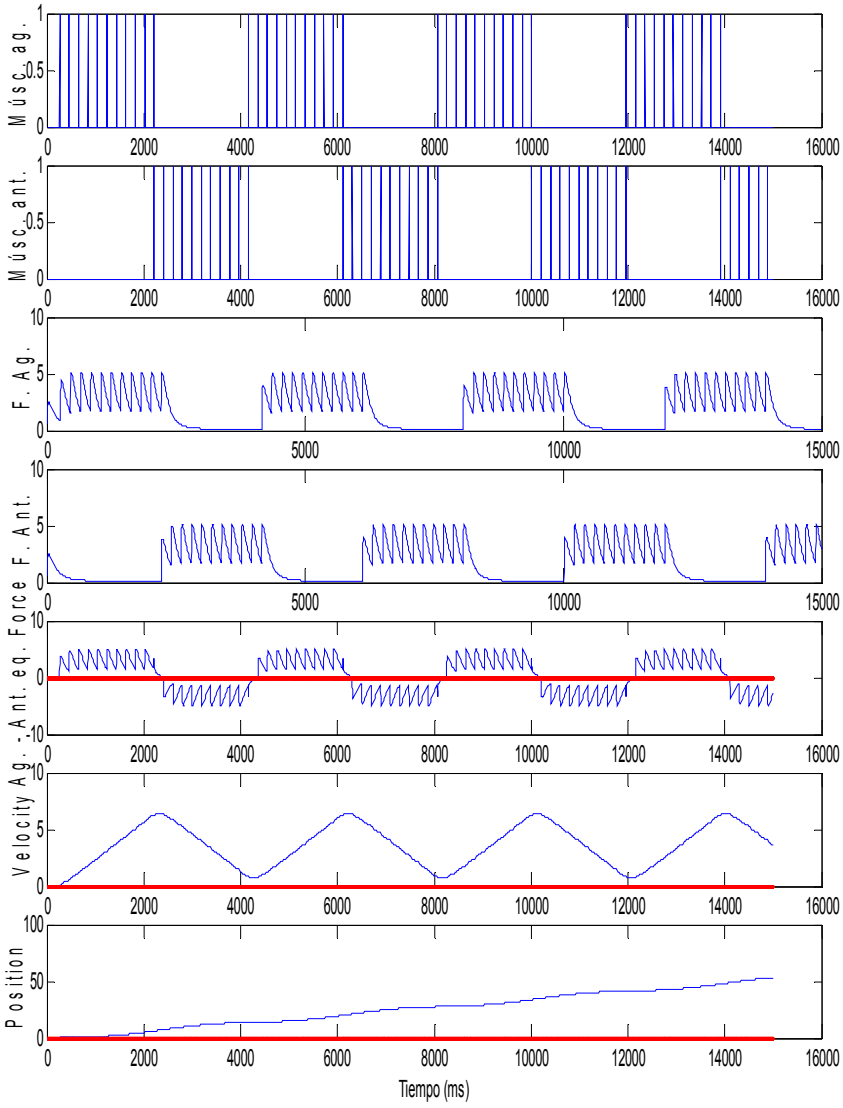


Ilustración 73: Resultado de la simulación del patrón de movimiento generado por la red neuronal y las fuerzas, velocidad y posición resultado. De arriba hacia abajo, fuerza agonista, fuerza antagonista, fuerza equivalente, velocidad y posición.

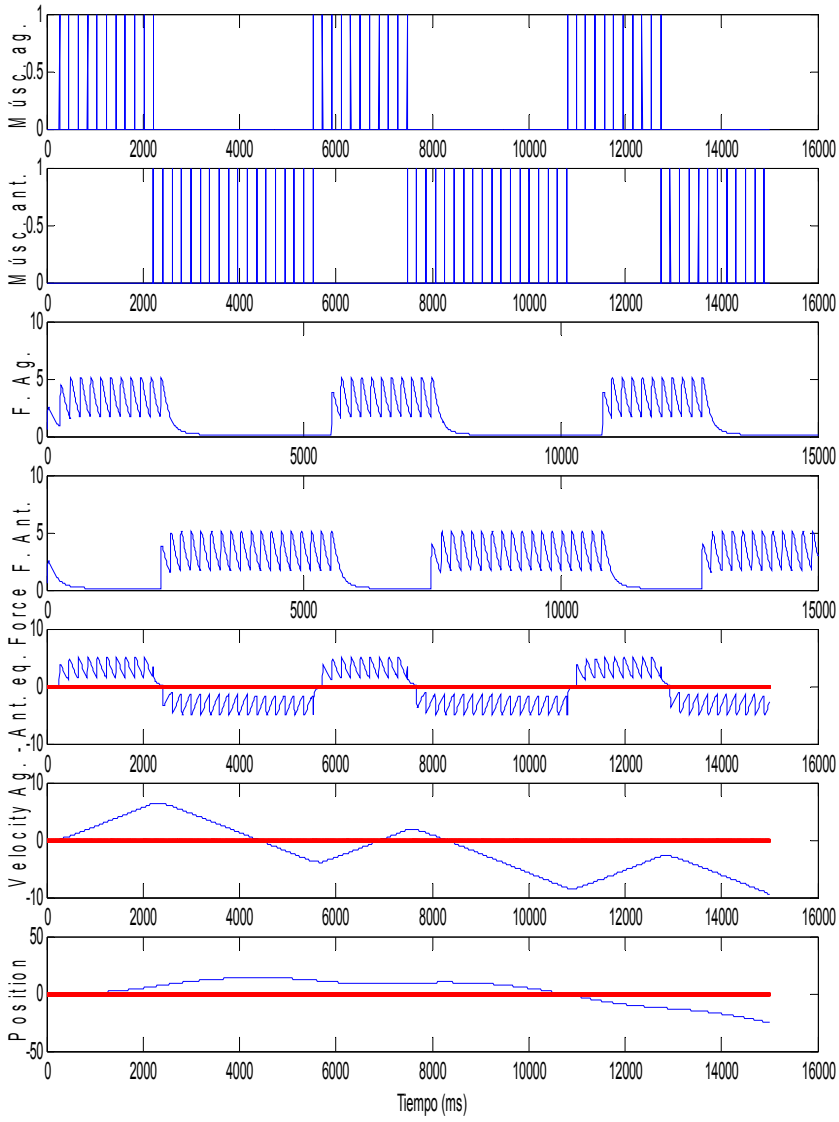


Ilustración 74: Resultado de la simulación con un patrón asimétrico. De arriba hacia abajo, fuerza agonista, fuerza antagonista, fuerza equivalente, velocidad y posición.

d) Escalado, magnitud de fuerzas, velocidad y posición:

Cada pulso de entrada hacia el músculo produce un incremento en la fuerza ejercida durante cierto intervalo de tiempo. El valor de dicho incremento depende de las características intrínsecas de cada

músculo. Una forma práctica de definir este valor consiste en ajustar los valores de fuerza máximo y mínimo del modelo propuesto en la Ecuación 39, donde implícitamente queda definido el incremento de fuerza con cada pulso de entrada. Por ejemplo, la gráfica 3 de la Ilustración 70 muestra un incremento de fuerza en torno a 3.8N (Newton) en el segundo 6 de simulación para un rango máximo comprendido entre 0 y 10N, como consecuencia de la llegada de un evento (gráfica 1 de la Ilustración 70).

La gráfica 5 de la muestra la diferencia entre los módulos de las fuerzas aplicadas. Recordemos que la distancia entre el eje de giro y el punto de aplicación de las fuerzas es igual para ambas (esta distancia cambia con el giro de la articulación), que la fuerzas son paralelas entre si y además se encuentra en el mismo plano de aplicación (véase la Ilustración 75). Esta fuerza Fe no cambia de valor debido al giro de la articulación siempre que las fuerzas ejercidas por ambos músculos no cambien. Sin embargo si lo hace el par de giro resultante *cpr* debido a la modificación de los ángulos durante la rotación.

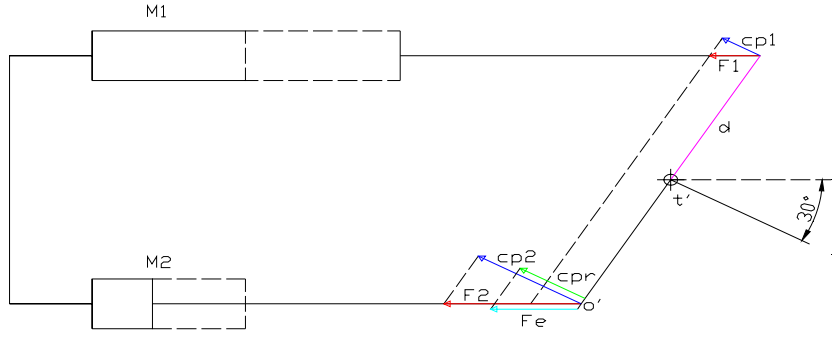


Ilustración 75: Cálculo de la fuerza equivalente y pares de giro que permiten la rotación sobre el punto t'. Las fuerzas F1 y F2 (color rojo) son producidas por los músculos M1 y M2. cp1 y cp2 (color azul) son las componentes de F1 y F2 o los pares debidos a las fuerzas F1 y F2 a una distancia d. cpr es el par resultante calculado como la diferencia entre cp1 y cp2. A su vez cpr es componente de Fe o fuerza equivalente cuyo módulo es igual a la diferencia entre F1 y F2 debido a que son fuerzas contrapuestas. Observación: cpr, cp2, F2 y Fe, tiene el mismo punto de aplicación o'. Se han dibujado desplazadas para aclarar la ilustración.

La gráfica 6 de la Ilustración 70 muestra la velocidad calculada como la integral de la fuerza equivalente entre la masa puntual. Recordemos que la velocidad es la integral indefinida de la aceleración

y que ésta es igual a la fuerza aplicada entre la masa. Observando la Ilustración 75, el vector velocidad tendría su punto de aplicación en o'.

Por último, la gráfica 7 (Ilustración 70), muestra la posición del punto de la articulación o' obtenida a partir de la integral de la velocidad. Obsérvese que los valores de posición del punto o' quedará dentro de los límites de giro siempre y cuando no sea superior a la distancia entre el punto t' y cada músculo.

5.4.3 Integración real con el sistema de control del robot bípedo Robonova-I

Se ha probado la planificación de trayectorias y la conversión a comandos motores. La información sensorial obtenida del robot se ha traducido a pulsos neuronales y tras ser procesadas por el motor de simulación descrito en el capítulo 4, se han convertido a señales PWM por el circuito de adaptación de señales (véase las Ilustración 61, Ilustración 63 y Ilustración 76). Este flujo de información queda resumido en la Ilustración 76. La tarea de planificación de trayectorias se desarrolla en la corteza cerebral motora.

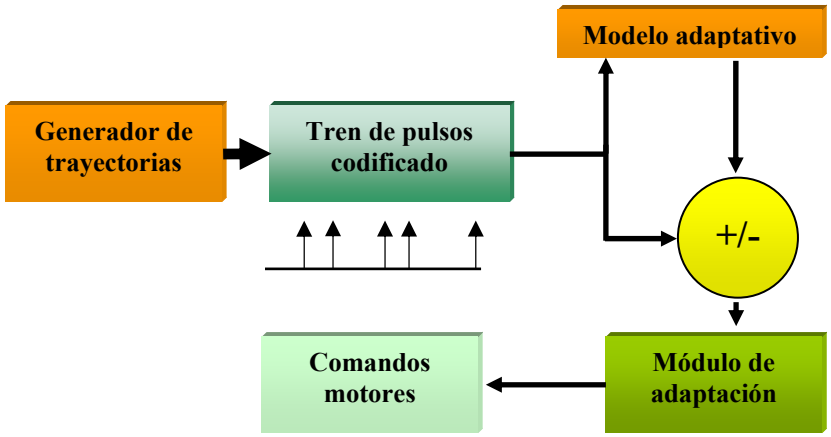


Ilustración 76: Esquema de interconexión para la obtención de los comandos motores.

En estos experimentos se ha realizado con pulsos neuronales para permitir la posterior incorporación de otros centros nerviosos como

el Cerebelo. Con estos experimentos se ha demostrado la viabilidad del control de robots con estructuras funcionales basadas en pulsos neuronales computados por motores de simulación como los descritos en los capítulos 3 y 4.

El experimento ha consistido en definir y ejecutar un conjunto de patrones de movimiento (levantarse, sentarse, mantener el peso sobre un pie). Entiéndase por patrón de movimiento como una secuencia de valores correlacionados en el tiempo y que describen los movimientos del robot. En el caso del robot objeto de estudio cada patrón de movimiento está formado por:

- Un conjunto de posiciones angulares objetivo para cada motor (16 motores en total) ordenados temporalmente.
- Un conjunto de velocidades para cada motor y para cada intervalo de dos posiciones objetivo, durante la ejecución del patrón.

La división en patrones se ha realizado estableciendo la premisa de que el estado inicial del movimiento y el estado final sean estados dinámicamente estables. (Entiéndase por estado estable, como el conjunto de posiciones de todos los motores que producen una postura del robot, donde la resultante de todas las fuerzas aplicadas sobre el centro de gravedad es igual a cero). Por ejemplo, un patrón de movimiento que permita al robot andar, parte del un estado estable “de pie erguido” y termina en un estado estable “de pie erguido”, tras dar un paso (con un conjunto de estados intermedios inestables).

5.4.4 Descripción del experimento

Es importante destacar la correcta secuencialización temporal en la ejecución de los diferentes patrones de movimiento. A este nivel, el sistema encargado de general las diferentes secuencias, puede asemejarse a un circuito digital secuencial. En estos circuitos se calcula el estado siguiente en base al estado actual y al conjunto de entradas presentes. En este ejemplo, el estado actual lo formaría la posición angular de cada motor, mientras que las entradas podrían ser la velocidad con que queremos que se desplace, el estado

objetivo a alcanzar, la posición del robot con respecto a un punto referenciado con el entorno, la fuerza gravitacional sobre centro de gravedad, etc. Es decir, multitud de informaciones sensoriales que permiten seleccionar el siguiente conjunto de sub-posiciones óptimas para alcanzar un estado estable.

La biología dispone de diferentes centros nerviosos dedicados a esta tarea. Para el caso de movimientos locomotrices, donde las acciones son repetitivas (por ejemplo caminar), el control se realiza de manera automática (refleja) con poca o ninguna intervención constante de centros nerviosos superiores [62]. No obstante la necesidad continuada de adaptación a las condiciones del entorno inmediato, exige un ajuste del movimiento óptimo y continuo [57]. Es decir, los movimientos locomotrices se ejecutan de forma automática y los centros de control nervioso superiores intervienen cuando es necesario.

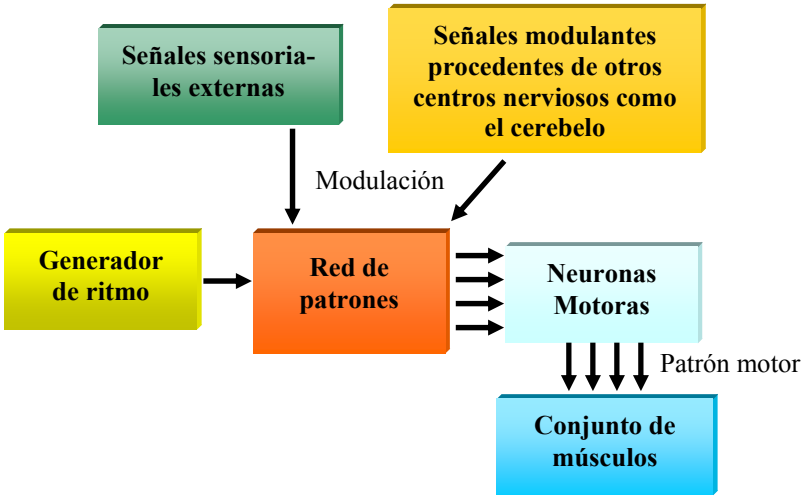


Ilustración 77: Esquema simplificado de un generador central de patrones CGP. Existen circuitos nerviosos dentro del CGP que, de forma automática y sin ninguna estimulación sensorial externa, generan señales rítmicas para la ejecución de los diferentes patrones de movimiento.

Uno de los centros nerviosos involucrados en la generación de movimientos rítmicos es el CPG "*Generador central de patrones*". Este centro nervioso controla tareas como la respiración o la locomoción. El funcionamiento, como se describió anteriormente, es parecido a un circuito secuencial, pero tiene ciertas diferencias. El

CPG genera de forma rítmica patrones de actividad básicos sin necesidad de la existencia de entradas sensitivas procedentes de estímulos externos. Además el patrón que genera difiere (en mayor o menor medida dependiendo de la especie) del patrón locomotor final que es aplicado a los músculos. El patrón locomotor básico generado por el CGP se modifica por la información sensitiva procedente señales sensoriales externas y de otros centros nerviosos dedicados a tareas específicas (véase la Ilustración 77). De este modo el patrón de movimiento se "adapta sutilmente" a las circunstancias cambiantes del entorno tanto en forma como en tiempo.

5.4.5 Resultados obtenidos

La información intercambiada con el robot son la respuesta de los acelerómetros (balanceo y cabeceo), señal de los interruptores de contacto en los pies, posiciones angulares y velocidades con las siguientes particularidades:

1. Configuración de velocidad durante la ejecución del patrón: Se configura la velocidad máxima al inicio de cada movimiento (válida para cada par de posiciones objetivos e igual para todos los motores) siendo constante durante dicho intervalo. Esto implica que la velocidad en el arranque inicial y el final la establece el controlador interno de cada servo (siguiendo la Ilustración 60).

2. La ejecución de los patrones de movimiento a nivel motor se hace en ciclo abierto. Sólo existe realimentación con los acelerómetros y los sensores de contacto de los pies. Esto es debido a la desconexión del sistema de control local durante la lectura de las posiciones.

3. La Ilustración 78 muestra la numeración utilizada para la identificación de los diferentes motores del robot. Es importante hacer notar, para la mejor compresión de las ilustraciones mostradas en esta sección, la referencia de posición en cero de cada servo. Es decir en el caso de movimientos simétricos cada par de servos gemelos tiene posiciones diferenciales. Por ejemplo, si el servo número 4 tiene la posición 120, el servo gemelo (servo 12) estará en la posición 200-120 = 80.

4. Otra observación a destacar es la calibración. Cada servo difiere en ángulo de giro para una misma coordenada objetivo. Esta calibración se tiene en cuenta en la generación de los patrones de movimiento, pero se ve reflejado en la Ilustración 82, Ilustración 83 y en la Ilustración 87, donde los movimientos gemelos se ven ligeramente diferentes.

5. Los interruptores de contacto disponen de un filtro para la eliminación de rebotes por hardware. El filtro simplemente comprueba que la señal se mantenga en estado alto durante al menos 1 milisegundo, reestableciendo esta condición cuando la señal pasa ha estado bajo por la detección de un rebote o la desconexión del contacto.

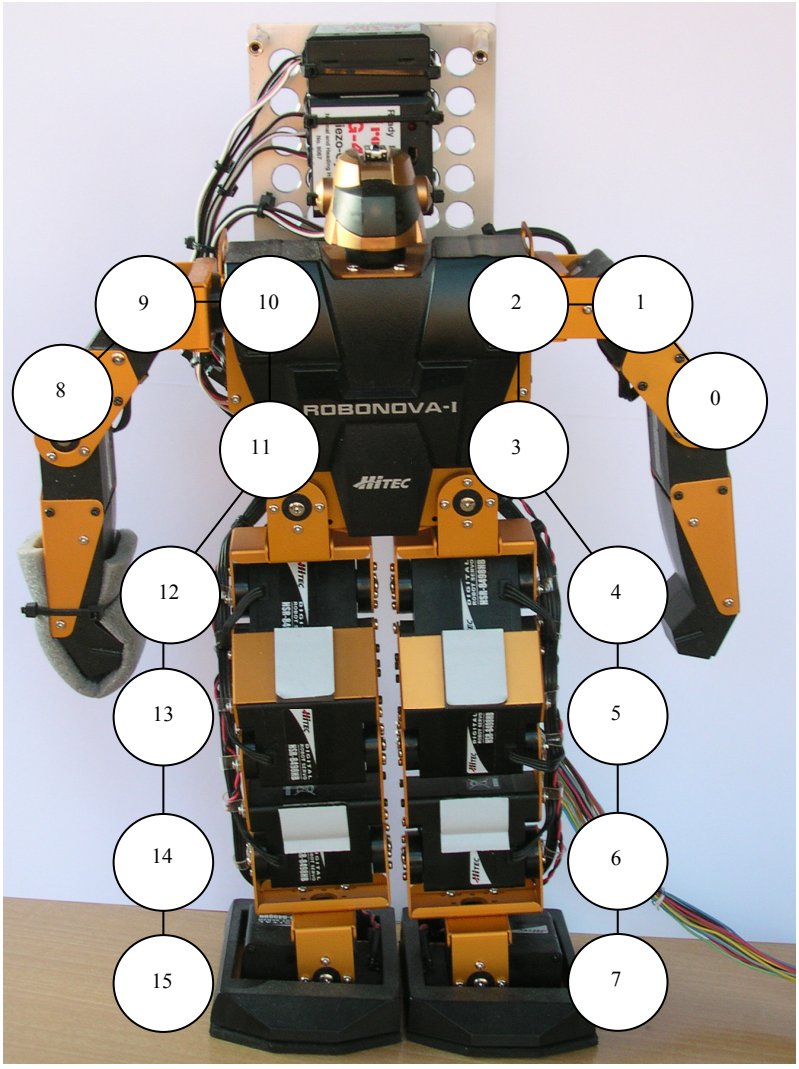


Ilustración 78: Numeración utilizada para la identificación de motores.

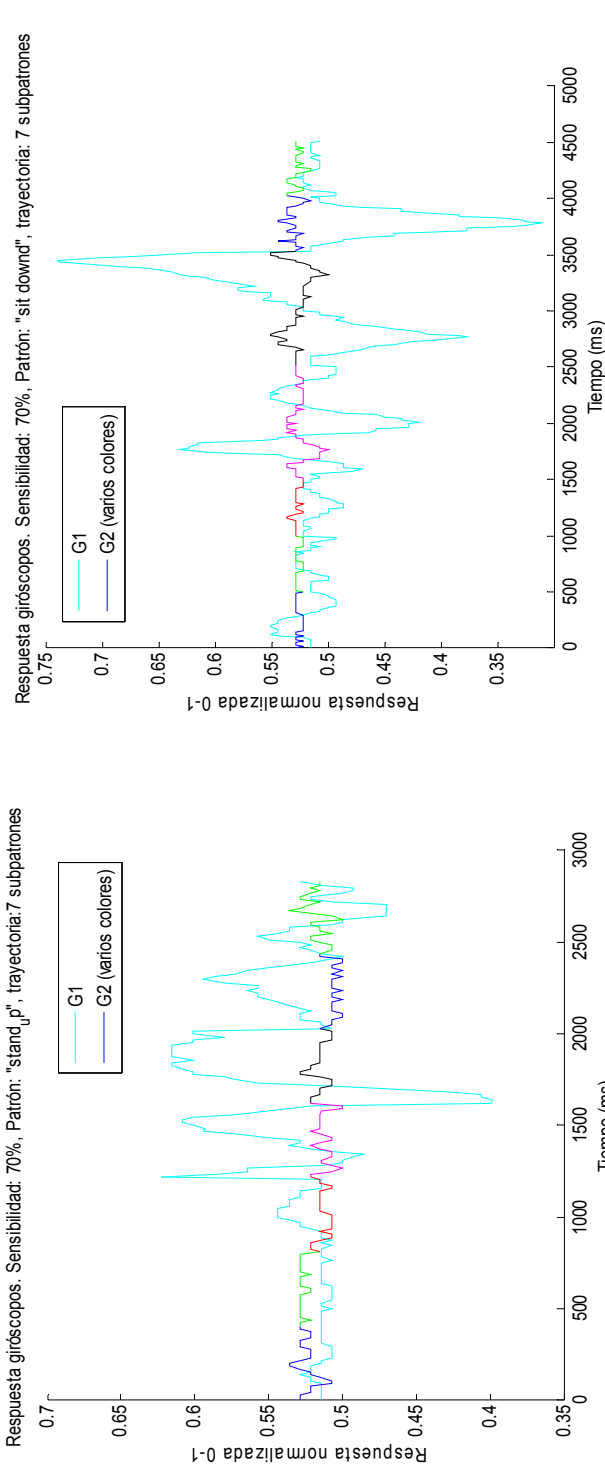


Ilustración 79: Respuesta de los giróscopos durante la ejecución de los patrones de movimiento "levantarse" (izquierda) y "sentarse" (derecha). Cada trozo de color significa la ejecución de un sub-patrón, formado por las coordenadas objetivo. Es decir, cada color muestra la respuesta de los giróscopos durante el intervalo entre el conjunto de coordenadas origen y el conjunto de coordenadas objetivo para cada servo (16 en total). La ganancia se ha ajustado al 70% para obtener una respuesta lo más sensible posible, sin la inclusión de demasiado ruido.

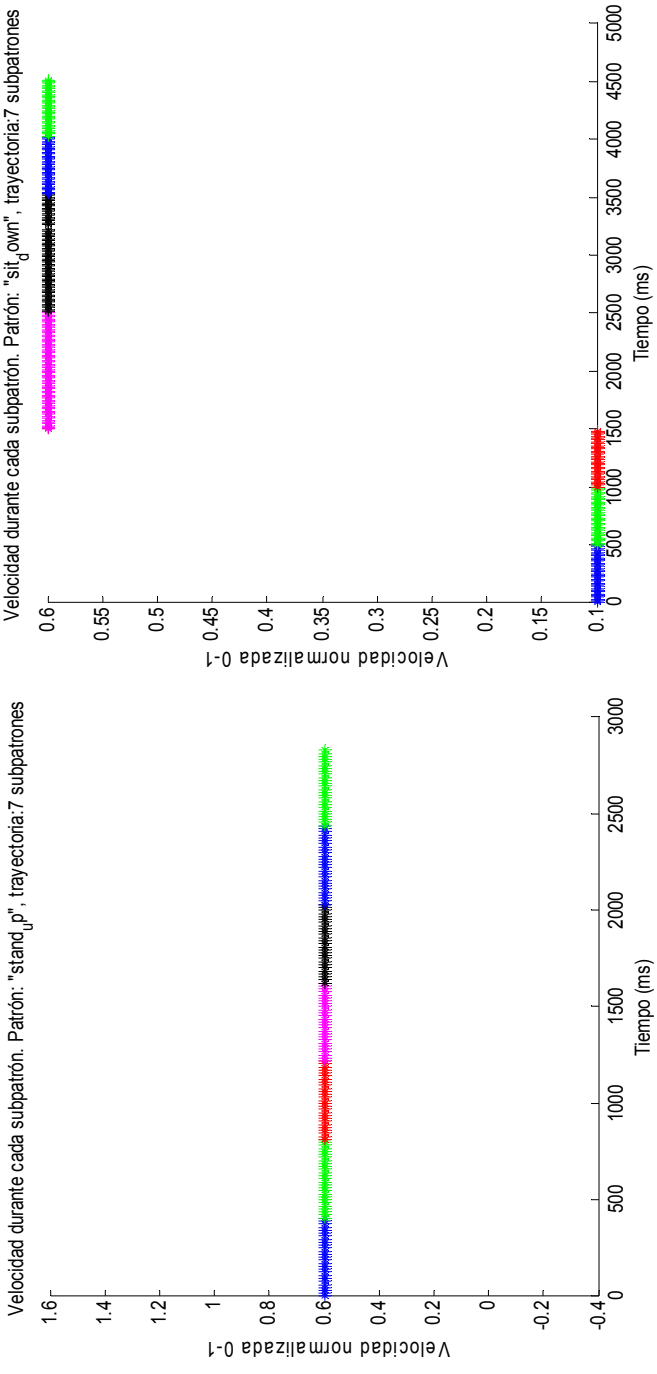


Ilustración 80: La gráfica muestra la velocidad configurada durante la ejecución del patrón de movimiento. Cada cambio de color muestra un cambio en las posiciones angulares objetivo. El rango de velocidades esta normalizado entre 0 y 1.

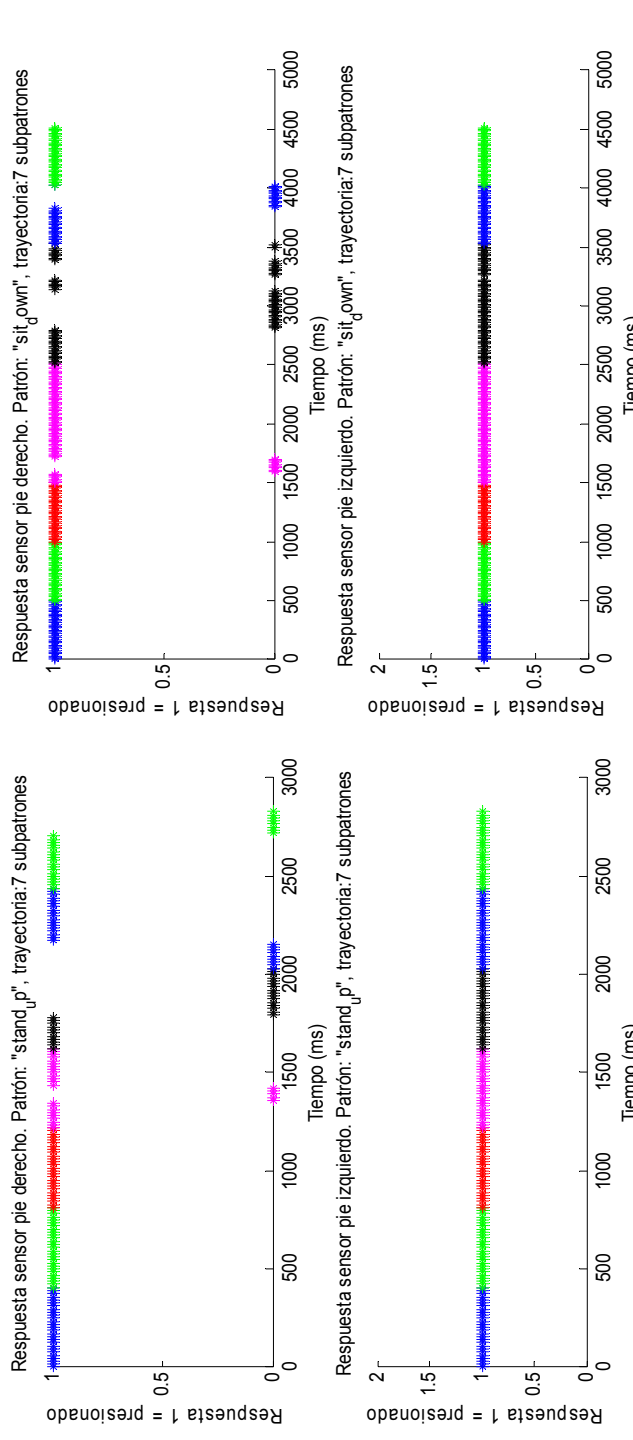


Ilustración 81: Respuesta de los micro-interruptores de los pies durante la ejecución del patrón de moviendo. El color muestra el tiempo durante el cual se mantienen las posiciones angulares objetivo. En este caso durante este movimiento el robot sufre un ligero balanceo perdiendo presión en unas de las piernas (gráfica superior).

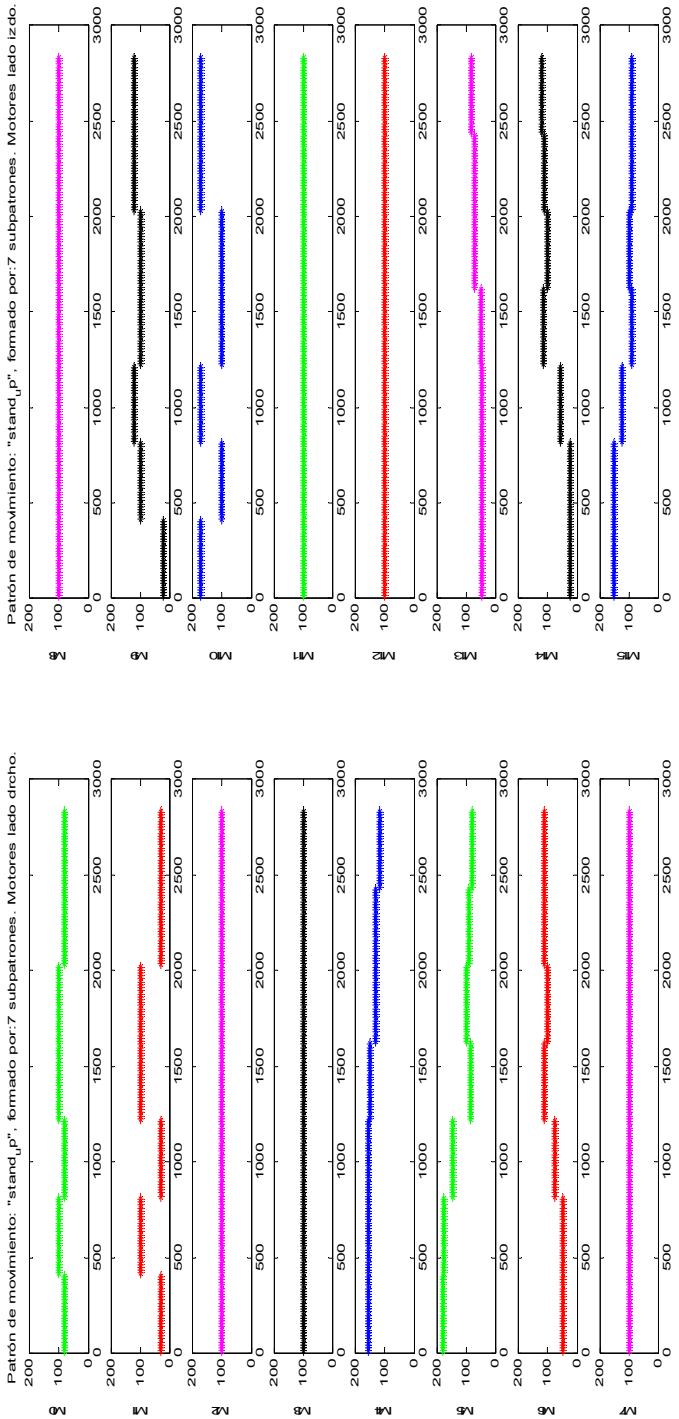


Ilustración 82: La figura muestra las coordenadas angulares en grados, enviadas a cada motor durante la ejecución del patrón de movimiento.

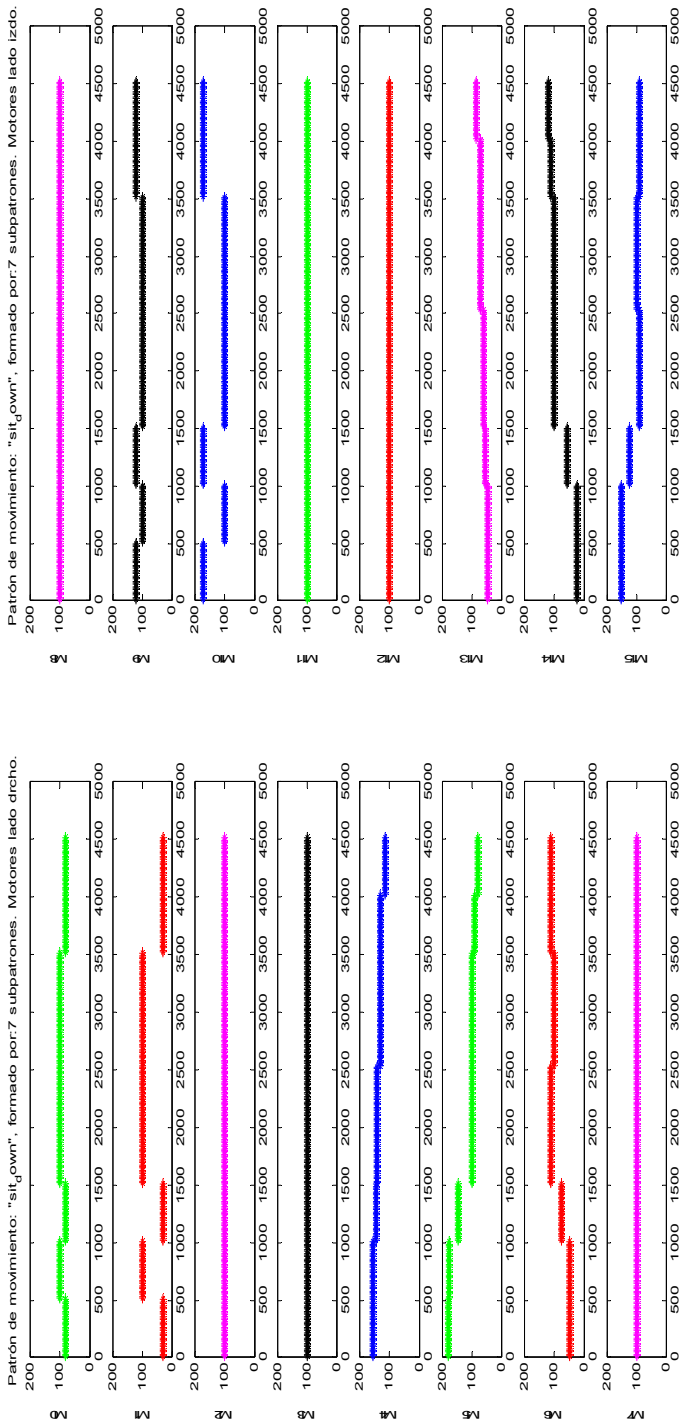


Ilustración 83: Coordenadas angulares enviadas a cada motor durante la ejecución del patrón de movimiento para el conjunto de motores del lado izquierdo y derecho.

Patrón compuesto (levantarse, mantener el peso sobre un pie, sentarse)

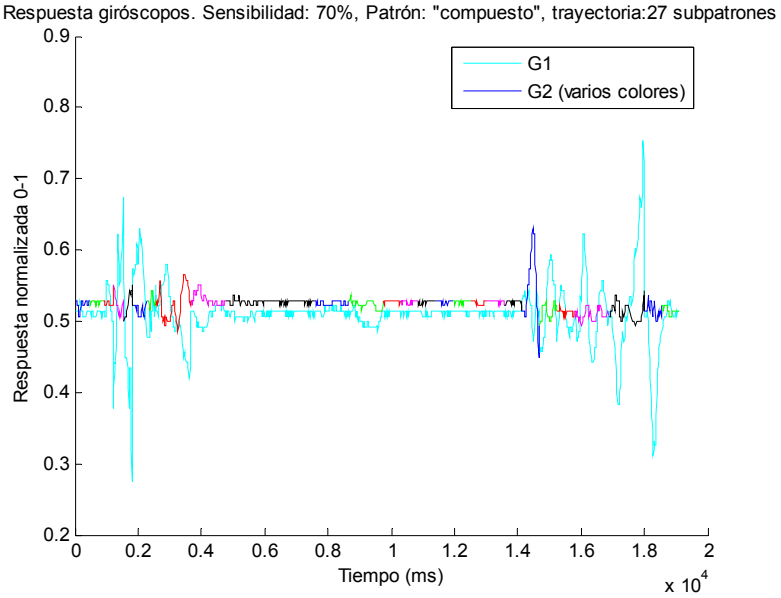


Ilustración 84: Respuesta de los giróscopos G1 y G2 (cabeceo y balanceo) compuesta de varios patrones de movimiento (levantarse, mantener peso sobre un pie y sentarse.

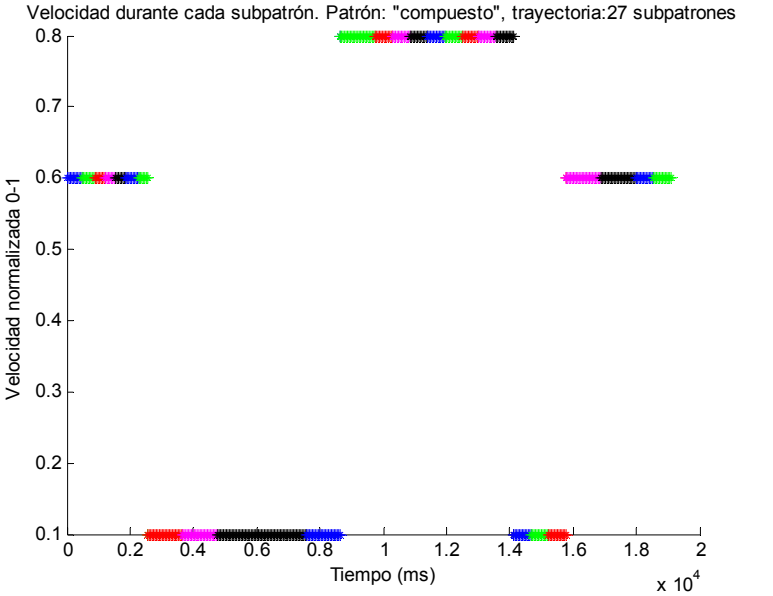


Ilustración 85: Conjunto de velocidades para el conjunto de motores, durante la ejecución del patrón de movimiento compuesto.

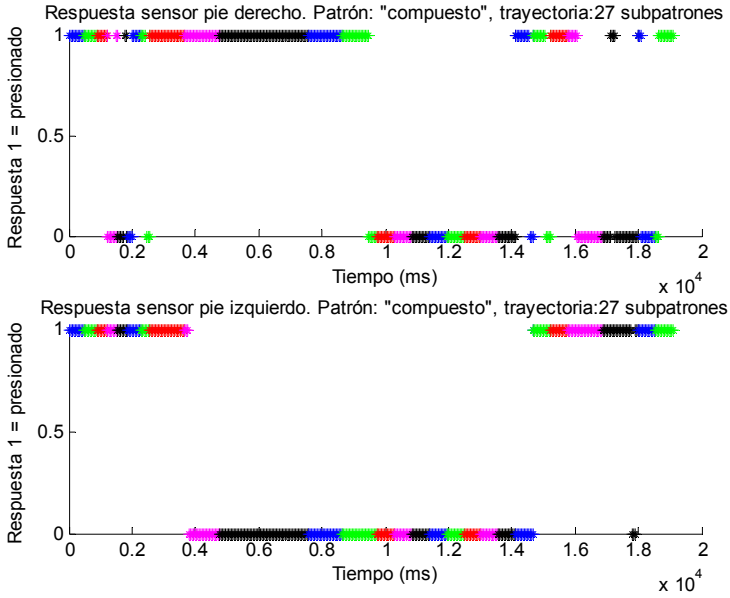


Ilustración 86: Respuesta de los interruptores de contactos de los pies durante la ejecución de los patrones de movimiento. Las vibraciones estructurales y el propio movimiento en si, produjeron modificaciones en la presión de cada pié.

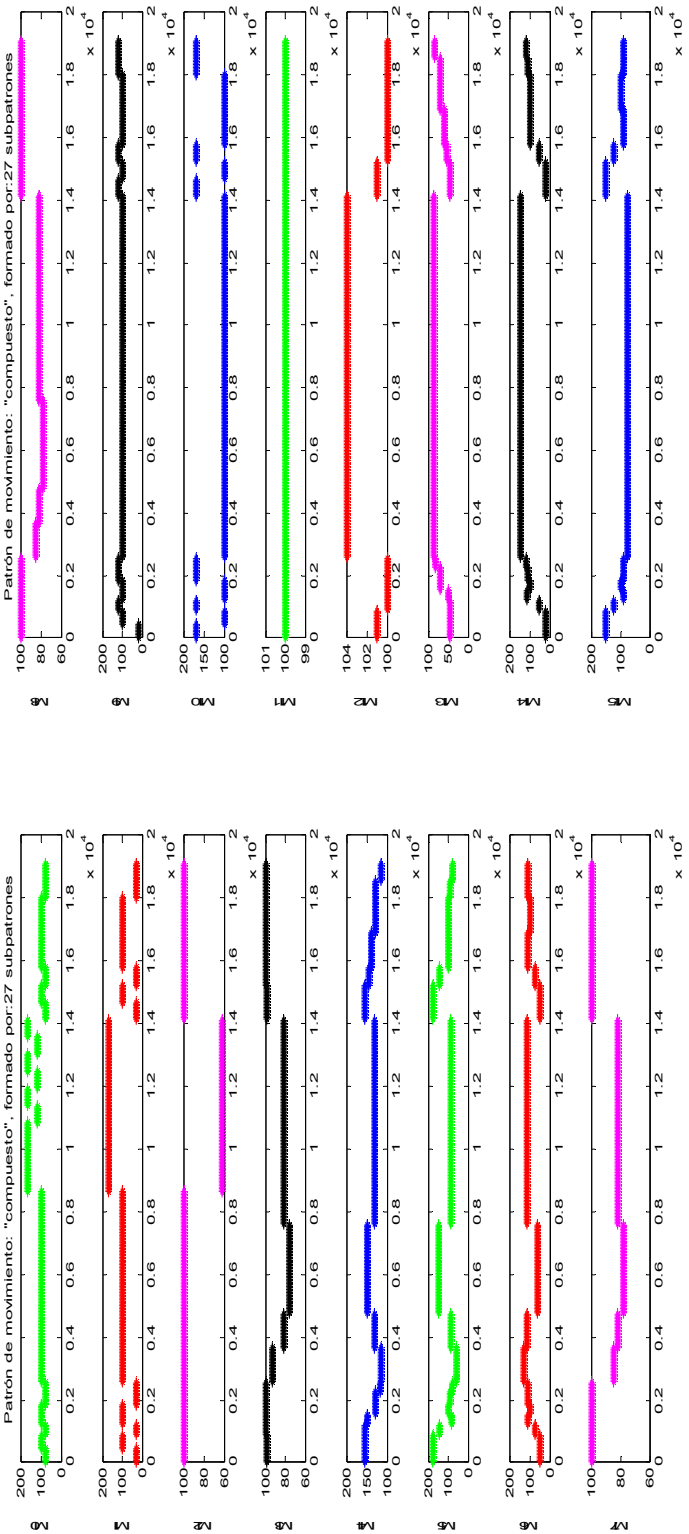


Ilustración 87: Conjunto de coordenadas angulares enviadas a cada motor durante la ejecución del patrón de movimiento compuesto. Obsérvese como los motores gemelos tiene posiciones diferenciales para los movimientos simétricos.

5.5 Conclusiones

Las estrategias de control de robots como agentes inteligentes están íntimamente relacionadas con la base tecnológica utilizada (entiéndase por tecnología basada en silicio, tecnología basada en el carbono, etc.). En este ámbito los sistemas biológicos han optado, como mecanismo eficiente de control, por sistemas nerviosos basados en células que intercambian información mediante pulsos eléctricos. No obstante hay que destacar que ésta es una característica beneficiosa bajo el conjunto de restricciones impuestas por las propiedades que rigen el funcionamiento de los sistemas vivos (entiéndase como el conjunto de propiedades químicas, electroquímicas, mecánicas, etc.). Por tanto, esta forma de comunicación y manejo de la información no tiene porqué ser igual de óptima en los sistemas basados en silicio.
Probablemente en un futuro el comportamiento "inteligente" de los sistemas artificiales esté basado en las propiedades funcionales de los sistemas biológicos, pero posiblemente, y a modo de hipótesis, no tendrán que reproducir exactamente su arquitectura, simplemente su funcionalidad.
Partiendo de esta premisa la solución a la búsqueda de un sistema de control artificial igual de eficiente que uno biológico pasa por el conocimiento del conjunto de reglas que rigen el traspaso funcional entre tecnologías (entiéndase entre la biología y el silicio). Este conocimiento es muy complejo y ha motivado y justificado la construcción y adaptación de los robots descritos en éste capítulo cuya experiencia y aportaciones quedan resumidas a continuación:

- Se ha construido y descrito dos plataformas robóticas (un brazo robot de dos grados de libertad y un humanoide) adaptadas para su control mediante redes neuronales de pulsos.

- Se ha descrito un sistema de control evolutivo basado en algoritmos genéticos que ha permitido obtener una visión global fuera de las alternativas de control tradicionales. Esta plataforma no sólo ha contribuido como base de experimentación para el desarrollo de este trabajo de Tesis, sino que ha servido como plataforma de experimentación y prueba de una red neuronal basada en el cerebelo [103] [7] [109] ob-

servando cómo es posible construir sistemas de control funcionales inspirados en la biología.

- En una segunda etapa, se ha puesto de manifiesto el conjunto de dificultades que aparecen en el control de sistemas físicos mediante redes neuronales basadas en pulsos, en especial con un robot bípedo compuesto por 16 servo-motores, tres giróscopos y dos sensores de contacto.

- Se ha descrito el hardware de control basado en circuitos FPGA que permite el manejo del robot bípedo mediante redes neuronales basadas en pulsos.

- Se ha establecido una comparación entre los sistemas sensomotores biológicos y los sistemas artificiales, aportando esquemas de conversión biológico/artificial. En especial se ha descrito un mecanismo de conversión de señales neuronales musculares a señales PWM coherentes con el control de servo-motores.

El control de sistemas físicos mediante controladores neuronales basados en pulsos resulta complejo, principalmente por el análisis tanto en la temporización como en la correcta secuencialización de cientos o miles de patrones de pulsos emitidos por las redes neuronales artificiales.

El comportamiento del sistema final y más concretamente la capacidad de control obtenida mediante los sistemas neuronales basados en pulsos, dependen en gran medida de las características sensoriales del sistema físico que se utiliza (entiéndase como una plataforma robotizada multisensorial). Los sistemas biológicos poseen sistemas sensoriales que proporcionan en tiempo real y de forma masiva una "medida" del estado del entorno y del estado del propio individuo difícilmente reproducible por la tecnología actual. Además estos sistemas son auto-reparables y auto-calibrados lo que proporciona una estabilidad funcional óptima al sistema.

A modo de conclusión final, la "evolución" de los sistemas artificiales en lo referente a la parte de control pasa por el desarrollo de sistemas sensoriales mucho más complejos que permitan un grado de

interacción con el entorno y con el propio individuo con mayor resolución.

Capítulo 6
Conclusiones y aportaciones

Este capítulo es un resumen del trabajo presentado en esta memoria de Tesis. En él se discuten los principales resultados obtenidos, la aportación científica, así como la innovación que representa.

6.1 Marco de la investigación realizada

Antes de comenzar el trabajo de esta Tesis doctoral se afrontó el desarrollo de distintos robots y con ello se adquirió experiencia en este campo sobre la dificultad de "creación de nuevos agentes" y retos en cuanto a esquemas nuevos de control. Este esfuerzo previo ha motivado, en gran parte, el trabajo presentado en esta memoria.

La Tesis se ha realizado en el marco de dos proyectos europeos:

- **SpikeFORCE:** Real-Time Spiking Networks for Robot Control (FP5-IST-2001-35271). (01-05-2002 hasta 30-09-2005)

- **SENSOPAC:** Sensory motor structuring of perception and action for emerging cognition (FP6-IST-028056). (01-01-2006 hasta 30-12-2010)

El objetivo de SpikeFORCE era desarrollar modelos de estructuras neuronales biológicamente plausibles de pulsos y estudiar mecanismos para su utilización en tareas de control de Robots. SENSOPAC va un paso más allá y pretende estudiar cómo, utilizando robots biomórficos y estructuras de control bio-inspiradas, podemos evaluar mecanismos eficientes de control y sensado activo para tareas de exploración. En este proyecto se estudia cómo se estructuran las "nociones cognitivas" en forma de modelos "abstraídos" de los

sistemas biológicos como por ejemplo: el ciclo cerebelo-oliva inferior así como otros centros nerviosos.
En el marco de ambos proyectos el desarrollo de tecnologías de simulación de redes neuronales de pulsos eficientes para su utilización en tareas de control de robots (en tiempo real), constituye un pilar central. Además, el estudio de nuevas formas de computación radicalmente distintas a las convencionales (entiéndase ésta última como tecnología de procesadores secuenciales) tiene un interés importante en si mismo. En este ámbito se pueden considerar las "arquitecturas de procesamiento" presentadas en este trabajo como arquitecturas escalables en donde la información de entrada salida son pulsos de neuronas. La arquitectura de procesamiento está compuesta por procesadores de propósito específico con una sola "instrucción": procesar un pulso.
Este tipo de arquitecturas permite estudiar la escalabilidad y nuevas formas de paralelización que tienen interés actualmente dado el gran número de recursos computacionales de los que podemos disponer en un solo circuito integrado basado en hardware reconfigurable.

6.2 Conclusiones

La mayor contribución de este trabajo consiste en el desarrollo de dos arquitecturas de procesamiento eficiente de pulsos neuronales para la simulación de sistemas neuronales biológicamente plausibles de tamaño mediano en tiempo real. Se ha desarrollado una aproximación dirigida por tiempo con una implementación híbrida software/hardware en donde se han desarrollado módulos de computación hardware en FPGA, primitivas de computación y módulos software ejecutados en un computador convencional conectado a la placa co-procesadora.
La otra aproximación dirigida por eventos es radicalmente distinta. En este caso todo el sistema se simula en la placa co-procesadora y el computador se utiliza meramente como herramienta de inicialización y monitorización. Esta segunda aproximación permite empotrar el motor de simulación en robots reales sin necesidad de conexión con computadores convencionales.

Los conceptos que se han estudiado, con este tipo de motores de simulación, precisan de tecnologías de control de robots en tiempo

real. Para ello se han desarrollado y complementado dos plataformas robóticas y los módulos de comunicación con el motor de simulación. Esta contribución es un buen ejemplo del trabajo para afrontar la convergencia de tecnologías de robótica (motores, sensores), tecnologías de computación de altas prestaciones (hardware reconfigurable y procesamiento paralelo) y esquemas de control bio-inspirado.

Los resultados fundamentales de esta Tesis no son simulaciones concretas sino medición de prestaciones de los motores de simulación desarrollados, validación de las plataformas robóticas desarrolladas así como los esquemas de comunicación.
El trabajo presentado en esta Tesis doctoral abre las puertas a futuras investigaciones relacionadas con sistemas biológicos como el cerebelo como objetivos de simulación.

6.3 Producción científica

A lo largo de este trabajo de investigación se han publicado los siguientes trabajos científicos:

Artículos con índice de impacto (SCI):

1. Eduardos Ros, Eva M. Ortigosa, Rodrigo Agis, Richard Carrillo and Michael Arnold, "**Real-Time Computing Platform for Spiking Neurons (RT.Spike)**", IEEE transactions on Neural Networks, July 2006, Volume 17, Number 4, pp. 1050-1063, (ISSN 1045-9227).

2. E. Ros, R. Carrillo, E. M. Ortigosa, B. Barbour, and R. Agís. **"Event-Driven Simulation Scheme for Spiking Neural Networks Using Lookup Tables to Characterize Neuronal Dynamics"**. Neural Computation. (2006) 18(12): 2959-2993.

3. Rodrigo Agís, Javier Díaz, Eduardo Ros, Richard Carrillo, Eva. M. Ortigosa, "**Hardware Event-driven Simulation Engine for Spiking Neural Networks**", International Journal of Electronics (Taylor & Francis Group), Volume 94, Issue 5, May 2007, pp. 469-480.

Artículos presentados en congresos y publicados en Lectures Notes in Computer Science (también con SCI):

1. Eduardo Ros, Rodrigo Agis, Richard R. Carrillo, Eva M. Orticosa. "**Post-synaptic Time-Dependent Conductances in Spiking Neurons: FPGA Implementation of a Flexible Cell Model**". 7th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2003 Proceedings, Part II, pp. 145-152, (ISBN: 3-540-40211-X). Maó, Menorca, Spain, June 3-6, 2003. Lecture Notes in Computer Science, vol 2686, Springer 2003.

2. Eva M. Ortigosa, Pilar M. Ortigosa, Antonio Cañas, Eduardo Ros, Rodrigo Agís and Julio Ortega Ac, "**FPGA Implementation of Multi-Layer Perceptrons for Speech Recognition**", FPL: 13th international conference on field programmable logic and applications Lisbon - Portugal, September 1-3, 2003.

3. Eduardo Ros, Eva M. Ortigosa, Rodrigo Agis, Richard Carrillo, Alberto Prieto and Mike Arnold. "**Spiking Neurons Computing Platform"**. 8th Internacional Work Conference on Artificial Neural Networks, IWANN 2005, pp470-478, (ISBN-13: 978-3-540-26208-4). Vilanova I la Geltrú, Barcelona, Spain, June 8-10, 2005.

4. R. Carrillo, Eduardo Ros, Eva M Ortigosa, Boris Barbour and Rodrigo Agis. "**Lookup Table Powered Neural Event-Driven Simulator"**, 8th Internacional Work Conference on Artificial Neural Networks, IWANN 2005, pp168-175, (ISBN-13: 978-3-540-26208-4). Vilanova I la Geltrú, Barcelona, Spain, June 8-10, 2005.

5. Rodrigo Agís, Javier Díaz, Eduardo Ros, Richard Carrillo, Eva. M. Ortigosa, **"Event-driven simulation engine for spiking neural networks on a chip"**, International workshop on applied reconfigurable computing (ARC2006) delft, the Netherlands, march 1-3, 2006. Lecture Notes in Computer Science. Vol: 3985, pp 36-45 ISBN-10 3-540-36708-X Springer Berlinb Heideidelberg New York.

6. Silvia Tolu, Eduardo Ros, Rodrigo Agis, **"Bio-inspired control model for object manipulation by humanoid robots"**, 9th Internacional Work Conference on Artificial Neural Networks, IWANN 2007, San Sebastián (Spain) June 20-22, 2007, pp 822-829. ISBN-10 3-540-73006-0 Springer Berlin Heidelberg NewYork.

Otras aportaciones a conferencias científicas:

1. Rodrigo Agis, Richard Carrillo, Antonio Cañas, Begoña del Pino y Francisco J. Pelayo, **Entorno hardware-software para experimentación basado en un micro-robot**. Actas de las II Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA'2002), pp. 261-266, (ISBN: 84-699-9448-4), Almuñecar (Granada), 18-20 Septiembre 2002.

2. Agis R., Ros E., Carrillo R., Orticosa. E.M., Pelayo F.J, Prieto A., **Implementación en PFGAs de una plataforma de simulación de Neuronas de Pulsos**. III Jornadas sobre computación reconfigurable y aplicaciones pp. 301-308, (ISBN: 84-600-9928-8), Escuela Politécnica Superior Universidad autónoma de Madrid. Madrid, 10, 11 y 12 de septiembre de 2003.

3. Agis R., Ros E., Díaz J., Mota S., Carrillo R., Orticosa E, Pelayo F., Prieto A., **Sistema de control basado en visión y propiacepción de robots con FPGA**. FPGAs: Computación y aplicaciones, pp. 667-674, (ISBN: 84-688-7667-4). Barcelona 13-15 de Septiembre del 2004. JCRA. 2004.

4. O.J.M. Coenen, C. Boucheny, M. Bezzi, D. Marchal, M.P. Arnold, E. Ros, R. Carillo, E.M. Ortigosa, R. Agis, B. Barbour, A. Arleo, T. Nieus, E. D'Angelo, **Adaptive spiking cerebellar models and real-time simulations**, In: Society for Neuroscience Abstracts, No. 827.4, San Diego, USA. 2004.

5. R. Agis, R. Carrillo, V. Moran, A.Gonzalez, C. Morillas, F. Pelayo and J.L.Bernier, **Monitoring a mobile robot using a web interface,** III International conference on Multimedia and ICTs in Education (mICTE2005), pp. 1288-1293, Vol III , (ISBN: 84-609-5994-5). Cáceres, June 7-10 2005.

6. Rodrigo Agis, Eduardo Ros, F.J.Pelayo, Máximo Barbaro, Giani Mereu, Luigi Raffo. "**Prototipo de sistema de comunicación SPI para dispositivos neuromórficos"**, Dentro del congreso español de informática (CEDI 2005), Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA 2005), pp 333-337, (ISBN:84-9732-439-0). Granada del 13-16 de septiembre del 2005.

7. Rodrigo Agis, Eduardo Ros, F.J.Pelayo, Richard Carrillo, Eva Ortigosa, Rafael Rodríguez, Javier Díaz, Sonia Mota. **"Sistema evolutivo de control de un brazo: Optimización de constantes dinámicas"**. Dentro del congreso español de informática (CEDI 2005), Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA 2005), pp 311-316, (ISBN:84-9732-439-0). Granada del 13-16 de septiembre del 2005.

8. Rodrigo Agis Melero, Facncisco F. Franco Ramón, Francisco J. Pelayo Valle, Eduardo Ros Vidal, Christian A. Morillas Gutiérrez, "Wellmerit: Robot experimental con visión estéreo" Computación reconfigurable y aplicaciones (JCRA 2006), pp. 147-151, (ISBN-10:84-611-1315-2). Cáceres 12, 13, 14 de septiembre 2006.

6.4 Aportaciones

Las principales aportaciones de este trabajo son las siguientes:

1. Partiendo de modelos neuronales biológicamente plausibles se ha establecido un modelo de referencia que constituye un buen compromiso entre fidelidad al modelo biológico, complejidad de computación y posibilidades de particularización mediante ajuste de algunas de las funciones (decaimiento pasivo, inyección gradual de carga, etc.)

2. Partiendo del modelo anterior, se ha definido una arquitectura de procesamiento de pulsos neuronales dirigida por tiempo, diseñando un cauce segmentado de datos para aprovechar mejor los recursos de computación, inherentemente paralelos, de los

dispositivos FPGA. Se ha estructurado todo el sistema de datos (pulsos neuronales) en distintas configuraciones de memoria empotrada o en placa para facilitar su acceso paralelo. Esto hace que la arquitectura desarrollada sea totalmente escalable y las prestaciones puedan multiplicarse replicando unidades funcionales en el mismo chip.

3. Se ha diseñado toda una plataforma híbrida software/hardware para la utilización del motor de simulación dirigido por tiempo. La topología de la red y el aprendizaje se simulan en software mientras que la actualización de estados neuronales se realiza en hardware dedicado en la placa co-procesadora (arquitectura de procesamiento desarrollada). Se ha comprobado la ganancia en prestaciones gracias a la utilización de la placa co-procesadora desarrollada con una simulación de un cerebelo artificial en tiempo real.

4. Se ha desarrollado una arquitectura de procesamiento de pulsos neuronales dirigida por eventos. En este caso todo el sistema se ha implementado en el dispositivo FPGA (con varios módulos de memoria externos). Se han desarrollado módulos software para la mera monitorización de las simulaciones. Se han depurado los diferentes riesgos en el cauce segmentado de datos. El hecho de implementar todo el sistema en la plataforma reconfigurable facilita su utilización como sistema de control empotrado. Se han obtenido prestaciones sobresalientes de más de dos millones de pulsos procesados por segundo.

5. Se han diseñado y complementado dos plataformas robóticas con módulos de comunicación específicos para los motores de simulación desarrollados. En estas plataformas se ha facilitado el acceso directo en tiempo real a sensores y motores. Estas plataformas constituyen herramientas de validación muy útiles para esquemas de procesamiento bio-inspirados en los que se requiere la interacción continua (tiempo real) del sistema con el entorno.

6. Estableciendo un paralelismo entre las funciones musculares motoras y la dinámica de sistemas basados en servomotores, se ha propuesto e implementado un mecanismo de conversión de

señales que permite ek control de movimientos del robot bípedo mediante el motor de simulación dirigido por eventos.

6.5 Trabajo futuro

El trabajo futuro en este campo es muy diverso. Por un lado el trabajo presentado constituye una herramienta muy válida para la simulación de sistemas neuronales biológicamente plausibles. En este trabajo se ha presentado la tecnología de simulación desarrollada (arquitecturas de procesamiento de propósito específico) y como trabajo futuro, su utilización para simulaciones concretas de sistemas como el cerebelo, la oliva inferior, etc. Además se han desarrollado y complementado plataformas robóticas que permiten experimentar con la abstracción de primitivas sensorimotoras en modelos y esquemas de control basados en el principio de percepción-acción en ciclo cerrado con agentes reales (“Embodiment”). Esto hace que esta tecnología tenga un gran potencial en el campo de la robótica y esquemas de control bio-inspirados empotrados en agentes activos.

Capítulo 7
Conclusions and main contributions

This chapter is a summary of the work presented in this PhD Thesis. Here we discuss the main results obtained, the scientific outcome, and the innovation points addressed.

7.1 Framework of the research task

Before starting the work of this PhD different robotic platforms were developed that helped the author to gain expertise in this field. Furthermore we realised about the inherent difficulty of "creating new agents" and the challenge of devising new control schemes. This previous creative effort has highly motivated the work presented in this Thesis.

The Thesis has been done in the framework of two EU projects:

- **SpikeFORCE:** Real-Time Spiking Networks for Robot Control (FP5-IST-2001-35271). (01-05-2002 till 30-09-2005)

- **SENSOPAC:** Sensory motor structuring of perception and action for emerging cognition (FP6-IST-028056). (01-01-2006 till 30-12-2010)

The goal of SpikeFORCE was to develop models of spiking neural structures biologically plausible and study mechanisms for their utilization in robot control tasks. SENSOPAC represents a step beyond the control task and addresses the study of how, using biomorphic robots and bio-inspired control schemes we can evaluate efficient control and active sensing strategies for exploration tasks. In this project is studied how "cognitive notions" are structured by

means of abstracted models in biological models, for instance in olivo-cerebellar structures and other neural subsystems.

For both projects the development of efficient spiking neural network simulation technologies to be used in real-time robot control tasks represents an issue of critical importance. Furthermore, the study of new computation schemes radically different to the conventional sequential processors based techniques remains as breaking through point in itself. The new "computing architectures" presented in this work can be considered as scalable architectures in which the input/output information is represented by means of neural spikes. The processing architecture is a specific purpose processor with a single "instruction": process a spike.
In this kind of architectures in possible to study the scalability issue and new parallelization techniques that become of specific interest provided the large number of computing resources currently available on single chips (reconfigurable hardware devices).

7.2 Conclusions

The main contribution of this work consists in the development of two efficient processing architectures for real-time medium-scale spiking neural networks simulations (of biologically plausible neural systems). A time-driven approach has been developed with a hybrid Hardware/Software implementation. Here we have designed hardware computing modules as specific purpose hardware in FPGA and software modules that are run on a host conventional computer in which the hardware co-processing board is plugged in.

The other approach driven by events is radically different. In this case all the system is simulated in the co-processing board and the host computer is only used as initialization and monitorization tool. This second approach allows embedding the simulation engine in real robots without requiring the connection to conventional computers. Therefore it represents a stand-along platform.
The concepts that are studied, with this kind of simulation engines, require real-time robot control technologies. For this purpose, the work here presented describes how two robotic platforms have been developed and complemented with the communication modules to

interface in real time the simulation engines. This contribution is a good example of the work that needs to be faced to integrate the different robotic technologies (motors and sensors), high performance computing technologies (reconfigurable hardware and parallel processing) and bio-inspired control schemes.

The main results of this Thesis are not specific simulations, but rather the performance of the simulation engines developed and the validation of the robotic platforms presented (including the communication schemes).
The work presented in this PhD Thesis opens the door to future research issues related with biologically plausible systems such as the cerebellum as simulation goal.

7.3 Scientific production

Along this research work different scientific publications have been produced:

Papers with Scientific Impact (SCI):

1. Eduardos Ros, Eva M. Ortigosa, Rodrigo Agis, Richard Carrillo and Michael Arnold, "**Real-Time Computing Platform for Spiking Neurons (RT.Spike)**", IEEE transactions on Neural Networks, July 2006, Volume 17, Number 4, pp. 1050-1063, (ISSN 1045-9227).

2. E. Ros, R. Carrillo, E. M. Ortigosa, B. Barbour, and R. Agís. **"Event-Driven Simulation Scheme for Spiking Neural Networks Using Lookup Tables to Characterize Neuronal Dynamics"**. Neural Computation. (2006) 18(12): 2959-2993.

3. Rodrigo Agís, Javier Díaz, Eduardo Ros, Richard Carrillo, Eva. M. Ortigosa, "**Hardware Event-driven Simulation Engine for Spiking Neural Networks**", International Journal of Electronics (Taylor & Francis Group), Volume 94, Issue 5, May 2007, pp. 469-480.

4. **Papers presented in conferences and Publisher in Lectures Notes in Computer Science (also with SCI):**

5. Eduardo Ros, Rodrigo Agis, Richard R. Carrillo, Eva M. Orticosa. "**Post-synaptic Time-Dependent Conductances in Spiking Neurons: FPGA Implementation of a Flexible Cell Model**". 7th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2003 Proceedings, Part II, pp. 145-152, (ISBN: 3-540-40211-X). Maó, Menorca, Spain, June 3-6, 2003. Lecture Notes in Computer Science, vol 2686, Springer 2003.

6. Eva M. Ortigosa, Pilar M. Ortigosa, Antonio Cañas, Eduardo Ros, Rodrigo Agís and Julio Ortega Ac, "**FPGA Implementation of Multi-Layer Perceptrons for Speech Recognition**", FPL: 13th international conference on field programmable logic and applications Lisbon - Portugal, September 1-3, 2003.

7. Eduardo Ros, Eva M. Ortigosa, Rodrigo Agis, Richard Carrillo, Alberto Prieto and Mike Arnold. "**Spiking Neurons Computing Platform"**. 8th Internacional Work Conference on Artificial Neural Networks, IWANN 2005, pp470-478, (ISBN-13: 978-3-540-26208-4). Vilanova I la Geltrú, Barcelona, Spain, June 8-10, 2005.

8. R. Carrillo, Eduardo Ros, Eva M Ortigosa, Boris Barbour and Rodrigo Agis. "**Lookup Table Powered Neural Event-Driven Simulator"**, 8th Internacional Work Conference on Artificial Neural Networks, IWANN 2005, pp168-175, (ISBN-13: 978-3-540-26208-4). Vilanova I la Geltrú, Barcelona, Spain, June 8-10, 2005.

9. Rodrigo Agís, Javier Díaz, Eduardo Ros, Richard Carrillo, Eva. M. Ortigosa, **"Event-driven simulation engine for spiking neural networks on a chip"**, International workshop on applied reconfigurable computing (ARC2006) delft, the Netherlands, march 1-3, 2006. Lecture Notes in Computer Science. Vol: 3985, pp 36-45 ISBN-10 3-540-36708-X Springer Berlinb Heideidelberg New York.

10. Silvia Tolu, Eduardo Ros, Rodrigo Agis, **"Bio-inspired control model for object manipulation by humanoid robots"**, 9th Internacional Work Conference on Artificial Neural Networks, IWANN 2007, San Sebastián (Spain) June 20-22, 2007, pp 822-829. ISBN-10 3-540-73006-0 Springer Berlin Heidelberg NewYork.

Other contributions in scientific conferences:

1. Rodrigo Agis, Richard Carrillo, Antonio Cañas, Begoña del Pino y Francisco J. Pelayo, **Entorno hardware-software para experimentación basado en un micro-robot**. Actas de las II Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA'2002), pp. 261-266, (ISBN: 84-699-9448-4), Almuñecar (Granada), 18-20 Septiembre 2002.

2. Agis R., Ros E., Carrillo R., Orticosa. E.M., Pelayo F.J, Prieto A., **Implementación en PFGAs de una plataforma de simulación de Neuronas de Pulsos**. III Jornadas sobre computación reconfigurable y aplicaciones pp. 301-308, (ISBN: 84-600-9928-8), Escuela Politécnica Superior Universidad autónoma de Madrid. Madrid, 10, 11 y 12 de septiembre de 2003.

3. Agis R., Ros E., Díaz J., Mota S., Carrillo R., Orticosa E, Pelayo F., Prieto A., **Sistema de control basado en visión y propiacepción de robots con FPGA**. FPGAs: Computación y aplicaciones, pp. 667-674, (ISBN: 84-688-7667-4). Barcelona 13-15 de Septiembre del 2004. JCRA'2004

4. O.J.M. Coenen, C. Boucheny, M. Bezzi, D. Marchal, M.P. Arnold, E. Ros, R. Carillo, E.M. Ortigosa, R. Agis, B. Barbour, A. Arleo, T. Nieus, E. D'Angelo, **Adaptive spiking cerebellar models and real-time simulations**, In: Society for Neuroscience Abstracts, No. 827.4, San Diego, USA. 2004.

5. R. Agis, R. Carrillo, V. Moran, A.Gonzalez, C. Morillas, F. Pelayo and J.L.Bernier, **Monitoring a mobile robot using a web interface,** III International conference on Multimedia and ICTs in Education (mICTE2005), pp. 1288-1293, Vol III , (ISBN: 84-609-5994-5). Cáceres, June 7-10 2005.

6. Rodrigo Agis, Eduardo Ros, F.J.Pelayo, Máximo Barbaro, Giani Mereu, Luigi Raffo. "**Prototipo de sistema de comunicación SPI para dispositivos neuromórficos"**, Dentro del congreso español de informática (CEDI 2005), Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA 2005), pp 333-337, (ISBN:84-9732-439-0). Granada del 13-16 de septiembre del 2005.

7. Rodrigo Agis, Eduardo Ros, F.J.Pelayo, Richard Carrillo, Eva Ortigosa, Rafael Rodríguez, Javier Díaz, Sonia Mota. **"Sistema evolutivo de control de un brazo: Optimización de constantes dinámicas"**. Dentro del congreso español de informática (CEDI 2005), Jornadas sobre Computación Reconfigurable y Aplicaciones (JCRA 2005), pp 311-316, (ISBN:84-9732-439-0). Granada del 13-16 de septiembre del 2005.

8. Rodrigo Agis Melero, Facncisco F. Franco Ramón, Francisco J. Pelayo Valle, Eduardo Ros Vidal, Christian A. Morillas Gutiérrez, "Wellmerit: Robot experimental con visión estéreo" Computación reconfigurable y aplicaciones (JCRA 2006), pp. 147-151, (ISBN-10:84-611-1315-2). Cáceres 12, 13, 14 de septiembre 2006.

7.4 Main original contributions

The main original contributions of this work are the following:

1. Based on biologically plausible neural models a reference model has been adopted. This constitutes a trade-off between biological plausibility, computation complexity and customization capability adjusting specific functions and parameters (passive membrane potential decay, gradual injection of charge, etc.).

2. Based on the previous model, a time-driven processing architecture for simulating spiking neurons has been developed. This has been done designing a segmented datapath in order to take full advantage of the inherent parallelism of the computing resources of the FPGA devices. The system data (mainly neural

spikes) has been structured in diverse configurations based on on-chip embedded memory and extended SRAM memory chips on-board. This allows the architecture to be fully scalable and the performance can be multiplied replicating functional units on the same chip.

3. A hybrid Software/Hardware platform has been designed for the utilization of the time-driven simulation engine. The network topology and the learning are simulated in software while the neural state variables are updated through dedicated hardware in the co-processing board (developed processing architecture). The gain in performance provided by the use of the co-processing board has been evaluated simulating in real-time an artificial cerebellum.

4. A spiking neural processing architecture driven by events has been developed. In this case the whole system has been implemented in the FPGA device (supported by several external memory chips). Software modules have been developed for monitoring the simulations. The different risks in the segmented datapath have been debugged. The whole system has been implemented in the reconfigurable platform. This facilitates its use as embedded control system. Outstanding performance rates (with more than two million spikes per second) have been obtained.

5. Two robotic platforms have been designed and complemented with specific communication modules for the developed simulation engines. In these platforms the direct access in real-time to sensors and motors has been facilitated. These platforms constitute very useful validation tools for bio-inspired processing schemes in which the continuous (real-time) interaction of the agent with its environment is required.

6. Making an analogy between the muscle functions in motor tasks and the dynamics of systems based on servo motors, a conversion mechanism of signals has been implemented to allow the movement control of the biped robot through the simulation engine.

7.5 Future work

The future work in this field is very diverse. On one hand, the presented work constitutes a very valid tool for the simulation of biologically plausible neural systems. In this work, the simulation technology developed has been presented (specific purpose processing architectures) and as future work remains its utilization for further concrete simulations of systems such as the cerebellum, the inferior olive, etc. Besides, robotic platforms have been developed and complemented allowing the experimentation of schemes for abstracting models of sensorimotor primitives and control schemes based on closed-loop perception-action cycles using real agents ("Embodiment"). This gives this technology a high potential in the robotic field and the exploration of bio-inspired control schemes with active agents.

Referencias Bibliográficas

[1] A. Jahnke, T. Schoenauer, U. Roth, K. Mohraz, H. Klar, "Simulation of Spiking Neural Networks on Different Hardware Platforms," ICANN97, (Springer-Verlag), Lecture Notes in Computer Science, vol. 1327, pp. 1187-1192, 1997.

[2] Aaditya V., Rangan, David Cai, "Fast numerical methods for simulating large-scale integrate-and-fire neuronal networks", Journal of Computational Neuroscience, Volume 22, Number 1. February, 2007.

[3] Agis R., Ros E., Díaz J., Mota S., Carrillo R., Orticosa E, Pelayo F., Prieto A., "Sistema de control basado en visión y propiacepción de robots con FPGA". FPGAs: Computación y aplicaciones, pp. 667-674, (ISBN: 84-688-7667-4). Barcelona 13-15 de Septiembre del 2004. JCRA 2004.

[4] Analog Devices . [Online]. http://www.analog.com/en/

[5] Atmel . [Online]. http://www.atmel.com/

[6] B.Yu and L. Zhang, "Pulse-coupled neural networks for contour and motion matchings," IEEE Transactions on Neural Networks, vol. 15(5), pp. 1186-1201, 2004.

[7] Boucheny, R. Carrillo, E. Ros, O. J-M D. Coenen, "Real-Time spiking neural network : an adaptive cerebellar model," Lecture Notes in Computer Science, vol. 3512, pp. 136-144, 2005.

[8] Brown, Guy. "The Energy of Life". New York: Free Press, 1999.

[9] C.M. Lin and Y.F. Peng, "Missile guidance law design using adaptive cerebellar model articulation controller", IEEE Transactions on Neural Networks, vol, 16 (3), pp. 636-644, 2005.

[10] Carlson, Niel. A. (1992). "Foundations of Physiological Psychology". Needham Heights, Massachusetts: Simon & Schuster. pp. 48

[11] Carsten Moschner1, Trevor J. Crawford2,3, Wolfgang Heide1, Peter Trillenberg1, Detlef Kömpf1 and Christopher Kennard3. "Deficits of smooth pursuit initiation in patients with degenerative

cerebellar lesions". Brain, Vol. 122, No. 11, 2147-2158, Oxford University Press November 1999.

[12] Celoxica, 2001-2004. [Online]. Available: http://www.celoxica.com.

[13] Compte Albert, Brunel Nicolas, Goldman-Rakic Patricia S, Wang Xiao-Jing. "Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model". Cerebral Cortex sep 2000, pp. 910-923.

[14] Control de velocidad mediante PWM "Frenado regenerativo". [Online]. http://www.isa.cie.uva.es/proyectos/servos/info/PWM/PWM.htm.

[15] D.J. Rossi, M. Hamann, "Spillover-mediated transmission at inhibitory synapses promoted by high affinity alpha6 subunit GABA(A) receptors and glomerular geometry", Neuron, vol.20(4), pp. 783-795, 1998.

[16] De Zeeuw CI, Simpson JI, Hoogenraad CC, Galjart N, Koekkoek SK, Ruigrok TJ. "Microcircuitry and function of the inferior olive". Trends in Neurosciences, Volume 21, Issue 9, 1 September 1998, Pages 391-400

[17] Delorme, A., Gautrais, J. van Rullen, R., Thorpe, S. SpikeNET. "A simulator for modelling large networks of integrate and fire neurons". Neurocomputing, Vols. 26-27, pp. 989-996. 1999.

[18] Delorme, A., Thorpe, S. SpikeNET. "An event-driven simulation package for modelling large networks of spiking neurons". Network: Computation in Neural Systems, Vol. 14, pp. 613-627. 2003

[19] Detlef Heck and Fahad Sultan, Revista "Investigación y Ciencia", Abril, 2002.

[20] Drubach, Daniel. "The Brain Explained". New Jersey: Prentice-Hall, 2000.

[21] E. Bruce Goldstein, "Sensación y Percepción", Capítulo 13, Percepción del dolor: activación neuronal e influencias cognogtivistas, 6ª edición.

[22] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" IEEE Transactions on Neural Networks, vol. 15(5), pp. 1063-1070, 2004.

[23] E. Ros, F.J. Pelayo, I. Rojas, F.J. Fernández, A. Prieto, "A VLSI Approach for Spike Timing Coding", Lecture Notes in Computer Science, vol.1607, pp.166-175, ISSN:0302-9743, Springer-Verlag, 1999.

[24] E.L. Graas, E.A. Brown and R.H. Lee, "An FPGA-based approach to high-speed simulation of conductance-based neuron models", Neuroinformatics, vol. 2., pp. 417-435, 2004.

[25] Eduardos Ros, Eva M. Ortigosa, Rodrigo Agis, Richard Carrillo and Michael Arnold, "Real-Time Computing Platform for Spiking Neurons (RT.Spike)", IEEE transactions on Neural Networks, July 2006, Volume 17, Number 4, pp. 1050-1063, (ISSN 1045-9227).

[26] Electronics Research Laboratory, Delft University of Technology (TUDelft), http://duteela.et.tudelft.nl/~elca/XilinxTutor/documents/dcmotor.pdf

[27] Encoder.[Online]. http://www.seattlerobotics.org/encoder/200009/S3003C.html

[28] F. A. Briggs and E. S. Davidson, "Organization of semiconductor memories for parallel-pipelined processors," IEEE Trans. Comput., vol. C-26, pp. 162-169, Feb. 1977.

[29] Física Clásica y Moderna. W.E. Gettys (Editorial McGraw-Hill) ISBN: 8476156359.

[30] Fujimoto Y., Fukuda N., Akabane T., "Massively parallel architectures for large scale neural network simulations". IEEE transactions on neural networks , 1992, vol. 3, n^{o}6, pp. 876-888.

[31] Fujitsu Robotics. [Online]. http://www.fujitsu.com/global/about/rd/200506hoap-series.html.

[32] G. La Camera, W. Seen, S. Fusi, "Equivalent networks of conductance- and current-driven neurons", Lecture Notes in Computer Science, vol. 2714, pp. 449-452, 2003.

[33] G. La Camera, W. Senn, S. Fusi, "Comparison between networks of conductance- and current-driven neurons: stationary spike rates and subthreshold depolarization", Neurocomputing, vol. 58-60, pp. 253-258, 20

[34] G. Le Masson, S. R. Le Masson, D. Debay, T. Bal, "Feedback inhibition controls spike transfer in hybrid thalamic circuits", Nature, vol. 417, pp. 854-858, 2002.

[35] G.Hartmann, G. Frank, M. Schaefer, C. Wolff, "SPIKE128K- An Accelerator for Dynamic Simulation of Large Pulse-Coded Networks," MicroNeuro'97, pp. 130-139, 1997.

[36] Gall, F. Prestori, E. Sola, A. D'Errico, C. Roussel, L. Forti, P. Rossi, E. D'Angelo, "Intracellular calcium regulation by burst discharge determines bidirectional long-term synaptic plasticity at the cerebellum input stage", The Journal of Neuroscience, vol. 25, pp. 4813-4822, 2005.

[37] Gerstner, W., Kistler, W.; Spiking Neuron Models. Cambridge University Press, (2002).

[38] Ghani, A, McGinnity TM, Maguire LP, Harkin J, (Sep 2006) "Area Efficient Architecture for Large Scale Implementation of Biologically Plausible Spiking Neural Networks on Reconfigurable Hardware", Proceedings of Field Programmable Logic and Application (FPL) , Madrid, Spain, September, Springer-Verlag.

[39] Glackin B., McGinnity T.M., Maguire L.P., Wu Q.X., Belatreche A., "A Novel Approach for the Implementation of Large Scale Spiking Neural Networks on FPGA Hardware", LNCS, pp. 552-563, 2005.

[40] Graphviz Graph Visualization Software , [Online], http://www.graphviz.org/About.php

[41] Grossberg, S. (1988). "Nonlinear neural networks: Principles, mechanisms, and architectures". Neural Networks, 1, 17-61. Reprinted in Carpenter, G.A. and Grossberg, S. (1991). Pattern Recognition by Self-Organizing Neural Networks. Cambridge, MA: MIT Press.

[42] H. H. Hellmich and H. Klar, "SEE: a concept for an FPGA based Emulation Engine for Spiking Neurons with Adaptive Weights," 5th WSEAS International Conference on Neural Networks and Applications (NNA '04), Udine (Italy), 2004, pp. 930-935, 2004.

[43] HEBB, D. O. (1949). "The Organization of Behavior". New York: John Wiley.

[44] Heik H. Hellmich and H. Klar:"SEE: a Concept for an FPGA based Emulation Engine for Spiking Neurons with Adaptive Weights", 5th WSEAS International Conference on Neural Networks and Applications (NNA), Udine (Italy), 25-27 March 2004, Page(s): 930-935.

[45] Hoare, C.A.R. "Quicksort", Computer Journal, pp. 10-15, April 1962.

[46] Hodgkin, A., and Huxley, A. (1952): "A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117. pp. 500-544.

[47] http://topographica.org/Home/index.html.

[48] Informática evolutiva: Algoritmos Genéticos. J. Julián Merelo. [Online]. http://geneura.ugr.es/ ~jmerelo/ie/ags.html

[49] J. Hill, W. McColl, D. Stefanescu, M. Goudreau, K. Lang, S. Rao, T. Suel, T. Tsantilas, R. Bisseling, "BSPlib: the BSP Programming Library," Parallel Computing, vol. 24(14), pp. 1947-1980, 1998.

[50] J. Reutimann, M. Giugliano, S. Fusi, "Event-driven simulation of spiking neurons with stochastic dynamics," Neural Computation, vol. 15, pp. 811-830, 2003.

[51] 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)

[52] Jahnke, T. Schoenauer, U. Roth, K. Mohraz, H. Klar, "Simulation of Spiking Neural Networks on Different Hardware Platforms," LNCS, vol. 1327, pp. 1187-1192, 1997.

[53] Jan L. Souman, Ignace Th.C. Hooge and Alexander H. Wertheim "Localization and motion perception during smooth pursuit eye movements". Biomedical and Life Sciences , Springer Berlin / Heidelberg. Volume 171, Number 4, pp 448-458, June 2006.

[54] Janke, U. Roth, H. Klar, "A SIMD/dataflow architecture for a neurocomputer for spike-processing neural networks (NESPINN)," Proc. MicroNeuro'96, pp. 232-237, 1996.

[55] Jing Z. Liu, Robert W. Brown, and Guang H. Yue, "A Dynamical Model of Muscle Activation, Fatigue, and Recovery". Biophysical Journal Volume 82 May 2002 pp. 2344–2359

[56] John E. Dowling. "Neurons and networks: an introduction to neurossince". Capítulo 4.

[57] Julia T Choi, Amy J Bastian, "Adaptation reveals independent control networks for human walking", Nature Neuroscience (01 Jul 2007).

[58] Kandel y otros "Principios de Neurociencia", Capítulo 40 " El sistema vestibular". Cuarta edición. 2001

[59] Kandel y otros "Principios de Neurociencia", Capítulo 42 " El cerebelo". Cuarta edición. 2001.

[60] Kandel y otros "Principios de Neurociencia". Capítulo 33: "Organización del movimiento" de Cuarta edición. 2001.

[61] Kandel y otros "Principios de Neurociencia". Capítulo 34: "Unidad de acción motora" pag 684. Cuarta edición. 2001.

[62] Kandel y otros "Principios de Neurociencia". Capítulo 37: "Locomoción". pp. 737.

[63] KUO C. Sistemas de Control Automático. Ed.: Prentice Hall.

[64] L. Steels. "Emergent Functionality in Robotic Agents through On-Line Evolution", in Brooks R.A., Maes, P.(eds.): Artificial Life IV, Proc. of the Fourth Int. Workshop on the Synthesis and Simulation of Living.

[65] L.S. Smith, D. S. Fraser, "Robust sound onset detection using leaky integrate-and-dire neurons with depressing synapses," IEEE Transactions on Neural Networks, vol. 15(5), pp. 1125-1134, 2004.

[66] M. Arnold, "Feedback learning in the olivary-cerebellar system," PhD Thesis, The University of Sydney, 2001.

[67] M. Berends, R. Maex, E. De Schutter, "A detailed three-dimensional model of the cerebellar granular layer," Neurocomputing, vol. 58-60, pp. 587-592, 2004.

[68] M. Bezzi, T. Nieus, O. J.-M. Coenen, E. D'Angelo, "An integrate-and-fire model of a cerebellar granule cell", Neurocomputing, vol. 58-60, pp.593-598, 2004.

[69] M. Ito, "The cerebellum and neural control", New York, Raven Press, 1984.

[70] M. Mattia, P. Del Giudice, "Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses," Neural Computation, vol. 12, pp. 2305-2329, 2000.

[71] M. Shaefer, T. Schoenauer, C. Wolff, G. Hartmann, H. Klar, U. Rueckert, "Simulation of Spiking Neural Networks – architectures and implementations", Neurocomputing, vol. 48, 647-679, 2002.

[72] Maguire LP, McGinnity TM, Glackin B, Ghani, A, Belatreche A, Harkin J, (Oct 2007) "Challenges for Large-scale Implementations of Spiking Neural Networks on FPGAs", Neurocomputing - Special Issue on Hardware of Neurocomputing, Elsevier Science.

[73] Makino, T.: "A Discrete-Event Neural Network Simulator for General Neuron Models". Neural Computation & Applications, 11. pp. 210-223. 2003.

[74] Manwani, A. and Koch, C. (1999). "Detecting and estimating signals in noisy cable structures", I: Neuronal noise sources. Neural Comput., 11: pp.: 1797-1829.

[75] Marian, I., Reilly, R.G. and Mackey, D. "Efficient event-driven simulation for spiking neural networks". Proceedings of 3rd WSES International Conference on: Neural Networks and Applications. Interlaken, Switzerland, February 2002.

[76] Martín-Smith, P.; Pelayo, F.J.; Ros, E.; A.Prieto: "Self-organization by temporal inhibition (SOTI)", Neural Processing Letters, Vol.12, No.3, pp.199-213, Dec. 2000.

[77] Mattia, M., & Del Guidice, P. "Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses". Neural Computation, Vol. 12(10), pp. 2305-2329. 2000.

[78] McMahon TA (1984). Muscles, Reflexes, and Locomotion. Princeton University Press, Princeton, New Jersey.

[79] Morris R, Anderson E, Lynch G, Baudry M (1986). "Selective impairment of learning and blockade of long-term potentiation by an N-methyl-D-aspartate receptor antagonist, AP5". Nature 319 (6056): 774-6.

[80] Multiplex. [Online]. http://www.multiplex-rc.de/

[81] N. Kopell and B. Ermentrout, "Chemical and electrical synapses perform complementary roles in the synchronization on interneu-

ronal networks," Proc. Natl. Acad. Sci. USA, vol. 101(43), pp. 15482-15487, 2004.

[82] N. Mehrtash, D. Jung, H.H. Hellmich, T. Schoenauer, V.T. Lu, H. Klar, "Synaptic Plasticity in Spiking Neural Networks (SP^2INN): A System Approach," IEEE Transactions on Neural Networks, vol. 14(5), 2003.

[83] N. Mehrtash, D. Jung, H.H. Hellmich, T. Schoenauer, V.T. Lu, H. Klar, "Synaptic Plasticity in Spiking Neural Networks (SP^2INN): A System Approach," IEEE Trans.Neural Networks, Vol. 14(5), 2003.

[84] National Semiconductors: http://www.national.com

[85] O. J.-M.D. Coenen, M. Arnold, T. Sejnowski, M. Jabri, "Parallel fiber coding in the cerebellum for life-long learning", Autonomous Robots, vol. 11 (3), pp. 291-297, 2001.

[86] OGATA K ," Ingeniería de Control Moderna". Ed.: Prentice Hall.

[87] P. Martin-Smith, F.J. Pelayo, E. Ros, A. Prieto: "Supervised VQ Learning based on Temporal Inhibition", Lecture Notes in Computer. Science, vol. 1606, pp.610-620, ISSN: 0302-9743, Springer-Verlag, 1999.

[88] Parallax [Online]. http://www.parallax.com/

[89] Parallelization of cellular neural networks for image processing on cluster architectures. Weishaupl T., Schikuta E. "Parallel Processing Workshops, 2003". Proceedings. 2003 International Conference on Digital Object Identifier, 2003 Page(s): 191-196.

[90] Pelayo,F.J.; Ros,E.; Arreguit,X,; Prieto,A.: "VLSI implementation of a neural model using spikes", Analog Integrated Circuits and Signal Processing. Kluwer Academic Publishers; ISSN: 0925-1030. Vol.13, no.1/2, May/June 1997. pp. 111-121.

[91] Peter Lennie, "The Cost of Cortical Computation", Current Biology, Volume 13, Issue 6, Pages 493-497.

[92] Philipona, O. J.-M.D. Coenen, "Model of granular layer encoding in the cerebellum", Neurocomputing, vol. 58-60, pp. 575-580, 2004.

[93] Porr Bernd, Florentin Wöngotter, "Temporal Sequence Learning, Prediction, and Control: A Review of Different Models and Their Relation to Biological Mechanisms", Neural Computation. Vol 17. pp. 245-319. 2005

[94] Porr, B. Wörgötter. "Isotopic Sequence Order learning", Neural Computation, 15, pp 831-864 (2003).

[95] Porr, B. Wörgötter. "Strongly Improved Stability and Master Convergente of Temporal Sequence Learning by Using Input Correlation Only". Neural computation 18, pp. 1380-1412. 2006.

[96] Protocolo HMI. [Online]. http://www.robonova.de/store/down/download/files/anleitung_8498.pdf

[97] R. Eckhorn, A. M. Gail, A. Bruns, A. Gabriel, B. Al-Shaikhli, M. Saam, "Different types of signal coupling in the visual cortex related to neural mechanisms of associative processing and perception," IEEE Transactions on Neural Networks, vol. 15(5), pp. 1039-1052, 2004.

[98] R. Eckhorn, H.J. Reitboeck, M. Arndt, and P. Dicke, "Feature linking via stimulus evoked oscillations: Experimental results from cat visual cortex and functional implication from a network model," in Proc. ICNN I, pp. 723–720, 1989.

[99] R. Eckhorn, R. Bauer, W. Jordan, M. Brosh, W. Kruse, M. Munk, H.J. Reitboeck, "Coherent oscillations: A mechanism of feature linking in the visual cortex?," Biol. Cyber. Vol. 60, pp. 121-130, 1988.

[100] R. J. Vogelstein, U. Mallik and G. Cauwenberghs, "Beyond event-driven communication: dynamically-reconfigurable spiking neural systems", The Neuromorphic Engineer, vol. 1(1), pp. 1,9, 2004.

[101] R. Maex and E. De Schutter, "Resonant Synchronization in Heterogeneous Networks of Inhibitory Neurons," The Journal of Neuroscience, vol. 23(33), pp. 10503-10514, 2003.

[102] R.A. Silver, D. Colquhoun, S.G. Cull-Candy, B. Edmonds, "Deactivation and desensitization of non-NMDA receptors in patches and the time course of EPSCs in rat cerebellar granule cells", Journal of Physiology, vol. 493(1), pp. 167-173, 1996.

[103] R.R. Carrillo, E. Ros, C. Boucheny, O. J-M D. Coenen, "A cerebellum-like spiking neural network for robot control", 7th international workshop on information precessing in cell and tissues (IPCAT), August 2007, Oxford UK.

[104] Ralph C. Merkle. "Energy limits to the computational power of the human brain", Foresight Update No. 6, August 1989.

[105] Reutimann, J., Guigliano, M., Fusi, S. "Event-driven simulation of spiking neurons with stochastic dynamics". Neural Computation, 15, pp. 811-830. 2003.

[106] Robonova.[Online]. http://www.robonova.com/

[107] Robótica manipuladores y robots móviles. Aníbal Ollero. Capítulos 5 y 8. ISBN: 84-267-131-30

[108] Rodrigo Agís, Javier Díaz, Eduardo Ros, Richard Carrillo, Eva. M. Ortigosa, "Hardware Event-driven Simulation Engine for Spiking Neural Networks"**,** International Journal of Electronics (Taylor & Francis Group), Volume 94, Issue 5, May 2007, pp. 469-480.

[109] Ros, E., Carrillo, R., Ortigosa, E. M., Barbour, B., Agís, R., 2006 Event-driven Simulation Scheme for Spiking Neural Models based on Characterization Look-up Tables. Neural Computation, Vol. 18(12), pp. 2959-2993, 2006.

[110] Ros, E., Ortigosa, E. M., Agis, R., Carrillo, R., Arnold, M., "Real-time computing platform for spiking neurons (RT-Spike)", IEEE transactions on Neural Networks, Vol. 17(4) pp. 1050-1063, 2006.

[111] Ros, E., Ortigosa, E. M., Agis, R., Carrillo, R., Prieto, A., Arnold, M., "Spiking neurons computing platform", LNCS, Vol. 3512, pp. 471-478, 2005

[112] S. J. Mitchell, R.A. Silver, "Shunting inhibition modulates neuronal gain during synaptic excitation," Neuron, vol. 38(3), pp. 433-445, 2003.

[113] S. Le Masson, A. Laflaquiere, T. Bal, G. Le Masson, "Analog circuits for modeling biological neural networks: Design and applications", IEEE Transactions on Biomedical Engineering, vol. 46 (6), pp. 638-645, 1999.

[114] S. Tia, J.K. Wang, N. Kotchabhakdi, S. Vicini, "Developmental changes of inhibitory synaptic currents in cerebellar granule neurons: role of GABA(A) receptor alpha 6 subunit", Journal of Neuroscience, vol. 16(11), pp. 3630-3640, 1996.

[115] SENSOPAC, (EU Project: FP6-IST-028056). [Online] http://www.sensopac.org/

[116] Servos Volz. [Online]. http://www.volz-actuators.com/en/

[117] SpikeFORCE, (EU Project: IST-2001-35271). [Online]. http://www.spikeforce.org

[118] Sugihara, E.J. Lang, and R. LLinas, "Unifrom olivocerebllar conduction time underlies Purkinje cell complex spike synchronicity in the rat cerebellum", Journal of Physiology, vol. 470, pp. 243-271, 1993.

[119] Systems, MIT Press, 1994 L. Steels y P. Vogt, "Grounding adaptative language games in robotic agents", European Conference on Artificial Life, 1997.

[120] T. Schoenauer, S. Atasoy, N. Mehrtash, H. Klar, "NeuroPipe-Chip: A Digital Neuro-Processor for Spiking Neural Networks," IEEE Trans. Neural Networks, vol. 13(1), pp. 205-213, 2002.

[121] Technical Library, Celoxica. [Online]. Disponible en: http://www.celoxica.com/techlib/default.asp.

[122] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest (1990): "Introduction to algorithms". MIT Press / McGraw-Hill.

[123] Thompson, Adedoyin Maria and Porr, Bernd and Florentin Wörgötter. "Stabilising Hebbian Learning with a Third Factor in a Food Retrieval Task", LNCS/LNAI 4095, pp 313-322, Springer 2006.

[124] Tuffy F, McDaid LJ, McGinnity TM, Santos JA, p.kelly@ulster.ac.uk, Sayers H M, (Sep 2006) "Inter-Neuron Communication for Spiking Neural Networks", Neurocomputing, Special Issue on Hardware architectures for Genetic, Neural and Fuzzy Systems, Elsevier.

[125] U. Ernst, K. Pawelzik, T. Geisel, "Synchronization induced by temporal delays in pulse-coupled oscillations," Phys. Rev. Lett., vol. 74, pp. 1570-1573, 1995.

[126] V.Dante, P. Del Giudice and A. M. Whatley, "Hardware and software interfacing to address-event based neuromorphic systems", The Neuromorphic Engineer, vol. 2(1), pp. 5-6, 2005.

[127] Van Vreeswjik, L.F. Abbott, G.B. Ermentrout, "When inhibition not excitation synchronizes neural firing," Journal of Computing Neuroscience, vol. 1, pp. 313-321, 1994.

[128] W.Gerstner, W., Kistler, "Spiking Neuron Models," Cambridge University Press, 2002.

[129] Weishaupl, T.; Schikuta, "How to parallelize cellular neural networks on cluster architectures", E. Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on
Digital Object Identifier, 2004 Page(s). pp.: 439-444.

[130] White, J. A., Rubinstein, J. T., and Kay, A. R. (2000). "Channel noise in neurons". Trends Neuroscience. 23: pp.: 131-137.

[131] Whitlock JR, Heynen AJ, Shuler MG, Bear MF. "Learning induces long-term potentiation in the hippocampus". Science 2006 Aug 25;313(5790):1093-7.

[132] Williams, R and Herrup, K (2001). "The Control of Neuron Number." Originally published in The Annual Review of Neuroscience **11**:423–453 (1988). Last revised Sept 28, 2001.

[133] Winters JM, 1990, "Hill-based muscle models: a systems engineering perspective", "in Multiple Muscle Systems: Biomechanics and Movement Organization", edited by JM Winters and SL Woo, Springer-Verlag, New York.

[134] Wulfram Gerstner, Werner Kistler: "Spiking Neurons Models", Chapter 4, pp. 106-108, (2002), Cambridge University Press. ISBN:0521813840.

[135] Xilinx, [Online]. Available: http://www.xilinx.com.

[136] Z. Nusser, S. Cull-Candy, M. Farrant, "Differences in synaptic GABA(A) receptor number underlie variation in GABA mini amplitude", Neuron, vol. 19(3), pp. 697-709, 1997.

[137] Zajac FE (1989). "Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control". CRC Critical Reviews in Biomedical Engineering, Volume 17, pp. 359-411.

[138] Ziegler, J.G. and Nichols, N.B, "Optimum settings for automatic controllers", Trans. ASME, 1942, 65, pp. 433-444.